University of Birmingham

Formal Verification of Privacy in Pervasive Systems

by

Loretta Ilaria Mancini

A thesis submitted to the University of Birmingham
for the degree of Doctor of Philosophy

School of Computer Science
College of Engineering and
Physical Sciences
University of Birmingham
May 2015

UNIVERSITY OF BIRMINGHAM

Abstract

College of Engineering and Physical Sciences
School of Computer Science

Doctor of Philosophy

Formal Verification of Privacy

in Pervasive Systems

by Loretta Ilaria Mancini

Supervisor: Dr. Eike Ritter

Co-Supervisor: Dr. Myrto Arapinis

Pervasive systems aim to enhance a user's everyday experience by sensing her environment thanks to small electronic devices which can be easily carried in a pocket. However, the use of these physically non intrusive sensing and context aware devices can result very intrusive from a privacy perspective since they give both third party attackers and curious service providers the means to monitor one's activities and hence violate her privacy. An example of pervasive device which most of us is very familiar with is a mobile phone. Mobile telephony equipment is daily carried by billions of subscribers everywhere they go. Avoiding linkability of subscribers by third parties, and protecting the privacy of those subscribers is one of the goals of mobile telecommunication protocols.

We use experimental and formal methods to model and analyse the security properties of mobile telephony protocols. As a result of our experimental and formal analysis, we expose novel threats to the user privacy in mobile telephony systems. These threats make it possible to trace and identify mobile telephony subscribers. For some of the attacks we demonstrate the feasibility of a low cost implementation. We propose fixes to these privacy issues. Moreover, we successfully prove that our privacy-friendly fixes satisfy the desired unlinkability and anonymity properties. We show how it is possible to model unlinkability and anonymity properties, for automatic verification, in case of stateless protocols with no initialisation phase. Moreover, we give a manual proof of unlinkability of a stateful protocol. This is one of the few proofs of bisimilarity available in literature for a full sized protocol. Finally, we develop the first extension of the `ProVerif` tool for the automatic verification of equivalence-based properties of stateful protocols.

This work shows in practice, by applying formal methods to a real world case study, how to formally verify privacy properties of pervasive systems and highlights the limitations of currently available formal tools. Moreover, we take the first steps towards the development of an automatic verification tool for the verification of equivalence-based properties of stateful protocols. Further work in this direction will eventually widen the class of security protocols and security properties verifiable using automatic verification tools.

"To Mum, Dad, Betta, Rich and Morgan"

Acknowledgements

I would like to thank my supervisor Eike Ritter for reading boring reports, proofs and bureaucratic documents and still being awake enough to give me advice and spot all my lost in translation mistakes. My greatest gratitude goes to Myrto Arapinis for being a superb example on how to do research, an invaluable motivator and the first supporter of my work. Myrto always had helpful criticism and useful suggestions and ideas all mixed up with a good dose of optimism, thank you! A special thank you goes to Mark Ryan for closely following my progresses and supporting my work with his great ability in quickly grasping concepts and finding weaknesses and strengths of an article. I would also like to thank Jon Rowe for his advice while being a member of my thesis group and Mark Lee for accepting to take his place, for his useful comments on my work and for being great fun. Thanks to the security group for being an invaluable source of useful comments and lively discussions and for being a great audience for interactive talks. This thesis would not have been written without the support of my friends and family. In particular, I would like to mention my parents for their unconditional support, my sister Elisabetta because life would not be the same if I could not share it with her. Rich for being very patient with my moods. Morgan for his weird sense of humour and contagious laugh, and Peter Oliveto for always being there for me. I would like to conclude mentioning all the fantastic academic, administrative and technical staff and the Ph.D. students of the school of computer science above all the ones who like to procrastinate, em, discuss important matters in the common room and have a pint or two in staff house.

This research was primarily funded by the Engineering and Physical Sciences Research Council (EPSRC), as part of the VPS: Verifying Interoperability Requirements in Pervasive Systems (EP/F033540/1).

Contents

Abstract iii

Acknowledgements vii

List of Figures xv

List of Tables xix

Mobile Telephony Acronyms xxi

1 Introduction 1
- 1.1 Motivation . . . 3
- 1.2 Privacy . . . 4
- 1.3 Security Protocols and Formal Methods . . . 6
- 1.4 Contributions . . . 13

2 Background: Applied Pi-Calculus 17
- 2.1 Applied pi-calculus . . . 19
  - 2.1.1 Syntax . . . 19
  - 2.1.2 Operational Semantics . . . 23
- 2.2 Equivalence Relations . . . 27
- 2.3 Modelling Security Properties . . . 31
  - 2.3.1 Privacy Related Properties . . . 32

3 Background: Introduction to Mobile Telephony Systems 39
- 3.1 GSM/UMTS Architecture . . . 40
- 3.2 GSM/UMTS Security Features . . . 42
- 3.3 Tools for Experimental Analysis . . . 45
  - USRP . . . 45
  - GNU Radio . . . 45
  - Open BTS . . . 45

Open BSC . . . 46
Osmocom-BB . . . 46
Femtocell . . . 46
HackRF . . . 46
BladeRF . . . 46
3.4 Previous Work on 2G/3G Security and Privacy . . . 49
3.5 Our Work on 2G/3G Privacy Analysis . . . 52

4 Analysis of Mobile Systems' Identity Management 55
4.1 Identification Procedure . . . 57
4.1.1 Identification Procedure Attack . . . 58
4.2 Paging Procedure . . . 58
4.3 TMSI Reallocation Procedure . . . 59
4.4 Subscriber Privacy Analysis . . . 61
4.4.1 IMSI Paging Attack . . . 62
4.4.2 Experimental Analysis . . . 62
4.4.2.1 Experimental Settings and Scenarios . . . 63
4.4.2.2 Findings/Results . . . 66
4.5 Discussion . . . 70

5 Analysis of Mobile Systems' Protocols 71
5.1 Pseudonymity Issues in Mobile Telephony Systems . . . 73
5.1.1 TMSI Reallocation Replay Attack . . . 73
5.2 User linkability issues in 3G systems . . . 74
5.2.1 3G Authentication and Key Agreement Protocol . . . 74
5.2.2 3G AKA Protocol Linkability Attack . . . 77
5.3 Implementation of some 3G Protocols Attacks . . . 79
5.3.1 Femtocell architecture . . . 79
5.3.2 Attack Procedure . . . 80
IMSI-Paging Procedure Attack . . . 81
AKA Protocol attack . . . 82

6 Privacy Friendly Fixes 85
6.1 Public Key Infrastructure . . . 86
Protecting the IMSI Paging Procedure . . . 87
Fixing the AKA Protocol . . . 88
Protecting the Identification Procedure . . . 90
Fixing the TMSI reallocation procedure . . . 90
6.2 Discussion of the Proposed Fixes . . . 91

7 Formal Verification of the Fixed Protocols 95
7.1 `ProVerif` Encoding . . . 96

7.1.1 Strong Unlinkability . . . 97
7.1.2 Strong Anonymity . . . 99
7.2 Automatic Verification Results and Remarks . . . 100
7.3 Unlinkability of the TMSI Reallocation Procedure . . . 102
7.3.1 Model of the TMSI Reallocation . . . 103
7.3.2 Proof of Unlinkability of the Fixed TMSI Reallocation . . . 105
7.3.3 Unlinkability Proof Sketch . . . 107
7.4 Remarks . . . 116

8 Automatic Verification of Equivalences for Stateful Processes: A `StatVerif` Extension 117
8.1 Related Work . . . 119
8.2 Our Contribution . . . 120
8.3 Background: `StatVerif` Process Language . . . 120
8.3.1 Syntax and Informal Semantics . . . 120
8.3.2 Operational Semantics . . . 123
8.3.3 Observational Equivalence . . . 126
8.4 StatVerif Extension to Observational Equivalence . . . 127
8.5 Clause Generation . . . 133
8.5.1 Clauses for the Protocol . . . 135
8.5.2 Clauses for the Attacker . . . 137
8.5.3 Clauses for the Mutability of Public States . . . 141
8.5.4 Soundness . . . 142
8.6 Implementation and Examples . . . 148
8.7 Discussion and Future Work . . . 161

9 Conclusions 163

A ProVerif Code 167

B Proof of the Unlinkability of the TMSI Reallocation Procedure 175
B.1 Definitions and Notation . . . 176
B.2 Proof of Lemma 7.1 . . . 179
B.3 Proof of Lemma 7.2 . . . 195

C StatVerif Extension Related Proofs 203
C.1 Proof of Theorem 8.5 . . . 203
C.2 Proof of Corollary 8.6 . . . 204
C.3 Preliminary Lemmas . . . 208
C.4 Proof of Lemma 8.8 . . . 209
C.5 Proof of Lemma 8.9 . . . 214

C.6 Results Preliminary to the Proof of Theorem 8.11 218
C.7 Proof of the Properties of the Type System 231
C.7.1 Substitution Lemma 234
C.7.2 Type Propagation Lemma 241
C.7.3 Typability of the Adversary (C[]) 245
C.7.4 Typability of the Protocol (A_0) 261
C.7.5 Typability of $C[A_0]$ 278
C.7.6 Subject Reduction 278

Bibliography **287**

List of Figures

3.1 2G/3G Architecture 41

4.1 2G/3G Identification Procedure. The network sends an identity request to the MS on a dedicated channel. The IMSI_Req parameter specifies that the requested identity is the long term identity, IMSI. The identity $IMSI$ of the mobile phone is sent in clear on the radio path. 57

4.2 3G IMSI Paging Procedure: the paging request message is broadcast on a common radio channel and contains the identity of the paged mobile station (in this case the $IMSI$) in cleartext. The paging response message contains the mobile phone's temporary identity $TMSI$ and is sent in clear on a dedicated channel established between the MS and the network (i.e. messages exchanged on this channel are meant to be originated by and or addressed to this particular MS). 59

4.3 2G/3G TMSI Reallocation Procedure: the TMSI reallocation procedure is always initiated by the network. The new TMSI, $TMSI_1$, along with the current Location Area Identifier, $newLAI$, is sent in an encrypted message to the mobile station in order to avoid users' linkability. The TMSI reallocation procedure is always executed on a dedicated channel 60

4.4 Experimental Tools 64

4.5 Osmocom-BB architecture 64

4.6 Trace of a UK Vodafone SIM card obtaining a new TMSI (0xb42c2fdd) on 22/03/12. The same TMSI is still in use on 25/03/12 after 3 days from its allocation. 65

4.7 Trace of a UK Orange SIM card. The TMSI used at location 234/33/1381 (packet no. 668) is accepted at location 234/33/29 (packet no.678), while the 3GPP standard mandates a TMSI reallocation at each change of location. 68

4.8 Trace of a UK Lebara SIM card attached to the Vodafone network while travelling on a train. The TMSI reallocation procedure is executed by reusing a previously established key. The MS first performs a location update (packet no. 4063), then the authentication procedure to establish a ciphering key (packets 4065, 4068), followed by the TMSI reallocation procedure (packets 4079, 4081). The following three TMSI reallocations (packets 9691, 9693, 71695, 71697, 92653, 92655) are executed without first performing the authentication procedure and hence reusing the previously established ciphering key. 69

5.1 TMSI Reallocation Procedure Attack: the attacker captures a legitimate TMSI reallocation command message and replays it. If the MS reply to the replayed message the attacker knows that it is indeed the victim MS. 72
5.2 3G Authentication and Key Agreement (AKA): the authentication procedure is always initiated by the network and is executed on a dedicated channel. 75
5.3 AKA Protocol Linkability Attack 78
5.4 Experimental Attack Setup 80
5.5 Linkability-Attack: Victim Found. 82
5.6 Linkability-Attack: Victim not Found. 83

6.1 IMSI Paging Procedure Fix. The IMSI paging request which is sent on a common channel is encrypted with the unlinkability key UK so to ide the long term identity $IMSI$. Each MS decrypts the IMSI paging request and check if the $IMSI$ contained in it is its own. If so the MS checks the freshness of the freshness of the request ($SQN_{MS} < SQN_N$), requests a dedicated channel and sends the response RES to the network. Otherwise the MS discards the message and aborts the procedure. 87
6.2 The fixed AKA protocol. The error messages are encrypted using the network public key. 88
6.3 Identification Procedure Fix. The identity response is encrypted with the public key of the network. The r denotes randomised encryption. 90
6.4 TMSI Reallocation Procedure Fix: this fix uses the SQN to ensure the freshness of the reallocation command. 91

8.1 StatVerif syntax 122
8.2 StatVerif semantics 125
8.3 Red Fun1, Red Fun2 and Red Comm reduction rules for `StatVerif` for observational equivalence 128

8.4 Translation of the protocol: null, replication, parallel, restriction, input, output and let 138
8.5 Translation of the protocol: lock, unlock, read and assign 139
8.6 Translation of the instrumented protocol 143
8.7 Type system 146

List of Tables

1.1 Automatic Verification Tools Comparison 12

7.1 `ProVerif` results of the on Fixed Procedures 101
7.2 `ProVerif` results on current 3GPP Procedures 101
7.3 Results of the Automatic Verification of the Fixed Procedures 102

Mobile Telephony Acronyms

GSM	**G**lobal **S**ystem for **M**obile Communication
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
MS	**M**obile **S**tation
BS	**B**ase **S**tation
IMSI	**I**nternational **M**obile **S**ubscriber **I**dentity
TMSI	**T**emporary **M**obile **S**ubscriber **I**dentity
SIM	**S**ubscriber **I**dentity **M**odule
SN	**S**erving **N**etwork
HN	**H**ome **N**etwork
MNO	**M**obile **N**etwork **O**perator

1

Introduction

Pervasive or ubiquitous computing are terms used to describe a reality where everyone is immersed in a system composed of various sensing and computing devices able to communicate with each other to enhance the user's experience seamlessly throughout his daily life. They aim to offer services on locus without the need to go to an office or service provider site and in a highly customizable and environment aware fashion. The reality of pervasive computing is the reality we live in and we have just started exploring, developing and enjoying its capabilities. Home care systems, vehicular networks, smart grid and more in general smart home systems, toll payment and wireless payment systems, personalised marketing and personalised tourist information, urban sensing, are just some examples of the pervasive systems continuously emerging thanks to a variety of environment aware data collected and elaborated by ubiquitous interconnected devices and sensors such as RFID tags, cameras, light sensors, smart cards, credit cards, loyalty cards, mobile phones, accelerometers, GPS enabled devices etc. These devices are generally small and easy to use and carry around; typically they do not prompt the user when collecting and/or communicating personal data. Hence, the user is often not aware of potential security and privacy threats deriving from the use of these technologies, and has little or no control over the generation, sharing

and use of the data itself. Therefore, along with the benefits, the ubiquitous presence of smart, context-aware sensing and computing devices brings with it various concerns, mainly related to the security and privacy properties of such systems.

Formal methods have proved to be effective in highlighting weaknesses of protocols, enabling designers and programmers to patch them and strengthen their security guarantees, as well as giving strong assurance on protocol properties by proving the absence of attacks undermining the stated security properties. For example, conference management [ABR12], electronic voting [CW12a, GRBR13, CS12], single-sign-on [BBDM14, ACC+08a], cloud storage [BBDM13], TPM [DKRS11, BXR13], RFID [CS10], and mobile telephony protocols [AMR+12] have been scrutinized using manual and automatic verification techniques. However, manual proof methods are lengthy and error prone, while automatic verification tools have to compromise in order to achieve decidability (at least for some subclass of processes) by bounding the number of agents and sessions and/or restricting the considered class of cryptographic functions. the

Although, many of the security properties of pervasive systems can be modelled and analysed in terms of the classical concepts of confidentiality, authentication and integrity, new definitions of properties are often required to capture the desired characteristics of specific pervasive systems, in particular when addressing privacy and its different aspects and shades. Some of the most recurring privacy related properties can be described in terms of location privacy, anonymity and unlinkability. However, some protocols require the definition of more specific properties, such as voter's privacy and forward privacy. Furthermore, the more general definititheons can have different shades and can be weaker or stronger to suit a protocol's requirements and the attacker capabilities.

Modelling and verifying these properties is a difficult task since the sources of

possible attacks are often hidden in implementation details or even in the protocol logic rather than in the cryptographic guarantees of the employed encryption algorithm. Moreover, the modelling of the privacy-related properties often relies on non standard definitions of equivalence and as a result some of the properties are not supported by the currently available automatic verification tools. Hence, modelling and verifying these novel properties is one of the challenges pushing for further development of the currently available automatic tools and, of course, of the theory supporting them.

1.1 Motivation

Every day we carry in our pockets a collection of devices giving us access to some pervasive system. Electronic passports and identity documents, electronic health cards, electronic keys, loyalty cards, tablets and smart phones not to mention more specialized gadgets such as sports bracelets and smart watches. All these devices store and/or collect personal and/or contextual information and communicate with a provider infrastructure to deliver a variety of services. In this respect, we can consider our pockets as part of a big pervasive system enhancing our day to day experience. At the same time, we can consider our pockets as the most powerful surveillance tool ever witnessed. In fact, the amount, quality and frequency of the collected personal data is of a magnitude which would not be possible without the support of modern electronic devices. Furthermore, all this personal information is generally transmitted over-the-air and then stored and elaborated in the cloud and hence far from the control of the user who is often not even aware of how and when the information is collected nor the kind of data that can be computed from it.

The personal nature of the data collected by ubiquitous devices makes security and overall privacy a central topic for pervasive systems. Moreover, the level of privacy

offered by a pervasive system could determine the acceptance or otherwise by the users of the system itself. For example in [Shi09] the author argues that users could be not willing to join participatory sensing projects without an adequate assurance regarding the provided level of privacy. The reaction of users to possible privacy threats can be as strong as mounting a boycott campaign, as witnessed with the introduction of RFID tags by Benetton [ben].

1.2 Privacy

It is difficult to be a privacy advocate without being addressed as a privacy freak or someone who has something to hide and it is often needed to justify oneself. I personally agree with D'Introna's viewpoint, in [Int97]. He identifies privacy as a very important characteristic for the definition of social relationships and more crucially a fundamental property to enable autonomy and individuality since knowing to be observed influences a person's behaviour. Indeed, we adopt different kind of behaviours depending on who is the observer. What we share and with who defines, for example, on one hand private and more intimate relationships and on the other work-related ones. Hence, privacy seems to be fundamental for the definition of oneself. However, privacy is a relatively novel concept that appears for the first time in a 1890 analysis in the Harward Law Review in relation to a case of privacy invasion by the press [WB90] and there is no universally accepted definition of privacy. In the context of pervasive systems privacy is a major concern [Cas05, Rya11, Shi09, KFD10] because, as previously mentioned, the quality, quantity and accuracy of collected personal data is of great magnitude. Furthermore, the user is often unaware of generating the data and has little or no control over its generation, exchange, transfer, and use. So, as Cas argues in [Cas05] pervasive systems could be considered as surveillance systems threatening individual's privacy and privacy should be one of the main security properties of pervasive systems. In particular lots of concern is about how the collected data is handled, if it is released or even

sold to third parties, where it is stored and for how long, and if it is aggregated and processed to deduce correlated information.

As in the real world, in the electronic world as well, privacy is a difficult property to define and we rather speak of privacy-related properties instead. These can be general ones or application specific ones. Moreover, privacy is a complex property to enforce and verify since it is multi-level and multi-protocol, i.e. it crosses more than one layer, if not all layer of the protocol stack, and typically has to be enforced by more than one protocol in order to be satisfied [AO05]. While some aspects of privacy are more suited to be enforced by policies as for example the sale of private information to third parties, others should be part of the verifiable properties embedded in the design of a system. However, privacy seems to be often in conflict with other system requirements as for example accountability, or in contrast with the nature of the system itself as in the case of location based services. For these reasons user's privacy is often overlooked or enforced by complex mechanisms which are difficult to verify by hand.

The most obvious privacy-related property is data confidentiality, however it is generally not sufficient to ensure confidentiality in order to achieve privacy, for example a user identifier could be public and his identity could be disclosed by correlating his activity [NS09, NS06, BZj06]. Moreover multiple accesses to a system from different locations by the same identifier could directly or indirectly reveal a user's position or his movement patterns. Hence, anonymity and unlinkability of a user's activity and access to a system are important aspects of a user's privacy. Location privacy is one of the privacy aspects which has been the subject of many studies [BS03, Kru09, GHT05], overall from the computational point of view [Kru09] and several techniques have been proposed to protect a user's location as for example obfuscation, pseudonymity and mix-zones [BS03]. Examples of application specific privacy requirements are the ones arising in the field of electronic voting such as voter's privacy, which concerns the impossibility of linking a voter to his ballot from both third parties and authorities, coercion resistance,

i.e. the impossibility for a voter to prove how they voted and revocable anonymity which is the possibility for an authorised entity to reveal the identity of a single voter. Another interesting privacy-related property which can be for example desirable for private information stored on portable devices is forward privacy which requires that the secrecy of certain information holds even after the device has been corrupted. For example, it may require that session keys are not retrievable even in the event that the master keys were disclosed after the device was corrupted. Formal definitions of anonymity and unlinkability are given in [ACRR10], definitions of untraceability and forward privacy are given in [BCdH10] and a definition of voter's privacy is given in [KR05]. These definitions are all equivalence-based and often pose challenges for the currently available automatic verification tools because of the definition of the property itself, because of the structure of the protocol or because of the algebraic properties of the involved cryptographic primitives.

In this thesis we are concerned about the latter set of properties and about developing the theory to automatically verify the compliance of a system to the relevant privacy related properties. In particular we focus on anonymity and user unlinkability as defined in [ACRR10].

1.3 Security Protocols and Formal Methods

Security protocols aim to protect sensitive data in particular when communicating over an unprotected connection, where there is little or no control over the flow of information and the attacker can intercept, manipulate, replay, inject, replace, substitute and compare messages as in the Dolev-Yao attacker model [DY81]. This makes the design of security protocols a notoriously difficult task since it does not only involve the protection of data through some sort of smart cryptography algorithm but as well the ability of foreseeing how the flow of messages could be

used to retrieve, deduce and, get access to sensitive information. Details hidden in the protocol logic often allow to break a protocol security without even breaking the underlying cryptography. A famous example is the attack to the Needham-Schroeder protocol [NS78, Low96a] which allows an attacker to authenticate instead of a legitimate user by performing a man-in-the-middle attack. A more recent example is the traceability attack on the French e-passport which allows one to trace an e-passport holder by performing a replay attack [CS10]. Hence, to declare a protocol secure one should check it against any possible adversary and any possible interaction. Moreover, one would need to state what being secure means for the protocol, i.e should specify the security properties the protocol aims to achieve, and this would depend on the purpose of the protocol and on the application requirements. Hence, to evaluate the security of a protocol it is very important to rigorously define both the protocol itself and the desired security properties.

Formal methods help establishing the security of protocols in three ways:

- rigorously modelling security protocols and the attacker model
- rigorously defining security properties
- evaluating the protocols against the desired properties

There are two different formal approaches to the problem of protocol security. The computational approach is closer to the actual protocol implementation, represents messages as bitstrings, cryptographic functions as polynomial time algorithms and the adversary as any probabilistic polynomial time algorithm and can give strong guarantees on the security of the cryptographic algorithms. However, proofs in this setting can be very long, difficult, error prone and not very accessible to careful scrutiny. Moreover they are not very well suited for automation.

In this thesis we will use symbolic methods instead. Symbolic methods abstract from the details of the cryptographic algorithms. Messages are represented by

terms and cryptographic primitives are represented by function symbols which can be applied to terms. The properties of cryptographic primitives are abstracted by algebraic properties. In general, cryptographic primitives are assumed to be perfect. Hence, the adversary can have any interaction with the protocol, injecting, modifying, replaying, intercepting messages but cannot break cryptography. One way of expressing security properties is in terms of reachability-based property, i.e in terms of the possibility of executing the protocol in the adversarial environment and reaching a state with specific characteristic, as for example a state where a certain term is emitted on a public channel. Another way is in terms of equivalence-based property, i.e. in terms of the indistinguishability between two protocols. Usually, one is called the ideal protocol since it satisfies the required property by construction, while the other is the real protocol. If the adversary cannot distinguish the real protocol from the ideal one, we say that the real protocol satisfies the required property.

As in the case of computational methods, in the symbolic world the security of protocols can be established by means of long, tedious and error prone manual proofs or by manually exhibiting a counter-example i.e. an attack. However, lots of work focus on automating the proofs and/or search for counter-examples. In this area there are still plenty of challenges and room for improvements and researchers continuously aim to expand automatic verification to larger classes of protocols and equational theories for both reachability and equivalence-based properties.

Some of the most famous formalisms in the field of symbolic methods are process calculi, constraint systems and strand spaces. In this thesis we will use the applied pi-calculus [AF01] which is a calculus to model cryptographic protocols aiming to overcome the limitations of the spi-calculus [AG97] by offering flexibility in the range of cryptographic primitives represented by function symbols and equational theories and making the adversary knowledge explicit thanks to the frame construct. Many results obtained using different formalisms such as spi-calculus and constraint systems can be transposed into the applied pi-calculus.

We report some of the most relevant. The general problem of security is shown to be undecidable even for a bounded number of sessions for both reachability-based properties [DLMS99] and for equivalence-based properties [H02]. Adding some restriction to the calculus and or to the equational theory does usually yield decidability results.

One of the most common restriction is to consider only a finite number of sessions. Automatic verifiers for bounded processes can be very useful to find protocol flaws, however when they do not find any attack, no guarantees of the absence of attack can be derived for a number of sessions greater than the one used to run the verifier on the protocol. To obtain general security results an unbounded number of sessions should be considered. Another common restriction to the calculus consists in forbidding else branching, and the equational theory is often required to be convergent.

For what concerns reachability based properties, NP-complete procedures which consist into transforming constraint systems into solved form are given in [CLCZ10, RT03]. Deducibility and static equivalence are shown to be decidable for subterm convergent theories in [AC06] but the algorithm has not been implemented and it is not parametric with respect to different equational theories. However, some procedures for deducibility and static equivalence of bounded processes and convergent equational theories have been implemented [BCD09, CDK09]. The YAPA tool [BCD09] additionally supports blind signatures and homomorphic encryption and it is sound and complete when it does not fail. Kiss [CDK09] is correct and terminates on subterm convergent theories.

The AVISPA tool [ABB+05, Pro] is a common platform for a set of four security protocol verifiers: On-the-fly Model-Checker (OFMC) [BMV03], Constraint-Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) [ACC14] are reachability properties verifiers for bounded protocols, OFMC is correct and complete and supports the specification of algebraic properties. While the Tree

Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) [BHKO04] supports unbounded processes, however when it finds an attack it does not report an attack trace so it is difficult to assess if the attack is a false attack due to the over-approximation. Moreover, TA4SP does not support user-defined equational theories.

The Scyther tool [Cre08] supports the verification of reachability properties for both bounded and unbounded number of sessions with no else branches and for a fixed set of primitives. When Scyther finds an attack it provides attack traces. Scyther is guaranteed to terminate for a maximum number of five sessions and it does not give false attacks.

A decision algorithm for trace equivalence of bounded processes with no else branching and a pre-defined signature consisting of pairing, symmetric and asymmetric encryption is given in [CCLD10] along with a prototype implementation for checking deducibility and static equivalence but not for deciding trace equivalence for which it is required to implement a further procedure that generates a pair of constraints for each possible interleaving. The AKiss tool implements two procedures for bounded processes with no else branches and for convergent rewrite systems with the finite variant property, one under-approximate trace equivalence and can be used to prove protocols correct, the other over-approximate trace equivalence and can be used to discard incorrect protocols. The tool can be used to check observational equivalence of determinate processes since in this case observational equivalence coincides with the under approximation of trace equivalence.

The decidability of observational equivalence of *simple* processes with no replication and with no else branches for subterm convergent equational theories is shown in [CD09]. The decision procedure relies on the decidability of the equivalence of constraint systems given in [Bau05]. This procedure has not been implemented so far. To the best of my knowledge the only tool able to automatically verify

both reachability-based [Bla09] and equivalence-based [BAF05] properties for unbounded processes with else branches and convergent theories is the `ProVerif` tool [Bla01]. Blanchet, Abadi and Fournet [BAF05] introduce the concept of bi-process, which is a pair of processes that differ only in the choice of some term. `ProVerif` can prove observational equivalence of bi-processes. However, the procedure is sound but not complete, meaning that the tool can prove that a property holds on the given protocol model but when it outputs an attack trace this could be a false attack. Moreover the tool may not terminate. `ProVerif` has many limitations [PR11], for example, it is not possible to fully model the algebraic properties of XOR or Diffie-Hellman exponentiation and it can only check observational equivalence of processes with same structure but differing in some terms. Some tools based on `ProVerif` have been developed aiming to overcome its limitations. Küsters and Truderung develop methods to transform theories using XOR and Diffi-Hellman exponentiation into theories accepted by `ProVerif` and give prototype implementations [KT11, KT09]. Delaune et al. [DRS08] introduce strong phases to model processes that synchronize at a certain point of their evolution after which no other process can execute instructions of the part of the protocol preceding the synchronization. Furthermore, they introduce a data swap mechanism so to expand the class of equivalent processes verifiable by `ProVerif` and hence better approximating observational equivalence. These extensions are implemented in the tool *proswapper* [KSR08]. A further step towards a better approximation of observational equivalence is taken by Cheval and Blanchet in [CB13]. They extend `ProVerif` to support rewrite rules with inequalities as side conditions, in particular they can have an alternative definition of *if then else* and avoid false attacks due to differences in the branching of the examined pairs of protocols. This allows for example to verify that `if` $a = a$ `then` P `else` P is observationally equivalent to `if` $a = b$ `then` P `else` P, while `ProVerif` would exhibit a false attack even on this simple process. Another `ProVerif` extension,

	KISS	AKISS	YAPA	AVISPA (TA4SL)	AVISPA (SATMC, OFMC, CL-AtSe)	ProVerif and extensions	Scyther
User defined theories	√	√	√	x	x (OFMC √)	√	x
Unboundedness	x	x	x	√	x	√	√
Branching	x	x	x	√	√	√	x
Attack traces	x	x	x	x	√	√	√
No false attacks	√	√	√	x	x(OFMC √)	x	√
Termination	√	x	x	x		x	√
Reachability	√	x	√	√	√	√	√
Equivalence	x	√	x	x	x	√	x

TABLE 1.1: Automatic Verification Tools Comparison

which is closely related to the work presented in this thesis, is the one developed by Arapinis et. al. in [ARR11]. They extend the `ProVerif` calculus so to explicitly model protocols with persistent state and implement a tool that translates processes with state into Horn clauses so to be able to automatically verify reachability-based properties by feeding the translation result to `ProVerif`. An extension of the applied pi-calculus and related definitions and equivalences for stateful processes is given in [ALRR14], interestingly they give a proof of the coincidence of labelled bisimilarity and observational equivalence. The WebSpi library [BBM12, BBDM13, BBDM14] allows the automatic and sistematic analysis of application level protocols using the `ProVerif` tool. WebSpi was used to analyse single-sign-on protocols and web storage services and proved effective in finding known and novel practical attacks.

In table 1.1, we summarize some of the characteristic of the mentioned tools. Further details on the comparison between automatic verification tools can be found in [DSHJ10, CLN09, LTV10, PR11]

The problem of privacy formalization is tackled by Arapinis et al.. They define weak and strong untraceablity properties in applied pi calculus and illustrate them using RFID tag related examples [ACRR09]. They later refine the given definitions and in [ACRR10] they define strong and weak unlinkability and strong and weak anonymity properties in applied pi calculus and apply them to a case study. They show that unlinkability does not imply anonymity and give a concrete example

of this showing that the protocol used by the French RFID e-passport preserves anonymity, but does not preserve unlinkability.

1.4 Contributions

Privacy in all its different forms is usually expressed from a symbolic methods point of view in terms of some sort of equivalence. However, automatic support for the verification of equivalence-based properties is not as developed as the one for the verification of reachability properties. So privacy properties do often challenge the currently available tools and theories.

In this work we contribute to the analysis of privacy-related properties by taking steps towards the automatic verification of equivalence based properties and by showing how our techniques work on a real world case study, namely mobile telephony systems. We show how some definitions of unlinkability and anonymity as given in [ACRR10] can be adapted to be used for automatic verification using the `ProVerif` tool. Using this technique we analyse a set of real world protocols used in mobile telephony. We point out some privacy flaws in the mobile telephony standard specifications and we confirm them on a real network thanks to a prototype implementation. Moreover, we propose and automatically verify some fixes of the flawed protocols. We will show that the technique we use to verify anonymity and unlinkability suits protocols with no initialization phase but fails when used in particular to automatically verify protocols with an initialization phase and states. We manually analyse the unlinkability property of a real protocol with states used in mobile telephony to assign new pseudonyms to mobile phones (called TMSI reallocation procedure). This analysis is complemented by an experimental study of the protocol. Thanks to both experimental and formal analysis we are able to expose some weaknesses of the use of pseudonyms in mobile telephony systems. As part of this study, we present a formal proof of

the unlinkability of a straightforward fix of the pseudonym's assignment protocol. This manual proof is one of the few manual proofs of observational equivalence in the literature [CW12b, KR05]. Finally, in an attempt to further the class of automatically verifiable protocols using `ProVerif` we propose an extension of it to support the automatic verification of equivalence based properties of protocols with states. This extension is based on the theory developed for the `StatVerif` tool. We will show some examples and point out the limitations of the current version of the tool.

Structure The first two Chapters focus on the background knowledge the work presented in this thesis is based on. In Chapter 2 we introduce the applied pi-calculus, a well established calculus for the analysis of security protocols. Chapter 3 presents mobile telephony systems and some of the tools available for the analysis of the communication over the radio link. In Chapter 4 we present flaws in the identity management of mobile telephony systems. More subtle attacks to the privacy of mobile telephony users are presented in Chapter 5. In Chapter 6 we present some privacy friendly fixes of the flawed protocols and subsequently we show how to automatically verify the fixed procedure in Chapter 7. Chapter 8 presents the work we undertook to extend `ProVerif` to the automatic verification of observational equivalence based properties of stateful protocols. Finally, in Chapter 9 we conclude with some final remarks and ideas for further work.

Publications The IMSI paging attack (Section 4.4.1), the AKA protocol attack (Section 5.2.2), their prototype implementation (Section 5.3), their fixes (Section 6) and formal verification (Section 7) were first published at the CCS12 conference:

- Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New privacy issues in mobile

telephony: Fix and verification. In *Conference on Computer and Communications Security*, CCS, pages 205–216. ACM, 2012

The pseudonyms management analysis (Section 4.4.2), the TMSI reallocation attack (Section 5.1.1) and the formal proof of unlinkability (Section 7.3) were first published at the NDSS14 conference

- Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, and Mark Ryan. Privacy through pseudonymity in mobile telephony systems. In *Network and Distributed System Security Symposium, NDSS*, 2014

The prototype implementation of the IMSI paging and AKA protocol attack was built by Nico Golde, Kevin Redon and Ravi Borgaonkar at the secT labs, Technische Universität, Berlin.

I personally contributed to the work presented in this Thesis by extracting the protocols from the mobile telephony standard documents, modelling them for the verification with the `ProVerif`tool and conducting the manual proofs of unlinkability of the fixed TMSI reallocation B and of soundness of our extension of `ProVerif`to the automatic verification of equivalence of stateful protocols C.

2

Background: Applied Pi-Calculus

In this Chapter we introduce the applied pi-calculus, a well established calculus to reason about security protocols. Furthermore, we give some definitions of equivalence and show how these can be used to define security properties and in particular how they are used in [ACRR10] to define privacy related properties such as unlinkability and anonymity.

In contrast with the computational methods which concern the verification of the cryptographic functions involved in security protocols, formal methods assume perfect cryptography and model security protocols as sequences of interactions between the agents involved in the protocol execution. Different formal methods approaches use different abstractions to represent security protocols.

Logic based approaches represent protocols in terms of logic formulae and inference rules. Process calculus-based approaches model protocol participants as processes running in parallel and interacting through message exchange. Other symbolic approaches represent protocols in terms of the sequences of messages describing the protocol traces.

Proving the security properties of the modelled protocols using manual proof techniques can be a long and error prone process. However, automatic verification tools have been developed to automate the proof process. López and Monroy [PM08] identify three main approaches to automatic verification: belief logic, state exploration (also known as model checking) and theorem proving.

Belief logic, introduced by Burrows et al. in [BAN90], is based on a set of assertions such as "A believes B" and "A sees B" and deduction rules. These abstractions allow capturing and inferring what agents can derive from the messages received while executing a protocol.

Model checking approaches [ABB+05, BMV03, ACC14] deal with the verification of protocols represented by the set of possible execution traces. Model checkers explore all the possible execution paths of the protocol model, while checking at each state if the required properties hold. If a property is not satisfied at a given state, the model checker can produce a counter-example by following the trace which led to that state. Model checking techniques work on bounded protocols and are well suited to find weaknesses. However they suffer from the state explosion problem since all possible protocol traces have to be checked against all possible interactions with the adversary.

Theorem proving-based approaches are logic based approaches. They aim at producing a proof that a property holds on the given representation of a protocol. Blanchet's `ProVerif` tool [Blab] belongs to this category. It uses prolog rules to represent protocol's messages and the attacker deduction capabilities. Abstractions are made to make the verification terminate on more processes. As a result of these abstractions `ProVerif` can produce false attacks, though it does not produce false proofs, i.e. if there is an attack `ProVerif` will find it, or it may not terminate. `ProVerif` accepts as input descriptions of protocols given in applied pi-calculus which is the formal language we use to analyse security protocols. As

we pointed out in Chapter 1, many automatic verification tools have been developed, however `ProVerif` is the only tool supporting the automatic verification of both reachability and equivalence properties for unbounded protocols with else branches and user defined equational theories.

In this thesis we will use the applied pi-calculus to model and reason about security protocols and their security properties. We adopt the `ProVerif` tool to automatically verify privacy properties defined in terms of equivalence relations and we further develop the theory behind `ProVerif` and `StatVerif` in order to support the automatic verification of observational equivalence based properties of stateful protocols.

2.1 Applied pi-calculus

The applied pi-calculus is a formal language for modelling concurrent processes introduced by Abadi and Fournet [AF01] to ease the modelling of cryptographic protocols. It extends the pi-calculus with a set of function names to model cryptographic primitives; a set of variables and an equational theory to model an attacker's capabilities; and active substitutions to model an attacker's knowledge.

In this Chapter we will introduce the applied pi-calculus and the formal definitions of privacy properties we are interested in verifying, namely unlinkability and anonymity.

2.1.1 Syntax

The messages exchanged during a protocol execution are modelled by *terms*. The *terms* of the applied pi-calculus are built over an infinite set of names, an infinite set of variables and a *signature* Σ, which is a finite set of function symbols each

with an arity. A function symbol with arity 0 is a constant symbol. Terms are defined by the following grammar:

L, M, N, T	$::=$	Terms
	$a, b, c, \ldots$	Name
	$x, y, z, \ldots$	Variable
	$f(M_1, \ldots, M_l)$	Function application

Where $f \in \Sigma$ and l matches the arity of f. A term is *ground* if it does not contain *free* variables. We use metavariables u, v, w to range over both names and variables.

We rely on a sort system for terms. Terms can be of a base sort or a channel sort. Base sorts include `Integer`, `Nonce` or `Key` or simply a universal base sort such as `Data`. If τ is a sort then `Channel`(τ) is a sort for channels carrying terms of sort τ. Variables can have any sort. For simplicity, function symbols take arguments and produce results of base sort only. We assume that terms are well-sorted and that substitutions preserve sorts.

The signature Σ is equipped with an equational theory E, which is a set of equivalence relations on terms that is closed under substitutions of terms for variables, application of function symbols and one-to-one renaming. We consider two terms to be equal $M = N$, if the equality $M = N$ holds with respect to the equational theory E ($E \vdash M = N$).

Example 2.1. *As an example, let's consider the following signature* Σ*:*
$\Sigma = \{\mathit{senc}/2, \mathit{sdec}/2, \mathit{aenc}/2, \mathit{adec}/2, \mathit{verify}/2, \mathit{sign}/2, \mathit{pub}/1, \mathit{fst}/1,$
$\mathit{snd}/1, \mathit{pair}/2, \mathit{true}/0\}$

And equational theory E*:*
$\mathit{sdec}(y, \mathit{senc}(y, x)) = x$
$\mathit{adec}(y, \mathit{aenc}(\mathit{pub}(y), x)) = x$

$\mathit{verify}(\mathit{pub}(y), \mathit{sign}(y, x)) = \mathit{true}$

$\mathit{fst}(\mathit{pair}(x, y)) = x$

$\mathit{snd}(\mathit{pair}(x, y)) = y$

The first two equations model respectively the properties of symmetric and asymmetric encryption. The third models a signature verification function. The last two equations model first and second projections of a pair. Let M and N be two terms:
$M = \mathit{fst}(\mathit{pair}(a, b))$ *and* $N = \mathit{adec}(k, \mathit{aenc}(\mathit{pub}(k), a))$.
We have that $M = N$ modulo the equational theory E and write $E \vdash M = N$.

The grammar for *processes* of the applied pi-calculus is the following:

P, Q, R	$::=$	Plain processes
	0	Null process
	$P \mid Q$	Parallel composition
	$!P$	Replication
	$\nu\, n.P$	Name restriction ("new")
	$\texttt{if } M = N \texttt{ then } P \texttt{ else } Q$	Conditional
	$u(x).P$	Message input
	$\overline{u}\langle N \rangle .P$	Message output

The null process does nothing. The parallel composition of P and Q represents the parallel execution of P and Q. The replication of a process P acts like the parallel execution of infinite copies of P. The name restriction $\nu\, n.P$ creates a new name n whose scope is restricted to the process P and then runs P. The conditional checks the equality of two terms M and N and then behaves as P or Q accordingly. Note that we check for equality modulo the considered equational theory, rather than syntactic equality of terms. The message input $u(x).P$ represents a process ready to input from the channel u. The actual message received will be substituted to x in P. The syntactic substitution of a term T for the variable x in the process

P is indicated by $P\{^{T}/_{x}\}$. The message output $\overline{u}\langle N\rangle.P$ describes a process ready to send a term N on the channel u and then to run P.

Plain processes are extended with *active substitutions*. The grammar for *extended processes* is the following:

A, B, C	$::=$	Extended processes
	P	Plain process
	$A \mid B$	Parallel composition
	$\nu\, n.A$	Name restriction
	$\nu\, x.A$	Variable restriction
	$\{^{M}/_{x}\}$	Active substitution

An active substitution $\{^{M}/_{x}\}$ offers the handle x to access the term M. If not under restriction the handle x has a global scope and can be thought of as being part of the environment or adversary knowledge. When restricted like in the process $\nu\, x.(\{^{M}/_{x}\} \mid P)$ it acts as a local declaration, like $\texttt{let}\ x = M\ \texttt{in}\ P$. We assume substitutions are cycle-free. We use σ and τ to range over substitutions, we write $T\sigma$ for the result of applying σ to the free variables of T.

Names and variables have scopes which are delimited by restriction and input. The set of free names $fn(A)$ of an extended process A is the set of all names a which occur in A and are not under restriction $\nu\, a$ in A. The set of free variables $fv(A)$ of an extended process A is the set of all variables x which occur in A and are not bound by an input $u(x)$ or under restriction $\nu\, x$ in A. The set of bound names $bn(A)$ of an extended process A is the set of all names a which occur in A and are under restriction $\nu\, a$ in A. The set of bound variables $bv(A)$ of an extended process A is the set of all variables x which occur in A and are bound by an input $u(x)$ or a restriction $\nu\, x$ in A. An extended process A is *closed* if all the variables occurring in A are either bound or defined by an active substitution.

A *frame* is an extended process built from 0 and active substitutions by parallel composition and restriction. We use ϕ and ψ to range over frames. The *domain* $dom(\phi)$ of a frame ϕ is the set of variables x which occur in ϕ and are not under restriction. The frame $\phi(A)$ of an extended process A is obtained by replacing every plain process in A with 0. The domain $dom(A)$ of an extended process A is the domain of $\phi(A)$. The frame of an extended process A can be seen as the static knowledge that A exposes to its environment at a given point of its execution. In fact, the frame of an extended process and hence the knowledge it exposes to the environment can grow during the process execution.

Example 2.2. *We can now model processes of the applied pi-calculus. As an example let us consider the following process:*

$$P = \nu\, k.c(x).\texttt{if}\ x = \mathit{hello}\ \texttt{then}\ \overline{c}\,\langle \texttt{pub}(k)\rangle\ \texttt{else}\ 0$$

Which creates a new key k, waits for a message on the public channel c. If the received message is `hello`*, the process sends its public key on the channel c.*
The frame of P is empty $\phi(P) = \emptyset$. P has two free names $fn(P) = \{c, hello\}$ and a bound one $bn(P) = \{k\}$. Moreover, P has a bound variable $bv(P) = \{x\}$ but no free ones. Hence, $dom(\phi(A)) = dom(A) = \emptyset$.

2.1.2 Operational Semantics

A *context* $C[\_]$ is a process or an extended process with a hole. An *evaluation context* is a context whose hole is not under a replication, a conditional, an input or an output. We say that a context closes A when $C[A]$ is closed, where $C[A]$ is the process obtained by filling C's hole with A.

Structural Equivalence. Structural equivalence $\equiv$ is the smallest equivalence relation on extended processes that is closed by α-conversion on both names and variables, by application of evaluation context, and such that:

PAR-0	A	$\equiv$	$A \mid 0$	
PAR-A	$A \mid (B \mid C)$	$\equiv$	$(A \mid B) \mid C$	
PAR-C	$A \mid B$	$\equiv$	$B \mid A$	
REPL	$!P$	$\equiv$	$P \mid !P$	
NEW-0	$\nu\, n.0$	$\equiv$	0	
NEW-C	$\nu\, u.\nu\, v.A$	$\equiv$	$\nu\, v.\nu\, u.A$	
NEW-PARA	$A \mid \nu\, u.B$	$\equiv$	$\nu\, u.(A \mid B)$	where $u \in fv(A) \cup fn(A)$
ALIAS	$\nu\, x.\{^M/_x\}$	$\equiv$	0	
SUBST	$\{^M/_x\} \mid A$	$\equiv$	$\{^M/_x\} \mid A\{^M/_x\}$	
REWRITE	$\{^M/_x\}$	$\equiv$	$\{^N/_x\}$	where $M =_E N$

Intuitively, structural equivalence is an equivalence relation relating processes which model the same thing but are syntactically different, as for example the parallel execution of two processes A and B which can be modelled as $A \mid B$ or $B \mid A$. The rules for parallel composition and restriction are self-explanatory. The ALIAS rule allows the introduction of an arbitrary active substitution, the SUBST rule describes how an active substitution is applied to a process it is in contact with. The REWRITE rule allows rewriting of terms which are equivalent with respect to the considered equational theory E.

Internal Reduction. Internal reduction is the smallest relation on extended processes closed by structural equivalence and application of evaluation contexts such that:

$$\begin{array}{ll} \textsc{Comm} & \overline{c}\langle M\rangle.P \mid c(x).Q \to P \mid Q\{^{M}/_{x}\} \\ \textsc{Then} & \texttt{if } M = M \texttt{ then } P \texttt{ else } Q \to P \\ \textsc{Else} & \texttt{if } M = N \texttt{ then } P \texttt{ else } Q \to Q \\ & \text{for ground terms } M, N \text{ where } M =_E N \end{array}$$

The rule Comm describes the synchronization of input and output actions on a channel c. The result of the comparison between terms in the conditional rules depends on the equational theory E. Intuitively, internal reductions capture the internal behaviour of processes, that is the actions a process can execute without interacting with the environment.

Labelled Reduction. The labelled reduction relation extends the internal reduction enabling interactions with the environment. The labelled reduction is a ternary relation $A \xrightarrow{\alpha} A'$ where the label α is of the form:

- $c(M)$ where c is a channel name and M is a term that may contain names and variables. This label corresponds to an input of M on the channel c.
- $\overline{c}\langle u\rangle$ where u is either a free channel name or a free variable of base sort. This label corresponds to the output of u on the channel c.
- $\nu\, u.\overline{c}\langle u\rangle$ where u is either a bound channel name or a bound variable of base sort. This label corresponds to the output of either a bound channel name or a bound variable u on the channel M.

The operational semantics is extended with the following labelled reductions:

IN $$c(x).P \xrightarrow{c(M)} P\{^M/_x\}$$

OUT-ATOM $$\overline{c}\langle u\rangle.P \xrightarrow{\overline{c}\langle u\rangle} P$$

OPEN-ATOM $$\frac{A \xrightarrow{\overline{c}\langle u\rangle} A' \quad u \neq c}{\nu\, u.A \xrightarrow{\nu\, u.\overline{c}\langle u\rangle} A'}$$

SCOPE $$\frac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu\, u.A \xrightarrow{\alpha} \nu\, u.A'}$$

PAR $$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \emptyset}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

STRUCT $$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

The rule IN allows a process to input a term M which is then substituted for the free variable x in the continuation P of the process. OUT-ATOM only allows the output of channel names and variables of base sort; the output of a term M can be obtained by reference, instantiating an active substitution $\nu\, x.\{^M/_x\}$ and outputting the handle x. The rule OPEN-ATOM allows the output of either a restricted name or a restricted variable of base sort and hence extends its scope to the environment.

We use the notation $\rightarrow^*$ or $\Rightarrow$ to indicate an arbitrary number of internal reductions. We write $\stackrel{\alpha}{\Rightarrow}$ to denote an arbitrary number of internal reductions followed by a labelled reduction and then an arbitrary number of internal reductions, i.e. $\stackrel{\alpha}{\Rightarrow}$, equals $\Rightarrow\stackrel{\alpha}{\rightarrow}\Rightarrow$

Example 2.3. *We can now see how processes evolve and interact with the environment. For example, let's consider the following processes:*
$P = \nu\, k.c(x).\texttt{if}\ x = hello\ \texttt{then}\ \overline{c}\,\langle \texttt{pub}(k)\rangle\ \texttt{else}\ 0$ *and*
$Q = \overline{c}\,\langle hello\rangle\,.c(y)$

When executed in parallel they can interact in the following way:

$$
\begin{aligned}
P|Q \;=\;& \\
& \nu\, k.c(x).\mathtt{if}\ x = \ hello\ \mathtt{then}\ \overline{c}\langle \mathtt{pub}(k)\rangle\ \ \mathtt{else}\ 0 \mid \overline{c}\langle hello\rangle.c(y) && (1)\\
& \equiv \nu\, k.c(x).\mathtt{if}\ x = \ hello\ \mathtt{then}\ \overline{c}\langle \mathtt{pub}(k)\rangle\ \ \mathtt{else}\ 0 \mid \nu\, x.(\overline{c}\langle x\rangle.c(y) \mid \{^{hello}/_{x}\}) && (2)\\
& \overset{\nu\, x.\overline{c}\langle x\rangle}{\rightarrow} \nu\, k.c(x).\mathtt{if}\ \ x = \ hello\ \ \mathtt{then}\ \ \overline{c}\langle \mathtt{pub}(k)\rangle\ \ \mathtt{else}\ \ 0 \mid c(y) \mid \{^{hello}/_{x}\} && (3)\\
& \overset{c(hello)}{\rightarrow} \nu\, k.\mathtt{if}\ \ hello = \ hello\ \ \mathtt{then}\ \ \overline{c}\langle \mathtt{pub}(k)\rangle\ \ \mathtt{else}\ \ 0 \mid c(y) \mid \{^{hello}/_{x}\} && (4)\\
& \rightarrow \nu\, k.\overline{c}\langle \mathtt{pub}(k)\rangle \mid c(y) \mid \{^{hello}/_{x}\} && (5)\\
& \equiv \nu\, k.\nu\, y(\overline{c}\langle y\rangle \mid \{^{\mathtt{pub}(k)}/_{y}\}) \mid c(y) \mid \{^{hello}/_{x}\} && (6)\\
& \overset{\nu\, y.\overline{c}\langle y\rangle}{\rightarrow} \nu\, k.\{^{\mathtt{pub}(k)}/_{y}\} \mid c(y) \mid \{^{hello}/_{x}\} && (7)\\
& \overset{c(\mathtt{pub}(k))}{\rightarrow} \nu\, k.\{^{\mathtt{pub}(k)}/_{y}\} \mid \{^{hello}/_{x}\} && (8)
\end{aligned}
$$

We can observe how the structural equivalence rule Alias *allows the introduction of an active substitution (steps 1-2 and 5-6) and hence of a handle variable which can be used to stand for a term. In fact, terms cannot be directly sent on channels. The output of the newly introduced restricted variables is obtained by applying the rule* Open-Atom *(steps 2-3 and 6-7). The rule* Open-Atom *frees the variables and makes them available for input (steps 3-4 and 7-8). Notice how the frame records the information exchanged by the processes.*

2.2 Equivalence Relations

Syntax and semantics of the applied pi-calculus enable us to describe concurrent processes and their behaviour. The context we introduced earlier represents an attacker, i.e. any process which may follow or not the protocol rules and interact with our "well behaving" processes. What we miss now is a way to define the security properties we want to verify. Security properties are often defined in terms of equivalences between processes, i.e. as equivalences between an ideal process which by definition satisfies the security property and the process modelling the protocol to be verified against that property. A proof of the equivalence of the

two processes is a proof that the examined protocol satisfies the desired security property. Several kinds of equivalence relations can be defined. In this Section we present the definition of observational equivalence, static equivalence, trace equivalence and labelled bisimilarity, in Section 2.3.1 we will introduce a formalisation of privacy related properties in applied pi-calculus.

Definition 2.1. *Observational equivalence* We write $A \Downarrow a$ if A can send a message on the channel a, that is, if $A \rightarrow^* C[\overline{a}\langle M\rangle.P]$ for some evaluation context $C[\_]$ that does not bind a. *Observational equivalence* ($\approx$) is the largest symmetric relation $\mathcal{R}$ between closed extended processes with the same domain such that $A \mathcal{R} B$ implies:

1. if $A \Downarrow a$ then $B \Downarrow a$
2. if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B'
3. $C[A] \mathcal{R} C[B]$ for all closing evaluation context $C[\_]$

Intuitively, observational equivalence captures the idea that two processes are indistinguishable if their observable behaviour is the same. The observable behaviour of a process is what an attacker (context) can observe and deduce using the equivalence relations on terms when the process interacts with the environment i.e. outputs on free channels. Note that the processes are required to be in the relation $\mathcal{R}$ for all closing contexts, this intuitively means for all possible instantiations of the free variables. For example, let $A = \texttt{if}\ sdec(k,x) = a\ \texttt{then}\ 0\ \texttt{else}\ \overline{c}\langle b\rangle$ and $B = \overline{c}\langle b\rangle$, A and B are not observationally equivalent for $C = \_ \mid \{^{enc(k,a)}\}/_x$ since $B \Downarrow c$ and A does not. However, B is observationally equivalent to $A' = \nu\, k.\texttt{if}\ sdec(k,x) = a\ \texttt{then}\ 0\ \texttt{else}\ \overline{c}\langle b\rangle$ since now for all closing evaluation contexts $sdec(k,x) \neq a$ and hence $A' \Downarrow c$ iff. $B \Downarrow c$.

Definition 2.2. *Static equivalence.* Two closed frames ϕ and ψ are *statically equivalent*, denoted $\phi \approx_s \psi$, if $dom(\phi) = dom(\psi)$ and there exists a set of names $\tilde{n}$ and substitutions σ, τ such that $\phi \equiv \nu\, \tilde{n}.\sigma$ and $\psi \equiv \nu\, \tilde{n}.\tau$ and for all terms

M, N such that $\tilde{n} \cap (fn(M) \cup fn(N)) = \emptyset$, we have $M\sigma =_E N\sigma$ holds if and only if $M\tau =_E N\tau$ holds. We say that two closed extended processes A and B are *statically equivalent* ($A \approx_s B$), when their frames are statically equivalent, denoted $\phi(A) \approx_s \phi(B)$.

Static equivalence captures the equivalence of two processes at a given stage of their execution, taking into account their output so far. The idea is that if an attacker cannot distinguish two terms applying the active substitutions generated so far by the two processes when interacting with the environment then the two processes are statically equivalent.

Example 2.4. *As an example, let's consider the following frames:*
$\phi_1 = \nu\, k.\{^{pub(k)}/_{y}\} \mid \{^{hello}/_{x}\}$,
$\phi_2 = \nu\, k.(\{^{k}/_{x}\} \mid \{^{pub(k)}/_{y}\})$ *and*
$\phi_3 = \nu\, l.\{^{pub(l)}/_{y}\} \mid \{^{hello}/_{x}\}$
We have that $\phi_1 \approx_s \phi_3$ *since they are structurally equivalent but* $\phi_1 \not\approx_s \phi_2$, *since* ϕ_2 *satisfies* $y = pub(x)$ *and* ϕ_1 *does not.*

Definition 2.3. *Trace equivalence* ($\sim$). Let

$$tr_A = A_0 \overset{\alpha_1}{\Rightarrow} A_1 \overset{\alpha_2}{\Rightarrow} \cdots \overset{\alpha_{n-1}}{\Rightarrow} A_{n-1} \overset{\alpha_n}{\Rightarrow} A_n$$

And

$$tr_B = B_0 \overset{\alpha_1}{\Rightarrow} B_1 \overset{\alpha_2}{\Rightarrow} \cdots \overset{\alpha_{n-1}}{\Rightarrow} B_{n-1} \overset{\alpha_n}{\Rightarrow} B_n$$

be two traces. We say that tr_A and tr_B are *trace equivalent*, $tr_A \sim tr_B$, if $A_i \approx_s B_i$ for all i.
Two processes A and B are *trace equivalent*, if for every trace tr_A of A there exists an equivalent trace tr_B of B and if for every trace tr_B of B there exists an equivalent trace tr_A of A.

Trace equivalence relates processes which can mimic each other's execution paths but it does not enforce that internal decisions happen at the same point. A stronger

equivalence relation, labelled bisimilarity, requires processes to be able to mimic each other through the stages of their execution. While trace equivalence is weaker and naturally captures the intuition of equivalence of processes it is more difficult to verify than observational equivalence which is a stronger equivalence relation.

Definition 2.4. *Labelled bisimilarity* ($\approx_l$) is the largest symmetric relation $\mathcal{R}$ on closed extended processes such that $A \mathcal{R} B$ implies:

1. $A \approx_s B$
2. if $A \to A'$ then $B \to^* B'$ and $A' \mathcal{R} B'$ for some B'
3. if $A \xrightarrow{\alpha} A'$ and $fv(\alpha) \subseteq dom(A)$ and $bn(\alpha) \cap fn(B) = \emptyset$; then $B \to^* \xrightarrow{\alpha} \to^* B'$ and $A' \mathcal{R} B'$ for some B'.

Abadi and Fournet present the following result [AF01]:

Theorem 2.5. *Observational equivalence is labelled bisimilarity:* $\approx = \approx_l$

Which is important from the verification point of view, because labelled bisimilarity can be proved instead of observational equivalence avoiding to have to deal with the universal quantification over evaluation contexts required by the observational equivalence.

However, it turns out that for the theorem to hold it should further be required that active substitutions can only be defined on terms which are not of channel sort otherwise Bengtson et al. give a counter-example [BJPV11] to Theorem 2.5 can be found. Indeed, if we consider $A = \nu\, c.(\overline{c}\langle a\rangle.\overline{d}\langle a\rangle \mid \{^c/_x\})$ and $B = \nu\, c.(0 \mid \{^c/_x\})$, we have that A and B are bisimilar because their frames are the same and both have no transitions. However, A and B are not observationally equivalent because the context $C = x(y)$ can distinguish them since $A \mid C \Downarrow_d$ but $B \mid C \not\Downarrow_d$. Proofs of coincidence of labelled bisimilarity and observational equivalence are given by Liu in [Liu11] and Arapinis et al. in [ALRR14].

2.3 Modelling Security Properties

Security properties can be defined in different ways, which can be categorized depending on the type of property:

- *Reachability properties* are properties that can be verified by checking every single execution trace of a process. If every possible execution trace satisfies the property condition we say that the property holds. Other kind of reachability properties may require that a particular state can or not be reached. An example of reachability property is *weak secrecy*. Weak secrecy requires that for any adversarial context there is no execution trace leading to the secret being revealed on a public channel. Formally, for all evaluation contexts C, $C[P] \not\rightarrow^* C[\bar{c}\langle s\rangle.Q]$ where s is the secret and c is a public channel.

- *Correspondence properties* are properties that can be verified by checking that for every execution trace the event e happens before the event e'. An example of correspondence property is *server authentication*. Server authentication requires that for all execution traces if the event $servicedelivered(x_A, k)$, which indicates that the service was delivered to the client x_A using the session key k, happens then the event $servicerequested(A, x_k)$, which indicates that A requested a service during a session protected by the session key x_k, happened before on the same trace. Formally, we explicitly define the events and their parameters in the model of the protocol. A simplified example is
  $Server \equiv \nu\, k.c(x_A).C[\overline{servicedelivered}\langle x_A, k\rangle]$
  $Client \equiv C'[\overline{c}\langle A\rangle.\overline{servicerequested}\langle(A, x_k)\rangle.P]$
  where we require $servicedelivered(x_A, k) \rightsquigarrow servicerequested(A, x_k)$ and $A = x_A, k = x_k$. This property ensures to the client A that if it believes to have ended a session with a server using a certain key k then that server

started a session with the client A using the key k. Other examples of security properties which can be defined as correspondence properties are client authentication, mutual authentication and integrity.

- *Equivalence properties* are properties that can be verified by checking that from an attacker point of view the real protocol looks the same as (i.e. is indistinguishable from) an ideal one which satisfies the desired property by construction. As an example a secrecy property could require the real system where a secret s is encrypted and sent over a public channel to be indistinguishable (i.e. observationally equivalent) from an ideal system where the secret is not sent at all (and hence satisfies the secrecy of s by construction). Formally, we could require the following observational equivalence to hold $C[\nu\, s.\overline{c}\,\langle senc(k,s)\rangle\,.P] \approx C[\nu\, r.\overline{c}\,\langle r\rangle\,.P]$. Note that here we assume that $senc$ is a non deterministic symmetric encryption function and that in the ideal process we send a random nonce r to model an ideal version of it. Other examples of security properties which can be defined as equivalences are privacy in electronic voting, strong and weak unlinkability and strong and weak anonymity.

2.3.1 Privacy Related Properties

In this section, we present the formalization of privacy related properties as given by Arapinis et al. in [ACRR10], namely strong unlinkability, weak unlinkability and strong and weak anonymity. For simplicity and because the strong properties are the focus of this thesis, we omit the formal definitions of the weak properties, we give an informal definition instead. Note that, the strong untraceability [ACRR09] and the strong unlinkability [ACRR10] definitions coincide, though the definition of strong unlinkability [ACRR10] is given in a more general way.

Definition 2.6. *Well-formed protocol.* A p-party protocol is said to be **well-formed** if it is a closed plain process P of the form:

$$\begin{aligned} P &\equiv \nu\, \tilde{n}.(!R_1 \mid \cdots \mid !R_p) \\ \forall i \in \{1,\ldots,p\}\ R_i &\equiv \nu\, id.\nu\, \tilde{m}.init_i.!(\nu\, s.main_i) \end{aligned}$$

Where:

1. names and variables never appear both bound and free in P
2. each name and variable is bound at most once in P
3. for all i, $init_i$ and $main_i$ are any two sub-processes (possibly empty) such that P is a closed plain process

Intuitively, each one of the R_i processes represent a participant of the protocol. For example, server and client are two parties involved in most security protocols.

Example 2.5. *Let's consider the following system:*

$$P \stackrel{def}{=} \nu\, k.(!S \mid !C)$$

Where the processes S and C share a key k and are defined as follows:

$$\begin{aligned} S &\stackrel{def}{=} init_S.!main_S \\ C &\stackrel{def}{=} \nu\, id.init_C.!main_C \end{aligned}$$

where $init_S$ and $init_C$ are empty processes and

$$\begin{aligned} main_S &\stackrel{def}{=} \overline{c}\,\langle req\rangle\,.c(x) \\ main_C &\stackrel{def}{=} \nu\, id.c(y).\mathbf{if}\ y = req\ \ \mathbf{then}\ \overline{c}\,\langle senc(k,id)\rangle \end{aligned}$$

We can think of S as being a server which identifies the clients C prior to offering some service. Server and clients share a secret key which is used to protect the clients' anonymity from external adversaries.

Definition 2.7. *Strong Unlinkability.* Let Σ be a signature and E an equational theory for this signature and let P be a well-formed protocol over Σ, as defined in Definition 2.6. For all $i \in \{1, \ldots, p\}$, we build the protocol P^{R_i} over Σ as follows:

$$\begin{aligned} P^{R_i} &\triangleq \nu\, \tilde{n}.(!R_1 \mid \cdots \mid !R_{i-1} \mid !R_i'' \mid !R_{i+1} \mid \cdots \mid !R_p) \\ R_i'' &\triangleq \nu\, id.\nu\, \tilde{m}.init_i.main_i \end{aligned}$$

P preserves *strong unlinkability* of R_i if $P \approx_l P^{R_i}$

Informally, strong unlinkability requires a system in which agents execute multiple times to look the same as a system in which agents execute at most once. The weak definition of unlinkability instead considers a system to preserve *weak unlinkability* if an attacker cannot link two messages as being part of two different sessions executed by the same principal, hence in this case the attacker can distinguish if a principal has executed the protocol more than once but not when. The weak unlinkability properties is given as a non standard trace equivalence property which makes use of annotated traces and would require some preliminary definitions to be set. Both strong and weak unlinkability are not currently fully supported by the automated proof tool `ProVerif`. In particular, strong unlinkability require careful modelling since the structure of the pair of processes to be verified (the real and the ideal) is not symmetric. Weak unlinkability is defined in terms of a non standard trace equivalence and it is not currently supported by `ProVerif`.

Example 2.6. *Let's consider the system introduced in Example 2.5:*

$$P \stackrel{def}{=} \nu\, k.(!S \mid !C)$$

where

$$\begin{aligned} S &\stackrel{def}{=} !main_S \\ C &\stackrel{def}{=} \nu\, id.!main_C \\ main_S &\stackrel{def}{=} \overline{c}\,\langle req\rangle\,.c(x) \\ main_C &\stackrel{def}{=} \nu\, id.c(y).\mathbf{if}\ y = req\ \mathbf{then}\ \overline{c}\,\langle senc(k, id)\rangle \end{aligned}$$

Server and clients share a secret key which is used to protect the clients' anonymity from external adversaries. However, note that the use of encryption is not ensuring unlinkability, because multiple uses of the service by the same user can be linked by an external observer. In fact, the system P can evolve to P_1 as follows:

$$P \overset{\alpha}{\Longrightarrow}^{*} \nu\, k.(!S \mid !C \mid \nu\, id_1.(!main_C \\ \mid \{^{senc(k,id_1)}/_{y_1}\} \mid \{^{senc(k,id_1)}/_{y_2}\}) \mid \{^{req}/_{x_1}\} \mid \{^{req}/_{x_2}\}) = P_1$$

where the client with identity id_1 executes twice. The unlinkability property requires the system P to be observationally equivalent to the system P^*:

$$P^* \overset{def}{=} \nu\, k.(!S \mid !C^*)$$

where:

$$S \overset{def}{=} \overline{c}\,\langle req\rangle\,.c(x)$$
$$C^* \overset{def}{=} \nu\, id.main_C$$

In this case a client can never execute twice (since there is no replication before $main_C$ in the client process) hence the system P^* can evolve as follows:

$$P^* \overset{\alpha}{\Longrightarrow}^{*} \nu\, k.(!S \mid !C^* \\ \mid \nu\, id_1.\{^{senc(k,id_1)}/_{x}\} \mid \nu\, id_2\{^{senc(k,id_2)}/_{y}\} \mid \{^{req}/_{z}\} \mid \{^{req}/_{v}\}) = P_1^*$$

We have that:

$$\phi(P_1) = \nu\, id_1.(\{^{senc(k,id_1)}/_{x}\} \mid \{^{senc(k,id_1)}/_{y}\}) \mid \{^{req}/_{z}\} \mid \{^{req}/_{v}\}$$

and

$$\phi(P_1^*) = \nu\, id_1.\{^{senc(k,id_1)}/_{x}\} \mid \nu\, id_2\{^{senc(k,id_2)}/_{y}\} \mid \{^{req}/_{z}\} \mid \{^{req}/_{v}\}$$

The two frames are not statically equivalent, since $x = y$ holds in $\phi(P_1)$ but not in $\phi(P_1^*)$. Hence, P and P^* are not labelled bisimilar. However a formal proof would

require a more involved argument, we here just notice that the only way for the process P^* to produce the output of an encrypted id ($senc(k, id_i)$) is by unrolling a new client process having a new id and hence could only produce frames of the form of $\phi(P_1^*)$. An attacker could distinguish the two systems, and could link executions involving the same identity in the system P. This happens because we modelled deterministic encryption and hence the encryption of the client's id can be linked to the same client across different sessions.
Let's now consider a simple modification of the system $\overline{P}$:

$$\overline{P} = \nu\, k.(!S\ |!\overline{C})$$

Where we introduce a secret random value r to be sent along with the clients identities (i.e. we model non deterministic encryption):

$$\overline{C} = \nu\, id.!(c(x).\mathbf{if}\ \ x = req\ \ \mathbf{then}\ \ \nu\, r.\overline{c}\,\langle senc(k, \langle r, id\rangle)\rangle)$$

The system $\overline{P}$ can evolve to $\overline{P}_1$ as follows:

$$\overline{P} \overset{\alpha}{\Longrightarrow}{}^{*}\ \nu\, k.(!S\ |!C\ |\ \nu\, id_1.(!c(x).\mathbf{if}\ \ x = req\ \ \mathbf{then}\ \ \nu\, r.\overline{c}\,\langle senc(k, \langle r, id\rangle)\rangle$$
$$|\ \{^{senc(k,\langle r_1,id_1\rangle)}/_{x}\}\ |\ \{^{senc(k,\langle r_2,id_1\rangle)}/_{y}\})\ |\ \{^{req}/_{z}\}\ |\ \{^{req}/_{v}\}) = \overline{P}_1$$

Where the client with identity id_1 executes twice. The system $\overline{P}$ and P^* are labelled bisimilar, hence $\overline{P}$ preserves strong unlinkability. Once again, a formal proof would require a more involved argument, we here just notice that both systems can never output the same encrypted term twice.

Definition 2.8. *Strong Anonymity.* Let Σ be a signature and E an equational theory for this signature and let P be a well-formed protocol over Σ. P preserves

strong anonymity if $P \approx_l P_{R_i}$, where P_{R_i} is defined as follows:

$$\begin{aligned} P_{R_i} &\stackrel{def}{=} \nu\, \tilde{n}.(!R_1 \mid \cdots \mid !R_p \mid R_w) \\ R_w &\stackrel{def}{=} \nu\, id.\nu\, \tilde{m}.init_w.!(\nu\, s.main_w) \\ init_w &\stackrel{def}{=} init_i\{id_w/id\} \\ main_w &\stackrel{def}{=} main_i\{id_w/id\} \end{aligned}$$

Where id_w is a name not occurring in P.

Informally, strong anonymity requires a system where the user id_w executes the role R_i (represented by the protocol P_{R_i}) to be undistinguishable from a system where the user id_w is not present at all (i.e. the protocol P which is the ideal protocol from id_w anonymity preservation point of view). The *weak anonymity* property instead requires an attacker to not be able to distinguish which identity initiated a transition, that is cannot link the use of services with the users' identity which used them, but may be able to distinguish the presence of different identities. Weak anonymity is defined in terms of a non standard trace equivalence and it is not currently supported by `ProVerif`.

3

Background: Introduction to Mobile Telephony Systems

The ubiquitous presence of mobile communication devices and the continuous development of mobile data applications, which result in high level of mobile devices' activity and exchanged data, often transparent to the user, make UMTS a good case study from the security and privacy point of view.

GSM (Global System for Mobile Communication) and UMTS (Universal Mobile Telecommunications System) are the most widely used mobile telephony standards with billions of users worldwide. GSM was developed by ETSI (European Telecommunications Standards Institute) in order to promote a common standard for the European cellular telephony as a replacement for the multitude of first generation standards. UMTS is specified and maintained by the Third Generation Partnership Project (3GPP), it was introduced in 1999 to offer a better support for mobile data applications by increasing the data rate and lowering the costs of mobile data communications. Furthermore, UMTS offers an improved security architecture with respect to previous mobile communication systems such as GSM. Both GSM and UMTS have been improved and extended several times. We will

use the terms 2G and 3G to indicate the wider set of standards including GSM and UMTS respectively. Most of protocols and issues presented in the following chapters are common to both 2G and 3G systems, when this will not be the case, it will be explicitly pointed out. In the rest of this thesis we will adopt a unified terminology to address mobile telephony systems components, however 2G and 3G standards may not use the same terminology even when addressing components having the same functions/purposes.

In the following sections, we will introduce the 2G/3G network architecture and we will describe in more details their security features. We will then summarize some of the well-known weaknesses of mobile telephony systems, along with relevant work on 2G/3G security.

3.1 GSM/UMTS Architecture

The 2G/3G network architecture, depicted in figure 3.1, integrates both GSM and UMTS components. The user side of the network consists of Mobile Stations (MS), this term is used in mobile telephony systems to indicate both the Mobile Equipments (ME) such as mobile phones, and the so-called SIM, or USIM card (Universal) Subscriber Identity Module in 2G, 3G systems respectively. The (U)SIM card identifies the user as a legitimate subscriber within a mobile telephony operator network. To access the services offered by a mobile operator, a MS connects through radio communication technology to the UTRAN (UMTS Terrestrial Radio Access Network) or GERAN (GSM/EDGE[1] Radio Access Network) network, that is a GSM access network. A mobile station directly communicates with a BTS (Base Transceiver Station) or Node B which covers the area the MS is located in. One or more Nodes B are connected to a Radio Network Controller (RNC), and one or more BTS are connected to a Base Station Controller (BSC)

[1]EDGE (Enhanced Data rates for Global Evolution) is a standard part of the 2G set aiming to provide faster bit rates for data application

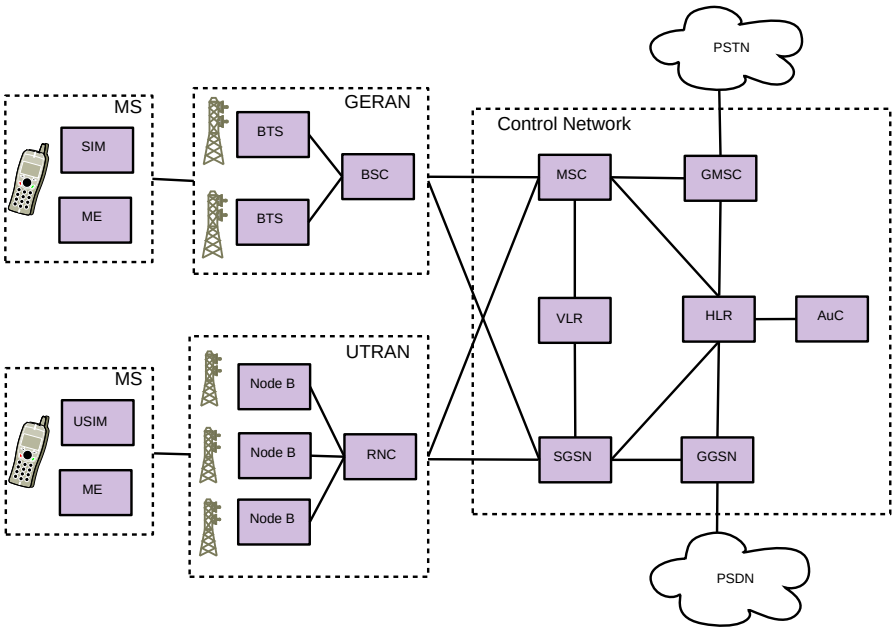

FIGURE 3.1: 2G/3G Architecture

defining a cell. A group of cells forms a Location Area. RNCs and BSCs manage the radio resources and inter-cell handover. They are the interface between the mobile station and the core network. The core network offers circuit-switch and packet-switch services. The Mobile Switching Centre (MSC) and Gateway Mobile Switching Centre (GMSC) offer inter and intra-network circuit-switching domain services, respectively, and interface the 2G/3G systems with the traditional fixed telephony network. The Serving GPRS[2] Support Node (SGSN) and the Gateway Serving GPRS Support Node (GGSN) offer, respectively, inter and intra-network packet-switching domain services as well as connecting 2G/3G networks with the internet. Within the core network the Home Location Register (HLR) stores permanent sensitive information of subscribers such as identities, service profiles, and activity statuses. These informations are linked to the SIM and recorded when stipulating a contract with the Mobile Network Operator (MNO). 2G/3G systems offer roaming capabilities between different mobile network operators, and between the different technologies (provided that the mobile equipment supports

[2]GPRS (General Packet Radio Service) adds packet switch functionalities to GSM

them), meaning that a mobile station can be connected to a visited network, the Serving Network (SN), which might be different from the subscriber's Home Network (HN) and which could be using a different standard. Each subscriber has a long term identifier IMSI (International Mobile Subscriber Identity) that is stored in the (U)SIM and a temporary identifier TMSI (Temporary Mobile Subscriber Identity), allocated by the serving network to protect the subscriber's identity privacy. The Visitor Location Register (VLR) stores temporary informations about subscribers visiting a given location area of the serving network and maintains TMSI/IMSI associations. The network operator and each subscriber share a unique long term secret key used for authentication purposes. This key is stored in the (U)SIM. The Authentication Centre (AuC) is a protected database storing associations between subscriber identities (IMSI) and long-term keys.

3.2 GSM/UMTS Security Features

In the rest of this thesis will usually consider a simplified network architecture so to be able to abstract from the network complexity and concentrate on the communication protocols taking place between a Mobile Station (MS) and the Serving Network (SN). When referring to a Mobile Station we will mean both a mobile phone and the (U)SIM contained in it and belonging to a legitimate subscriber of a mobile telephony service offered by a legitimate provider. When referring to a Serving Network we mean both the UTRAN/GERAN Base Station that is directly communicating with the MS and the complex structure of databases and servers connected with it and forming the 2G/3G core network. Each MS has a unique identity associated with the physical device IMEI (International Mobile Equipment Identity) and a unique identity associated with the (U)SIM (implicitly linked to the subscriber of a contract with a MNO), that is called International Mobile Subscriber Identity (IMSI). When a mobile station connects to a network, a temporary identity (TMSI Temporary Mobile Subscriber Identity)

is assigned to it and is used, instead of the IMSI to identify the MS. To avoid mobile stations traceability, the 2G/3G standards require periodic updates of the temporary identity to be performed.

The intended security features of 2G and 3G systems are slightly different, in fact 3G systems aim to overcome the weaknesses of 2G systems such as the lack of mutual authentication and the use of weak ciphering algorithms.

2G systems aim to provide the following security features [3GP10e]:

- **Subscriber identity confidentiality**: the property that the IMSI is not made available or disclosed to unauthorized individuals, entities or processes.
- **Subscriber identity authentication**: the corroboration by the mobile network communicating with a mobile station that the subscriber identity (IMSI or TMSI), sent by the mobile subscriber to identify itself, is indeed the one claimed.
- **User data confidentiality**: the property that the user information exchanged on traffic channels is not made available or disclosed to unauthorized individuals, entities or processes.

The 3G communication system aims to improve the security features offered to mobile telephony subscribers, requiring for example mutual authentication between the user and the network and specifying more precisely the provided user's privacy properties in terms of identity confidentiality or anonymity, location confidentiality and user untraceability or unlinkability. The security properties stated by the 3G standard are the following [3GP10d]:

- **User Identity Confidentiality** namely:
  - *user identity confidentiality*: the property that the permanent user identity (IMSI) of a user to whom a service is delivered cannot be eavesdropped on the radio access link;

  - *user location confidentiality*: the property that the presence or the arrival of a user in a certain area cannot be determined by eavesdropping on the radio access link;
  - *user untraceability*: the property that an intruder cannot deduce whether different services are delivered to the same user by eavesdropping on the radio access link.

- **Entity Authentication** namely:
  - *user authentication*: the property that the serving network corroborates the user identity of the user;
  - *network authentication*: the property that the user corroborates that he is connected to a serving network that is authorized by the user's HN to provide him services; this includes the guarantee that this authorization is recent.

- **Confidentiality** namely:
  - *cipher algorithm agreement*: the property that the MS and the SN can securely negotiate the algorithm that they shall use subsequently;
  - *cipher key agreement*: the property that the MS and the SN agree on a cipher key that they may use subsequently;
  - *confidentiality of user/signalling data*: the property that user/signalling data cannot be overheard on the radio access interface;

Both 2G and 3G systems rely on an Authentication and Key Agreement (AKA) protocol, on the use of encryption of the confidential data transmitted on the radio channel and on the use of periodically changing temporary identities, in order to achieve the above mentioned security properties. Moreover, 3G systems use integrity protection of the sensitive data. Unlike the 2G authentication protocol, the 3G AKA protocol allows MS and SN to establish not only a ciphering key but also an integrity key, and achieves mutual authentication of MS and HN.

3.3 Tools for Experimental Analysis

In this section we give an overview of some open source tools (hardware and software) available for the experimental analysis of mobile telephony systems or more in general of radio communication systems.

USRP [Ett09] (Universal Radio Peripheral) is a family of relatively cheap hardware platforms which allow to implement software supported radio devices, i.e. radio communication devices that rely on software elaboration of the radio signal. Generally, the USRP is controlled by software installed on a host computer. The host communicates with the USRP board thanks to an Ethernet or USB connection. The schematic of the USRP board as well as the open source drivers are freely available. The USRP can be used to work with different radio bandwidths depending on the chosen transmitter/receiver board plugged into it. The software most commonly used in combination with the USRP for the radio signal manipulation is GNU Radio.

GNU Radio [gnu06] is an open source software development toolkit providing radio signal processing packages to ease the development of software radio systems. It can be used in combination with compatible hardware such as USRP boards or in simulation mode.

Open BTS [Bur] is a software radio implementation of the base station side of the GSM air interface and it uses a SIP softswitch or PBX (i.e. a software switch to connect phone calls from one line to another) hence it combines the GSM air interface with low-cost VoIP for call and message forwarding (and reception). It can be used to implement for example private GSM networks and can be used in combination with USRP and GNU Radio to obtain an inexpensive fully customizable open source GSM network infrastructure.

Open BSC [WFS+] started as an open source implementation of a GSM Base Station Controller (BSC) and evolved into the implementation of a software "GSM network in a box". In fact, it is a software implementation of the main functionalities of many components of the GSM core network including BSC (Base Station Controller), MSC (Mobile Switching Center), HLR (Home Location Register), AuC (Authentication Center), VLR (Visitor Location Register), EIR (Equipment Identity Register). OpenBSC supports some commercially available off-the-shelf base stations.

Osmocom-BB [WMEoc] is an open source implementation of a mobile station GSM protocol stack which can be used with compatible models of mobile phones.

Femtocell is a short range base station for indoor coverage. Commercially available 3G femtocells have been hacked. The hacking of these devices allows the analysis of 3G networks and the exposure of device and protocol vulnerabilities [GRB12, Tom13].

HackRF [Gad] is an open source Software Defined Radio peripheral capable of transmission and reception of radio signals. It enables test and development of radio technologies.

BladeRF [blaa] is a Software Defined Radio (SDR) platform that enables the development of fully customized RF devices from firmware to software. Its transceiver covers a wide spectrum from FM to 4G LTE.

The software and hardware described above make the analysis and exploitation of mobile telephony systems possible for a wider audience by lowering down the costs and providing open source specifications, while so far the available hardware

in terms of base stations and mobile stations was only the one whose design was covered by industry secret.

A Software Defined Radio platform should not end at the hardware, which is why there is such a strong emphasis on documentation and tutorials. Starting with basic radio architecture and spanning into modulation techniques, high throughput USB Linux kernel driver design, basic telecommunication coding schemes, and MIMO, the platform aims to be the perfect tool for learning modern software radio design.

The availability of programmable old GSM mobile phones and the osmocom-BB software allows for example to:

- implement an over-the-air sniffer: to sniff GSM over the air signalling traffic. A phone call sniffer could be implemented as well [NMb], but in this case some hardware modification is required together with the use of some software tool to crack the ciphering key used for the encryption of the phone call;
- locate a mobile phone given a mobile phone number: this technique uses silent SMS messages [NMb] or silent phone calls [KKHK12] to locate the victim;
- retrieve the temporary identity TMSI of a given mobile phone: this can be achieved by using the same technique as the locating attack;
- send legal and forged messages on the air: this can simply be obtained by modifying the osmocom-BB software in order to create the desired messages. This could be done for example, in order to analyse the mobile network reaction to malformed or unexpected messages.

Apart from offering real mobile network operator's functionalities OpenBSC allows further to:

- implement a GSM IMSI catcher to retrieve mobile phone IMSIs and locate mobile phone users;
- send, replay, forge, manipulate messages;
- capture all the incoming and outgoing communications. Note that by using openBSC, it is not possible to implement a full over the air sniffer, i.e. it is not possible to capture the traffic of mobile phones not attached[3] to it, this is due to hardware constraints.
- induce mobile phone to attach and use no encryption for the sensitive traffic.

The USRP/openBTS combination allows to:

- implement a GSM over-the-air sniffer (in this case the hardware behaviour is customizable);
- send, replay, forge, manipulate messages;
- implement a GSM IMSI catcher to retrieve mobile phone IMSIs and locate mobile phone users [Pag10];
- induce mobile phone to attach and use no encryption for the sensitive traffic.

To the best of our knowledge there are no available open source implementations of UMTS base stations or mobile phones, however the USRP board could be programmed in a similar fashion as in the openBTS project to implement an open source UMTS base station.

[3] a MS is attached when a dedicated channel has been assigned to it by the base station

3.4 Previous Work on 2G/3G Security and Privacy

Most of the work on security of mobile telephony systems concerns secrecy, integrity and authentication properties [BSW00, NMb, ASS09]. There are only few formal and experimental studies concerning the level of privacy provided to users by mobile telephony systems.

2G Vulnerabilities.

The identification procedure, consisting in the request of the user identity by the network followed by a cleartext reply containing the user identity, is acknowledged in the standard as a breach of the user identity confidentiality [3GP10d, p. 19, s. 6.2]. This procedure is exploited by the well-known "IMSI catcher" attack, which is the best known attack to mobile telephony users' privacy. It consists in forcing a mobile phone to reveal its identity (IMSI) [Fox97, Str07] by triggering the identification procedure from a fake operator base station (configured with the corresponding mobile network and country code settings). Moreover, the 2G fake base station can then force the attached mobile station to use no encryption for data and signalling communication and hence it can capture ingoing and outgoing plaintext traffic. Until fairly recently, implementing an IMSI catcher required specialised software and equipment such as base stations. However, such devices have become more and more affordable thanks to software emulation [Pag10].

Foo Kune et al. [KKHK12] present a study on the use of the paging procedure to locate mobile telephony users. They perform a tracking attack relying on passive sniffing of paging response messages triggered by placing silent phone calls (obtained by hanging up before the receiving phone rings) to the victim phone. This technique allows to reveal the presence of the victim in an area monitored by the attacker. Munaut and Nohl [NMb] previously outlined a similar technique. They performed a GSM sniffing attack, which allows to eavesdrop a GSM phone call

by using a modification of the osmocom-BB [WMEoc] opensource implementation of the GSM protocol stack and an old Motorola mobile phone. Differently from Foo Kune et al., they used a silent SMS to trigger the paging responses needed to locate the victim. Although these works take advantage of the fact that a TMSI is allocated for a long time window, they do not analyse the security and privacy provided by the TMSI reallocation procedure. Moreover, in order to perform the attack, the adversary needs to know the mobile number of the victim.

Engel showed at the 25C3 conference [Eng] how network signalling messages, triggered when sending/receiving SMS messages, can be used to localize mobile telephony users. He suggests that network operators should use home routing, i.e. forwarding through the home network, as a countermeasure to this SMS tracking attack. This attack requires access to the intra-network communication infrastructure, which although possible may require subscription to a pay per query service. In Chapters 4, 5, we will analyse the privacy of the more exposed over-the-air communication which is available to any attacker with a radio enabled device. We will not assume to have control over the less easily accessible intra-network communication channel.

The *gsmmap* project [NMa, MM] uses a variant of the open source GSM protocol stack developed within the osmocom-BB project to assess and visually render on a map the level of security and privacy provided by network operators across the world. In particular, their aim is to check if network operators are protecting the users from well known attacks by adopting countermeasures such as the use of A5/3 encryption, padding randomization, and full authentication for outgoing calls and SMS to prevent impersonation and interception, and the use of regular TMSI updates, and home routing to prevent Engel's SMS tracking attack.

3G Weaknesses Resulting from 3G/2G-interoperability. 3G mobile stations are as well vulnerable to 2G IMSI catcher, in fact 3G systems are fully interoperable with 2G systems, and 3G mobile stations can roam in 2G networks.

A 3G mobile stations can be induced to attach to a 2G base station by broadcasting a stronger signal with respect to the 3G one. To avoid this, 3G mobile stations can be configured to use only 3G networks. To the best of our knowledge the only implementation of a pure 3G IMSI catcher is the one presented in [GRB12] and is realised using a modified femtocell. Previously proposed attacks on 3G security exploit the vulnerabilities which are propagated from GSM to 3G when providing interoperability between the two systems. Most of the reported attacks of this kind take advantage of well-known weaknesses of the GSM authentication and key agreement protocol, such as the lack of mutual authentication and the use of weak encryption. These attacks allow an active attacker to violate the user identity confidentiality, to eavesdrop on outbound communications [MW04] and to masquerade as a legitimate subscriber obtaining services which will be billed on the victim's account [ASS09]. However, these attacks cannot be carried out on pure 3G networks, because they rely on the lack of mutual authentication in GSM and on the possibility of downgrading the communication from 3G to GSM.

3G specific. To the best of our knowledge, the only attack that does not rely on GSM/3G interoperability has been presented by Zhang and Fang in [ZF05]. This attack is a variant of the false base station attack and takes advantage of the fact that the mobile station does not authenticate the serving network. It allows the redirection of the victim's outgoing traffic to a different network, for example a network which uses a weaker encryption algorithm or one which charges higher rates than the victim's one. Zhang and Fang's attack concerns impersonation, service theft, data confidentiality and privacy. The attacks they propose assume an active attacker controlling a rogue base station and a mobile phone in order to perform man-in-the-middle attacks and redirection of the over-the-air communication.

Previous Formal Analysis of 3G protocols The 3G AKA protocol in its pure form (i.e. with no GSM support) has been formally proved to meet some of the specified security requirements [3GP01], such as authentication and confidentiality of data and voice communication. However, privacy related properties such

as unlinkability and anonymity, which are the focus of our work, are not analysed in [3GP01]. The framework applied in [3GP01] cannot be used to specify unlinkability and anonymity properties, let alone reason about them. The formal framework we used allows us to precisely define and verify privacy related properties. Hence, we can discover privacy attacks on the modelled protocols and propose solutions which we then formally prove to satisfy the desired privacy properties.

Other Work on 3G Privacy Enhancement A new framework for authentication has been proposed to provide subscriber privacy with respect to the network [KO06]. In particular, the authors aim to achieve MSs anonymity with respect to the serving network, and location privacy of mobile stations with respect to the home network. To achieve this purposes, they propose a new mechanism for the location update and a three way handshake protocol, to be used instead of the currently used 3G AKA protocol. However, this work is not supported by a formal model of the AKA protocol, nor does it provide a formal verification of the properties enforced by the proposed protocols. Moreover, their attacker model considers the network as not fully trusted, while in this thesis we are only concerned with third party attackers controlling the radio link communications.

3.5 Our Work on 2G/3G Privacy Analysis

In Chapters 4, 5, and 6 we will investigate the privacy provided by 2G and 3G system. In particular, in Chapter 4 we recall a well known privacy weakness of mobile telephony which makes it possible to break user anonymity, and we reveal possible privacy leakages produced by signalling procedures of the 2G/3G protocol stack. Further, we perform some experiments to test the effectiveness of the changing pseudonym mechanism used by mobile telephony systems. In Chapter 5, we show two attacks to the user's privacy. The first one is a replay attack to the TMSI reallocation procedure and affects both 2G and 3G systems, the second affects only 3G

systems and takes advantage of the information leakage resulting from the error messages produced by the execution of the AKA protocol. Finally, in Chapter 6 we provide privacy enhancing versions of the procedures analysed in Chapters 4 and 5 and we give guarantees on the achieved privacy properties thanks to formal verification. For this purpose we will use both automatic and manual techniques and point out advantages and disadvantages of both approaches.

4

Analysis of Mobile Systems' Identity Management

In this chapter we describe some of the mobile systems protocols that are dealing with the identity of mobile phone users and point out breaches of the user's privacy, which expose a subscriber's identity and allow an attacker capable of sending and receiving messages over-the-air to identify the presence of a target mobile phone (MS) in a monitored area, or even track its movements across a set of monitored areas. As we will see, the attacker does not need to know any keys, nor perform any cryptographic operation. In fact, the vulnerabilities exposed in this chapter take advantage of the poor management of the user identities, due to the fact that at the time mobile telephony systems were designed, active attacks were considered too costly and hence unlikely to be performed. In fact, mounting an attack required very expensive equipment such as a base station. However as we briefly discussed in Section 3.3, nowadays the development of open source devices and software has lowered significantly these costs.

We assume that the attacker has unlimited access to the radio link between the mobile station and the base stations but no access to the communication within the

core network or between the base stations and the core network. More precisely, we assume a 'Dolev Yao' attacker [DY81] who has full access only to the radio link, where he can sniff, inject, replay, and modify[1] formal analysis messages, but cannot break the cryptography involved in the protocol, *i.e.* cannot encrypt/decrypt without knowing the required encryption/decryption keys.

We only consider a simplified network architecture, since we are interested in third party attackers having access only to the radio path and not to the wired network infrastructure. This architecture involves simply the mobile stations and the network. The network models both the UTRAN/GERAN Base Station that the MS is directly communicating with and the complex structure of databases and servers connected with it and forming the UMTS/GSM control network. In particular we do not distinguish between serving network and home network. Hence we abstract away from any communication within the network and model only communication between mobile stations and the network. This abstraction allows us to hide details which are uninteresting for the purposes of our analysis and keep the models used for verification small, but at the same time precisely model the interactions over-the-air between MS and network, which are the subject of our analysis. In the rest of this chapter we will present two procedures of the mobile telephony protocol stack that involve the mobile station identity, namely the identification procedure and the paging procedure and we will show how these procedures can be used to violate a user's privacy. Furthermore we will present the TMSI reallocation procedure that is the procedure implementing the pseudonym changing mechanism adopted by mobile telephony systems in order to provide anonymity and unlinkability. We will close this chapter by experimentally evaluating the effectiveness of the pseudonym changing mechanism on real networks.

In describing the protocols we use SMALL_CAPS for constants and *italic* for variable message parameters. The content of encrypted messages is enclosed in curly

[1]The attacker considered in our analysis is quite powerful. However, the attacks we propose do not require all the characteristics of the DolevYao attacker in particular for example none of the attacks require modification of the messages over-the-air

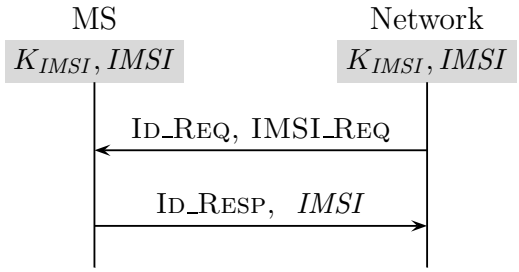

FIGURE 4.1: 2G/3G Identification Procedure. The network sends an identity request to the MS on a dedicated channel. The IMSI_REQ parameter specifies that the requested identity is the long term identity, IMSI. The identity $IMSI$ of the mobile phone is sent in clear on the radio path.

brackets {}.

4.1 Identification Procedure

The identification procedure [3GP10c] allows the network to request the credentials of a mobile station. In particular, the network can request a mobile station's IMSI, that identify the user's SIM card or better the user as a legitimate one having a contract with the MNO, or a mobile station's IMEI, that identify the specific handset. The identification procedure is always initiated by the network. It is typically used when a mobile station connects to a network for the first time. Once the network identifies the MS, it can initiate a procedure to establish a shared encryption key and then usually assigns a temporary identity, TMSI, to the MS. The temporary identity is transmitted to the MS in an encrypted fashion and is used instead of the IMSI in the following communications.

The identification procedure consists of two messages: the identity request message and the identity response message. The parameter IMSI_REQ specifies that the identity requested by the network is the IMSI. The identity response message is sent in clear along with the requested identity. (see Figure 4.1).

4.1.1 Identification Procedure Attack

The identification procedure is acknowledged in the 2G standard as breaching the user identity confidentiality. The same procedure is used by 3G systems as well. It allows a passive attacker to overhear an MS's IMSI. Moreover, any attacker controlling a tool able to act as a base station can trigger at any moment the identification procedure and capture the identity response, and thus can trace the presence of a particular subscriber in the range of the device. Moreover, he can track the movements of the subscriber by installing tracking devices in different areas. This attack to the subscribers' privacy is known as IMSI catcher attack [Fox97, Str07, Weh09]. Typically, the attacker would control a fake BS that advertise to be a legitimate MNO's base station so that its subscribers would attach to it. Once a subscriber attaches, the fake BS just sends an identity request and waits for the response.

4.2 Paging Procedure

The paging procedure defined in [3GP11b] is used to locate a mobile station in order to deliver a service to it, for example an incoming call. The network sends paging requests in all the most recently visited location areas in order to find the mobile station. The paging request message is sent on a Common Control Channel (CCCH) and contains the identity of one or more mobile stations. The paging procedure is typically run after a TMSI is assigned to the mobile station by the network, and hence the TMSI should be used to identify a MS in the paging request. However, the IMSI can be used when the TMSI is not known by the network.

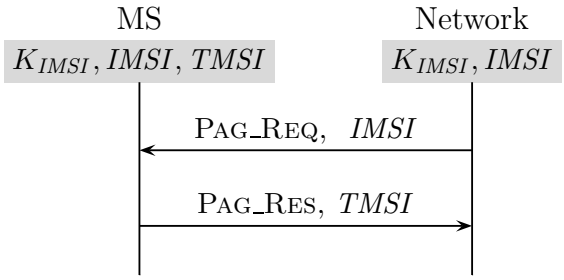

FIGURE 4.2: 3G IMSI Paging Procedure: the paging request message is broadcast on a common radio channel and contains the identity of the paged mobile station (in this case the $IMSI$) in cleartext. The paging response message contains the mobile phone's temporary identity $TMSI$ and is sent in clear on a dedicated channel established between the MS and the network (i.e. messages exchanged on this channel are meant to be originated by and or addressed to this particular MS).

A mobile station receiving a paging request message establishes a dedicated channel to allow the delivery of the service and sends a paging response message containing the most recently assigned TMSI (see Figure 4.2).

4.3 TMSI Reallocation Procedure

A mobile station (MS) is uniquely identified by means of its IMSI. To avoid over-the-air attackers from identifying and linking a user's transactions, a temporary identity called TMSI is assigned by the network and is used to identify the mobile station in protocol messages sent in clear over-the-air. The mobile station identity (its TMSI, if available, or its IMSI) is always included in the first message sent from the MS to the network after the establishment of a dedicated channel. This allows the network to identify the MS before delivering a service to it. For example, the identity is carried in location update requests, CM (Call Management) requests, and paging responses. The use of TMSIs avoids the exposure of the long term unique identity (IMSI) and hence provides third-party anonymity to mobile telephony subscribers. The 3GPP standard specifies that a new TMSI should be

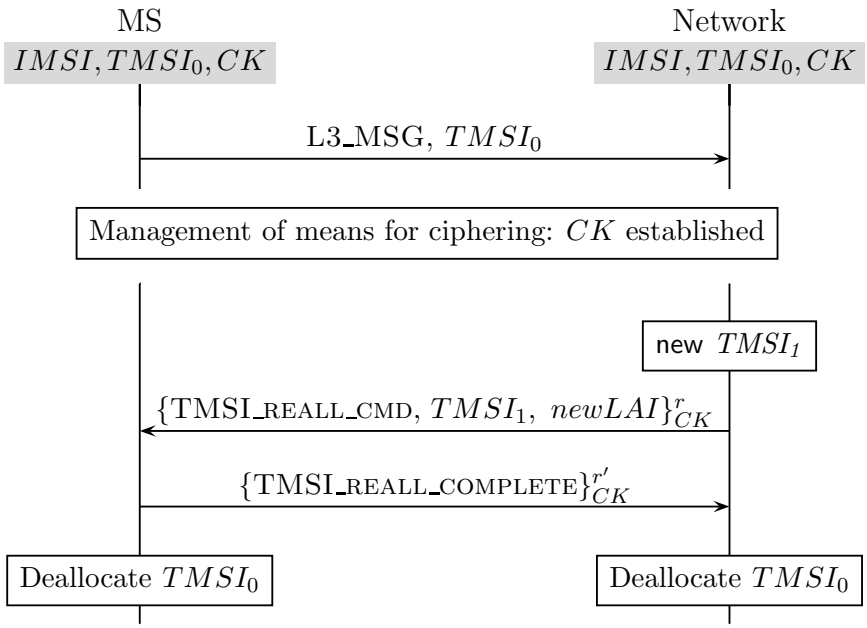

FIGURE 4.3: 2G/3G TMSI Reallocation Procedure: the TMSI reallocation procedure is always initiated by the network. The new TMSI, $TMSI_1$, along with the current Location Area Identifier, $newLAI$, is sent in an encrypted message to the mobile station in order to avoid users' linkability. The TMSI reallocation procedure is always executed on a dedicated channel

assigned at least at each change of location area. Besides this constraint, the choice of how often a new assignment is performed within a location area is left to the network operators [3GP10c]. In order to prevent an adversary linking the old TMSI with the new one, the assignment of a new TMSI is performed in ciphered mode. The session key used to encrypt the new TMSI is established by executing the AKA protocol.

The TMSI reallocation procedure assigns a new pseudonym (TMSI) to a mobile station. The new TMSI is sent to the mobile station in an encrypted fashion. Figure 4.3 depicts the TMSI reallocation procedure as defined in the 3GPP standard [3GP10c, 3GP10d]:

- The mobile station sends a first message on a dedicated channel. This message contains the current MS's temporary identity $TMSI_0$;

- upon receipt of this message, the network can identify the MS and establish means for ciphering of the subsequent communication on the dedicated channel;
- the rest of the communication is then encrypted and consists of a TMSI reallocation command message containing a new pseudonym $TMSI_1$ chosen by the network and the current location area $newLAI$ (the area within which $TMSI_1$ is meaningful);
- this message is followed by a TMSI reallocation complete message which is sent by the MS to acknowledge the completion of the reallocation procedure.

If the network does not receive the expected acknowledgment from the MS, it maintains both $TMSI_0$ and $TMSI_1$ as valid pseudonyms for the the IMSI. The network can perform a TMSI reallocation at any time whilst a dedicated channel is established. The standard does not fully specify how often this procedure should be performed. However, it mandates that it should at least be performed at each change of location [3GP10c]. The standard defines two options for the management of the means for ciphering (i.e. to establish the ciphering key CK):

(1) either a fresh ciphering key is established by executing the 2G or 3G authentication procedure (see 3G AKA protocol Section 5.2.1 for more details);

(2) or a previously established ciphering key can be restored by means of the security mode set-up procedure, which allows the MS and the network to agree on a ciphering algorithm.

4.4 Subscriber Privacy Analysis

In this section we will report on the privacy weaknesses of the paging procedure using IMSI and of the TMSI reallocation procedure. In particular, we will show

how it is possible to use the paging procedure using IMSI to link the IMSI and the TMSI currently used by a MS and hence to locate and trace a mobile telephony users. Secondly, we will experimentally analyse the use of TMSIs in mobile networks and highlight critical scenarios from the privacy point of view.

4.4.1 IMSI Paging Attack

The possibility of triggering a paging request for a specific IMSI allows an attacker to check a specific area for the presence of mobile stations whose identity is known to the attacker and to correlate IMSI and TMSI. The observation of the related paging response allows the correlation of the victim's IMSI with his current TMSI. In practice, an attacker would need to confirm the link between the paged IMSI and the related TMSI by replaying the attack several times.

4.4.2 Experimental Analysis

If a third party that eavesdrops on the radio link was able to identify wireless messages as coming from a particular mobile phone, he would be able to track the location of the mobile phone user in real-time. This could lead to stalking and To ensure confidentiality of a newly assigned TMSI, it is transmitted encrypted using a session key. Those session keys are established through an *authentication and key agreement procedure* (both in GSM and 3G), but for efficiency reasons, a previously established session key can be restored, to be used in several subsequent sessions, by means of the security mode set-up procedure.

Our aim is to analyse what conditions are required in order for this arrangement to guarantee user privacy as intended. In particular, two aspects appear to be important:

1. TMSI reallocation will protect user privacy only if TMSIs are re-allocated often enough, and at the right times (e.g., when users move between locations). The 3GPP standard does not rigorously define the conditions under which TMSI reallocation takes place, leaving the choice to the network operators. This potentially leaves users open to privacy abuses.

2. The success of TMSI reallocation requires that a wireless eavesdropper cannot link the new TMSI to the old one. Encrypting the TMSI in the allocation message is necessary (but may not be sufficient) to ensure that. It turns out that other factors, in particular the use of fresh session keys for each TMSI reallocation, are also necessary to guarantee unlinkability of old and new TMSIs. The 3GPP standard does not mandate this, again leaving user privacy subject to choices made by network operators.

We monitored over-the-air communications of idle and active MSs in order to understand how real networks implement user identity confidentiality through the use of TMSIs, both in terms of frequency of reallocation, and ciphering keys used. Our experiments confirm that the reuse of previously established keys is a commonly adopted policy. However, we show that in case the reuse of encryption keys is adopted for the execution of the TMSI reallocation procedure, this enables a linkability attack which makes it possible to link old and new TMSIs.

Our experiments were carried out using an old GSM Motorola C115 mobile phone in France, UK, Greece, and Italy and using SIM cards from all the major UK, Greek, and Italian network operators.[2]

4.4.2.1 Experimental Settings and Scenarios

The Motorola C115 has a TI Calypso baseband chipset which is supported by the Osmocom-BB project [WMEoc]. The Osmocom-BB project includes an open

[2]More specifically, we used O2, T-Mobile, Vodafone, and Orange in the UK; Vodafone and Wind in Greece; Bouygues and Orange in France; and Wind, Vodafone and TIM in Italy.

FIGURE 4.4: Experimental Tools

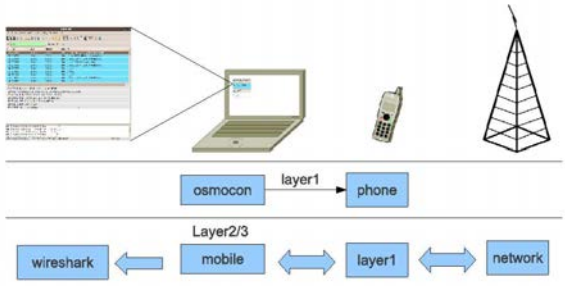

FIGURE 4.5: Osmocom-BB architecture

source implementation of the GSM baseband and various other applications aiming to implement a GSM mobile station. The radio communication functions are implemented in the firmware which is flashed from a laptop into the mobile phone through the Osmocon software, by means of a T191 unlock cable (Figure 4.4). The firmware implements layer 1 of the GSM protocol stack, while layers 2 and 3 are implemented in specialised applications running on the laptop and communicating with the mobile phone through the T191 cable (Figure 4.5). In particular, we used the 'mobile' application which implements layer 2 and 3 of the GSM protocol stack to provide all the basic functions of a mobile phone (network registration, location update, making and receiving calls, and sending and receiving SMSs). The mobile phone activities are logged on a shell terminal and the radio communication is encapsulated in UDP packets sent to a configurable IP address. This traffic can be captured through the Wireshark network traffic analyser [Com]. Interactions

with the mobile phone are enabled by a telnet command interface. This allows one to manually select a network, start phone calls, send SMSs and service requests, etc. We captured over-the-air messages using the 'mobile' application in different

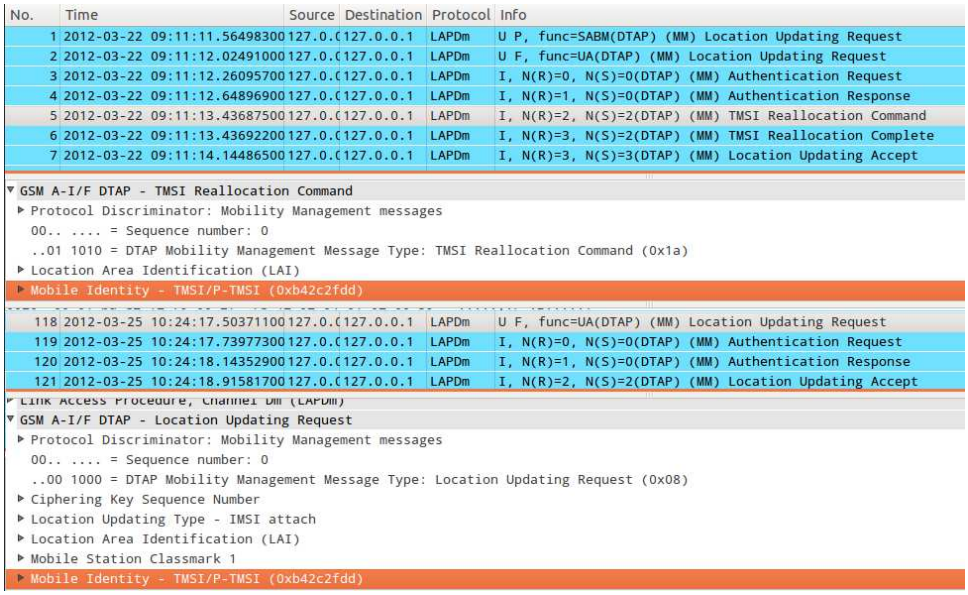

FIGURE 4.6: Trace of a UK Vodafone SIM card obtaining a new TMSI (0xb42c2fdd) on 22/03/12. The same TMSI is still in use on 25/03/12 after 3 days from its allocation.

settings: (1) mobile station in idle state and not moving; (2) mobile station in idle state and moving across two urban areas; (3) mobile station involved in activities such as receiving or starting phone calls, receiving or sending SMSs, and requesting services as for example call diversions.

Since the 3GPP standard merely gives guidelines, real networks differ in the implementation details of the TMSI reallocation. To understand if the different implementations achieve the privacy guarantees they were intended for, we analysed the traffic captured with the mobile application. In particular, we are interested in finding out if the frequency of TMSI reallocation execution is high enough to defeat passive and active tracking attacks, if the policy of changing TMSI at least at each change of location is actually implemented so to obtain at least location

dependent privacy, and if the frequency of execution of the TMSI reallocation procedure is related to the amount of activity of the MS (i.e., to how often the TMSI is exposed to overhearing).

4.4.2.2 Findings/Results

The 3GPP standard relies on the use and frequent reallocation of TMSIs in order to provide user's untraceability. In particular, it mandates that the TMSI reallocation should be performed whenever the MS moves between "location areas" (identified by location area identifiers, LAIs). However, it is known that location areas often extend over several square kilometres, and a subscriber's movements are typically confined within one or two location areas [GHB08, BDE09]. So location areas may be too large to trigger TMSI reallocations in practice. We report on three different scenarios showing that some of the actual implementations of the strategy for changing pseudonyms to avoid tracking are not offering enough privacy guarantees to the mobile telephony subscribers. Our observations and their consequences on users' privacy are discussed in this section[3].

The TMSI reallocation procedure is rarely executed. Although in the standard the privacy offered to mobile phone bearers is based on frequent updates of TMSIs, our experiments show that the same TMSI can be allocated for several hours and even days. Moreover, turning on and off the MS does not usually result in a new TMSI being allocated. As an example Figure 4.6 shows that a TMSI allocated on 22/03/2012 had not been updated by 25/03/2012, making the phone trackable for a period of 3 days. This behaviour can be observed for the major UK, Greek, French and Italian network operators. An attacker could take advantage of the long life of a TMSI and monitor a few sub-areas using short range devices in order to obtain a fine grained tracking of his victim within a same LAI.

[3]The traces that allowed us to draw the conclusions presented are made available for inspection [tra]

We observed that the major UK network operators and the Vodafone and TIM Italian operators rarely execute the TMSI reallocation even in presence of MS activity, but the first message sent by a MS when requesting or receiving a service contains its TMSI, hence exposes it to eavesdropping third parties. As mentioned in Section 3.4, TMSI liveness makes it possible to locate mobile telephony users without alerting them. The attack consists in paging the victim and hence provoking a paging response. To reduce the set of answering TMSIs to the victim's one, the attacker must repeat the process several times because more than one MS could be sending a paging response at the same time. This is possible only if the TMSI is not reallocated even in case of activity exposing the TMSI (*e.g.* receiving calls). The attack in [KKHK12] thus relies on the low frequency of TMSI reallocations and demonstrates that changing pseudonyms, as mechanism to provide location privacy, is not effective without a policy for changing of pseudonyms which takes into account the actual exposure of the pseudonym caused by the mobile station activity.

A change of location area does not imply a change of TMSI although such a change is mandated by the 3GPP standard. We observed this behaviour when capturing the signalling messages of a mobile station moving by coach between different cities in the UK, using the Orange and the O2 networks where we observed the same pseudonym being accepted in different location areas with no further execution of the TMSI reallocation procedure. Assuming an average speed of 70Km/h we observed that a new TMSI was assigned after about 45 min (about 53km) and a second one after about 60 min (about 70km) while we observed a change of LAI every 5 min on average and hence a new TMSI should have been allocated, on average, about every 3km. Figure 4.7 shows an example trace where a TMSI used at location 234/33/1381 (packet no. 668) is accepted at a different location 234/33/29 (packet no.678).

The fact that a TMSI was accepted in two neighbouring LAIs contradicts the specification that a TMSI reallocation should be performed at least at each change of

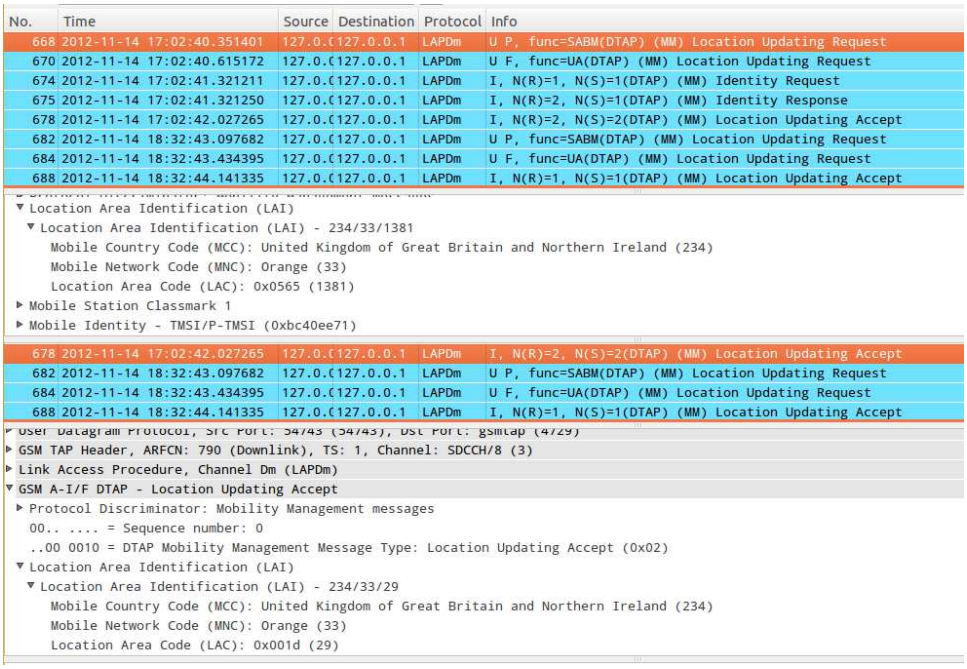

FIGURE 4.7: Trace of a UK Orange SIM card. The TMSI used at location 234/33/1381 (packet no. 668) is accepted at location 234/33/29 (packet no.678), while the 3GPP standard mandates a TMSI reallocation at each change of location.

location. However, changing pseudonym when changing location area would provide location-dependent privacy to the user since it would prevent passive tracking across different LAIs. The combination of the two behaviours reported so far (i.e. keeping the same TMSI for a long period of time and not changing it when changing location area) enables the attacker to both track his victim within an area and follow him across different areas without doing any extra effort other than passively sniffing.

Previously established keys are restored and used to encrypt the TMSI reallocation procedure. Our captures confirm that the reuse of previously established keys is a policy adopted by real networks and that in particular is used for the execution of the TMSI reallocation procedure. The experiments we performed suggest that major UK and Italian network operators[4] reuse previously established keys instead of performing the authentication procedure before each

[4]UK: Vodafone and T-mobile; Italy: Vodafone.

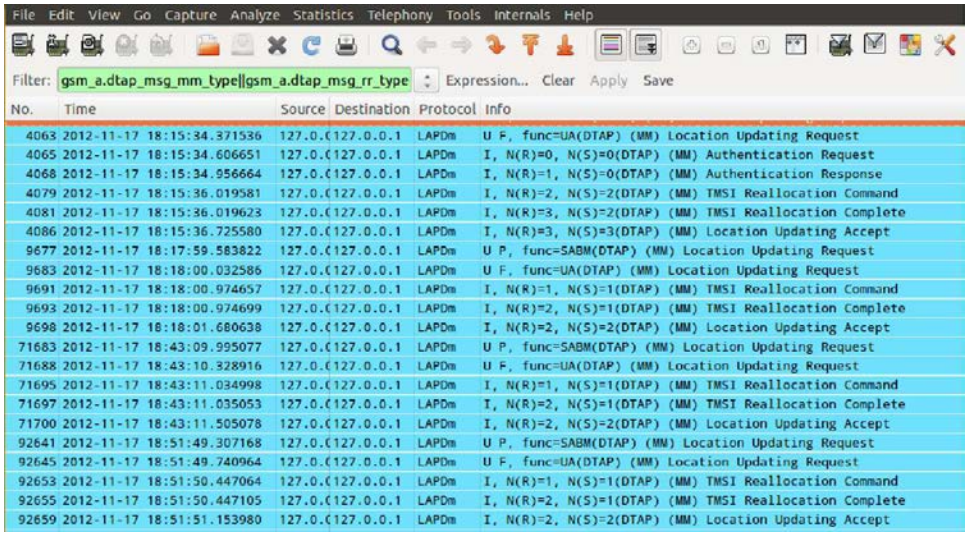

FIGURE 4.8: Trace of a UK Lebara SIM card attached to the Vodafone network while travelling on a train. The TMSI reallocation procedure is executed by reusing a previously established key. The MS first performs a location update (packet no. 4063), then the authentication procedure to establish a ciphering key (packets 4065, 4068), followed by the TMSI reallocation procedure (packets 4079, 4081). The following three TMSI reallocations (packets 9691, 9693, 71695, 71697, 92653, 92655) are executed without first performing the authentication procedure and hence reusing the previously established ciphering key.

execution of the TMSI reallocation procedure. Figure 4.8 shows a trace from a UK Lebara SIM card attached to the Vodafone network performing a location update (packet no. 4063). Then the execution of the authentication procedure establishes a new ciphering key (packets 4065, 4068) and consecutively the TMSI reallocation procedure (packets 4079, 4081) is executed. The subsequent TMSI reallocations (packets 9691, 9693, 71695, 71697, 92653, 92655) are executed without previously performing the authentication procedure and hence we can deduce that they are reusing the previously established ciphering key.

The reuse of a previously established ciphering key enables a replay attack. We describe this attack in Section 5.1.1.

4.5 Discussion

The experiments we conducted show how the adoption of pseudonyms is not a sufficient condition to ensure the privacy of mobile telephony users and that real network implementations leave plenty of room for tracking attacks. We suggest network operators should adopt activity related policies in order to prevent active tracking attacks. In general, the execution of the TMSI reallocation procedure should be more frequent even when the MS is in idle state, so to prevent mere passive tracking.

Using pseudonyms is a good mechanism to ensure the user's privacy, provided that there is enough possibility of mixing within the network, which is usually the case in mobile telecommunication networks. However, the efficiency of the pseudonym change strategy depends on many factors which the 3GPP standard leaves as implementation choices.

We showed that the implementation choices made by real network operators do not provide a satisfactory level of privacy and leave space for different kinds of tracking attacks. Moreover, we showed that the loose standard specification can produce implementations of the TMSI reallocation procedure which are subject to a linkability attack.

Our experiments were conducted on the GSM network. However, the TMSI reallocation procedure is adopted in 3G+ networks as well. Further work and specialized equipment would be needed so to investigate the usage scenario of TMSIs and TMSI reallocation in 3G+ networks.

5

Analysis of Mobile Systems' Protocols

In the previous Chapter we showed how the paging procedure can be exploited to perform a simple linkability attack and we presented an experimental analysis of the identity management thanks to which we exposed the presence, in real mobile networks, of critical behaviour from a privacy point of view. Most of these privacy critical scenarios lead straightforwardly to breaches of the mobile telephony users' privacy. In this Chapter, we will show two different attacks which are more subtle. The attacks we found are both replay attacks and both allow to track the presence of a user in a monitored area or across a set of monitored areas. The first attack concerns the 3G authentication and key agreement protocol. The second attack is a replay attack on the TMSI reallocation procedure which is enabled by the fact that the reuse of previously established keys is allowed by the standard and, as we showed in Section 4.4.2.2, is an adopted policy in practice as well. We close this Chapter explaining how we checked the feasibility of the 3G AKA attack and of the IMSI paging attack (Section 4.4.1) in practice on real 3G networks using a hacked femtocell.

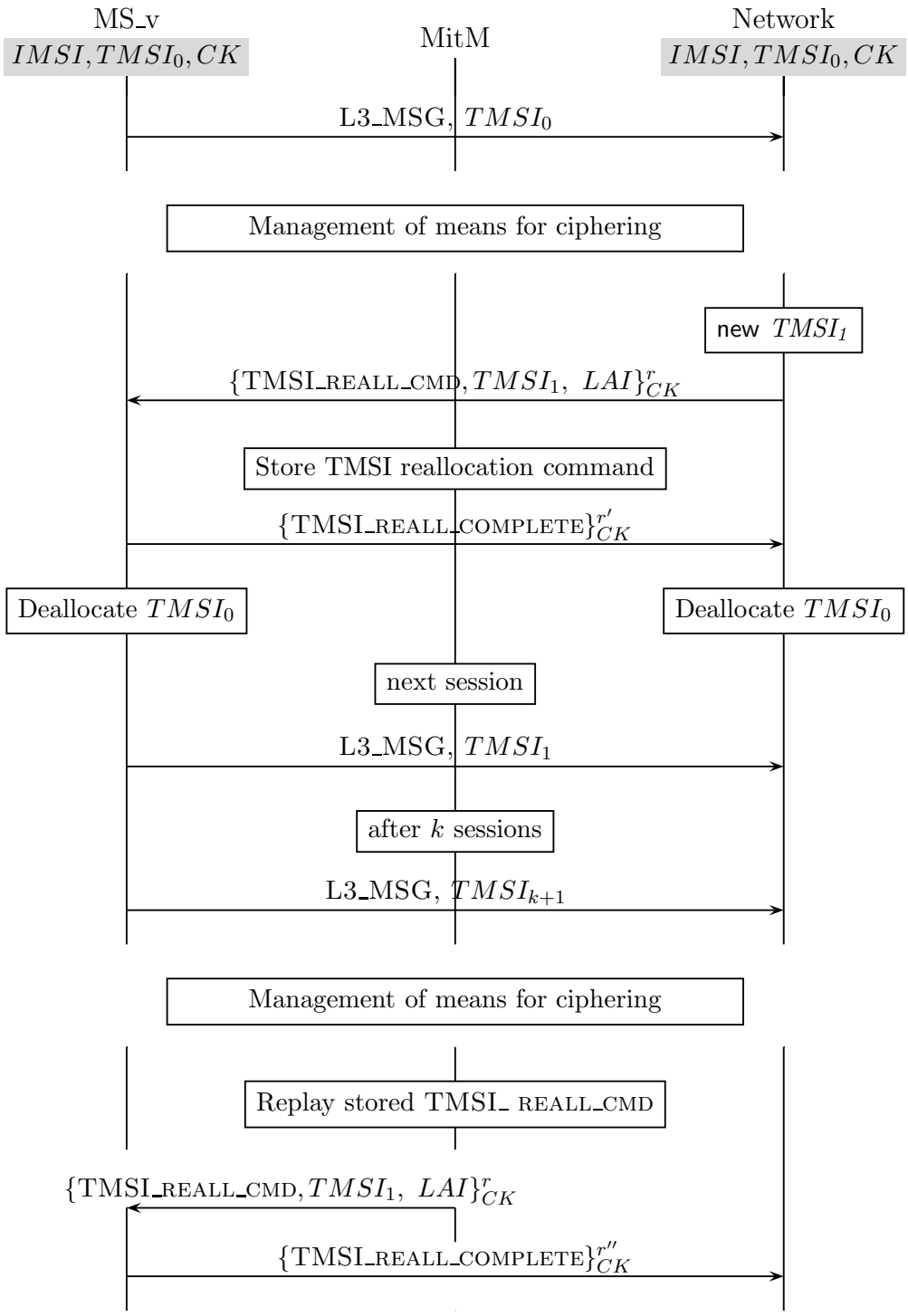

FIGURE 5.1: TMSI Reallocation Procedure Attack: the attacker captures a legitimate TMSI reallocation command message and replays it. If the MS reply to the replayed message the attacker knows that it is indeed the victim MS.

5.1 Pseudonymity Issues in Mobile Telephony Systems

In Chapter 4 we analysed the TMSI reallocation procedure from an *experimental* point of view. In this section we demonstrate a replay attack on this procedure which allows a third party to violate a user's privacy in spite of the reallocation protocol. This attack is enabled by the 3GPP standard policy allowing to restore previously established keys. As shown in Chapter 4 this policy is adopted by real mobile telephony operators.

In Chapter 6 we will formally analyse a simple fix of the TMSI reallocation procedure. In particular, we will prove that the TMSI reallocation procedure provides unlinkability in case a new ciphering session key is established before each execution of the TMSI reallocation procedure and we will discuss other possible countermeasures.

5.1.1 TMSI Reallocation Replay Attack

The reuse of a previously established ciphering key enables replay attacks such as the one depicted in Figure 5.1. An attacker, controlling a radio device able to sniff and inject messages over-the-air, first captures a TMSI reallocation command (the second message in Figure 5.1). Later on, when the MS has possibly already changed its pseudonym but not yet established new keys, the attacker can replay the captured TMSI reallocation command (one message before last in Figure 5.1). Since reuse of the session key CK is allowed, the victim's MS successfully decrypts the reallocation message and sends the TMSI reallocation complete message. This allows the attacker to distinguish the victim's MS from any other even though in the meantime a different TMSI ($TMSI_{k+1}$ in Figure 5.1) was assigned to the

victim's MS. Indeed, any other MS would not successfully decrypt the message and thus would not send any reply.

5.2 User linkability issues in 3G systems

5.2.1 3G Authentication and Key Agreement Protocol

The Authentication and Key Agreement (AKA) protocol achieves mutual authentication between a MS and the network[1], and establishes shared keys to be used to secure the subsequent communications. The keys are not exchanged during the protocol but computed locally by the MS and the SN. According to [3GP10d], the authentication procedure is always initiated by the SN to the purpose of:

- Checking whether the identity provided by the MS is acceptable or not.
- Providing parameters enabling the MS to calculate a new UMTS ciphering key.
- Providing parameters enabling the MS to calculate a new UMTS integrity key.
- Allowing the MS to authenticate the home network.

Each MS with identity *IMSI* and the network share a different secret long-term key, K_{IMSI}, assigned to the subscriber by the mobile operator and stored in the USIM. The secret key allows the MS and the network to compute shared ciphering and integrity session keys to be used for encryption and integrity check of communications.

[1]The AKA protocol is executed after a dedicated channel has been assigned to the MS.

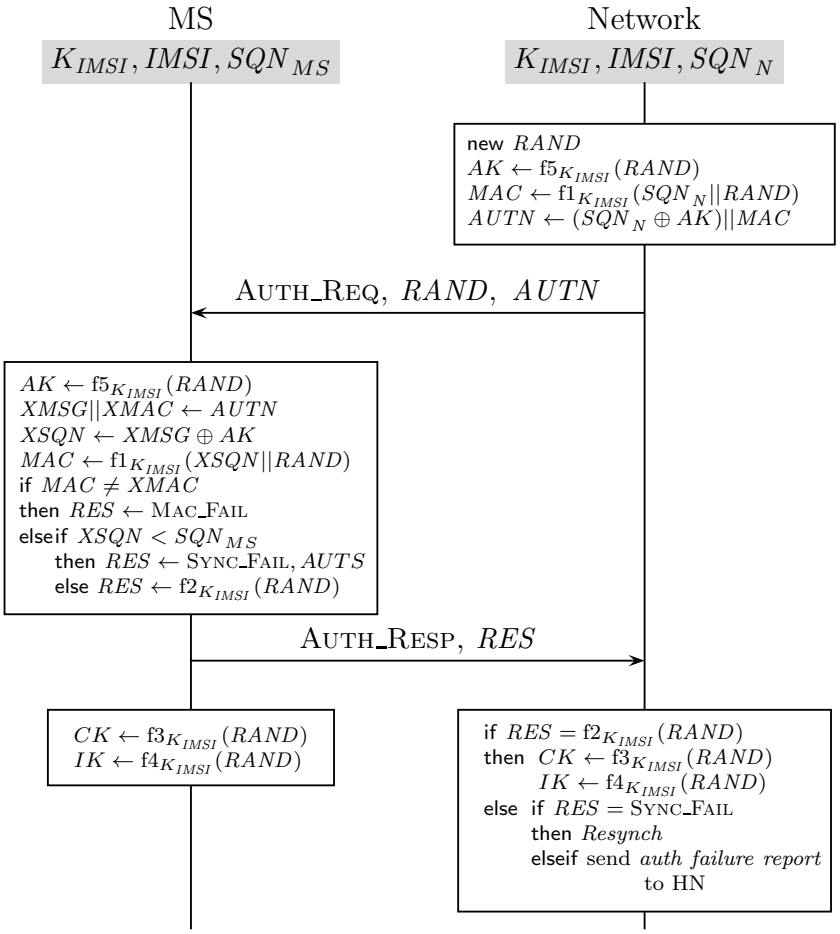

FIGURE 5.2: 3G Authentication and Key Agreement (AKA): the authentication procedure is always initiated by the network and is executed on a dedicated channel.

A successful authentication procedure establishes a so-called *security context* which identifies the set of keys to be used for secrecy and integrity purposes; a Key Set Identifier (KSI) is used to retrieve the established security context through different sessions. Once a security context is established it is considered valid until the authentication procedure is next executed or the deletion of KSI is requested by the network.

The 3G AKA protocol [3GP10d], depicted in Figure 5.2, consists of the exchange of two messages: the authentication request and the authentication response. Before

sending an authentication request to the mobile station, the network computes the authentication data: a fresh random challenge $RAND$, the authentication token $AUTN$, the expected authentication response $\mathrm{f2}_{K_{IMSI}}(RAND)$, the integrity key IK, and the encryption key CK (see Figure 5.2). The functions f1, f2, f3, f4 and f5, used to compute the authentication parameters, are keyed cryptographic functions computed using the shared key K_{IMSI} [3GP11c]. The authentication function f1 is used to calculate the message authentication code MAC; f2 is used to produce the authentication response parameter RES; the key generation functions, f3, f4 and f5 are used to generate the ciphering key CK, the integrity key IK and the anonymity key AK, respectively.

The network always initiates the protocol by sending the authentication challenge $RAND$ and the authentication token $AUTN$ to the mobile station. $AUTN$ contains a MAC of the concatenation of the random number with a sequence number SQN_N generated by the network using an individual counter for each subscriber. A new sequence number is generated either by increment of the counter or through time based algorithms as defined in [3GP10d]. The sequence number SQN_N allows the mobile station to verify the freshness of the authentication request to defend against replay attacks (see Figure 5.2).

The MS receives the authentication request, retrieves the sequence number SQN_N and then verifies the MAC (condition $MAC = XMAC$ in Figure 5.2). This step ensures that the MAC was generated by the network using the shared key K_{IMSI}, and thus that the authentication request was intended for the mobile station with identity $IMSI$. The mobile station stores the greatest sequence number used for authentication, so far SQN_{MS}. This value is used to check the freshness of the authentication request (condition $XSQN < SQN_{MS}$ in Figure 5.2) to avoid replay attacks.

The mobile station computes the ciphering key CK, the integrity key IK and the authentication response RES and sends this response to the network.The network authenticates the mobile station by verifying whether the received response is equal to the expected one ($RES = \mathrm{f2}_{K_{IMSI}}(RAND)$). The authentication procedure can fail on the MS side either because the MAC verification failed, or because the received sequence number $XSQN$, is not in the correct range with respect to the sequence number SQN_{MS} stored in the mobile station. In the former case, the mobile station sends an authentication failure message indicating MAC failure (MAC_FAIL) as the failure cause. In the latter case, the authentication failure message indicates synchronisation failure (SYNC_FAIL) as the failure cause and the AUTS parameter is sent to allow the network to resynchronize.

After successful authentication, the SN sends a security mode command message to the MS, indicating which one of the allowed algorithms to use for ciphering and integrity checking of the following communications. If the network receives a synchronisation failure then it will perform resynchronisation by sending an *Authentication Data Request* with a *synchronisation failure indication* to the HN/Auc in order to retrieve valid authentication vectors. This request contain the $RAND$ used during the AKA and the $AUTS$ parameter sent by the MS in the authentication failure message. If the network receives a MAC failure or $RES \neq \mathrm{f2}_{K_{IMSI}}(RAND)$ then it sends an *Authentication Failure Report* to the HN indicating among other parameters $IMSI$, failure cause and $RAND$. The network may trigger the identification procedure and re-attempt authentication.

5.2.2 3G AKA Protocol Linkability Attack

To detect the presence of a victim mobile station MS_v, in one of his monitored areas, an active attacker just needs to have previously intercepted one legitimate authentication request message containing the pair $(RAND, AUTN)$ sent by the

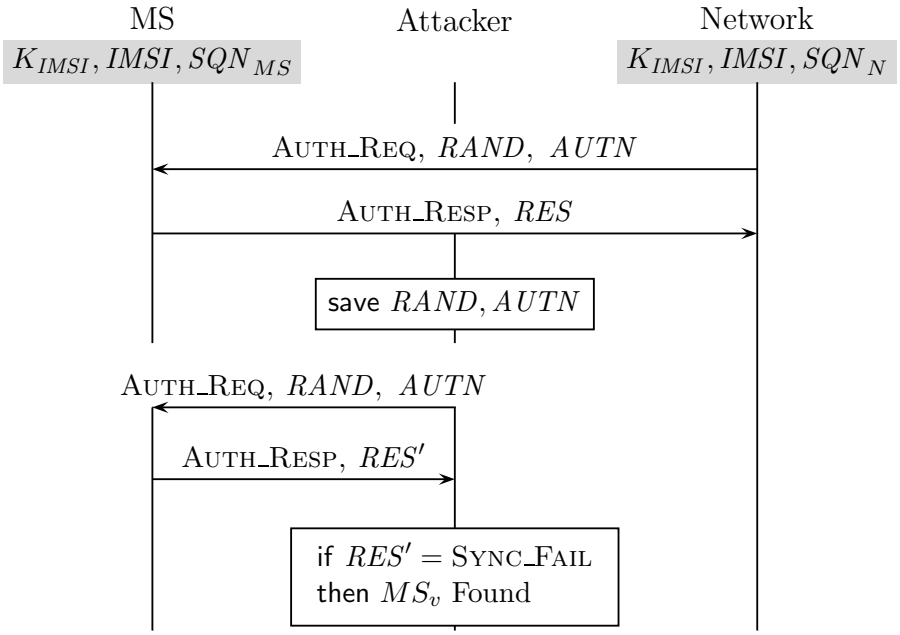

FIGURE 5.3: AKA Protocol Linkability Attack

network to MS_v. The captured authentication request can now be replayed by the adversary each time he wants to check the presence of MS_v in a particular area. In fact, thanks to the error messages, the adversary can distinguish any mobile station from the one the authentication request was originally sent to. On reception of the replayed authentication challenge and authentication token $(RAND, AUTN)$, the victim mobile station MS_v successfully verifies the MAC and sends a synchronisation failure message. However, the MAC verification fails when executed by any other mobile station, and as a result a MAC failure message is sent. The implementation of few false BSs would then allow an attacker to trace the movements of a victim mobile station, resulting in a breach of the subscriber's untraceability. The proposed attack is shown in Figure 5.3. Note that this attack affects only 3G mobile systems, in fact 2G systems adopt a different authentication protocol which does not provide mutual authentication (i.e. the mobile station does not authenticate the network) and does not involve error and recovery procedures in case the authentication of the network fails. Thus, the distinguishing attack on the error messages cannot be performed in 2G networks.

5.3 Implementation of some 3G Protocols Attacks

In order to test the attacks presented in Chapter 4 and Section 5.2.2 in a deployed telecommunication network, we used a commercially available femtocell. Although, the particular femtocell hardware is tied to the network operator SFR, the proposed attacks are not. Indeed, we tested the attacks using mobile phones registered to different operators, hence just using SFR as serving network. The authentication token $AUTN$ is still provided by the victim's Home network. So by testing our attacks on T-Mobile, O2, SFR, and Vodafone victim MSs, we establish that all these tested networks are vulnerable to the attacks described above. However, we want to stress here that our implementation has the only purpose of showing the feasibility of our attacks and confirm that real cellular networks follow the 3GPP standard specifications and thus are vulnerable to the proposed attacks. The same attacks could be mounted by appropriately programming a USRP [Ett09], which is a hardware device able to emit and receive radio signals. In this case, one could obtain wider range attack devices in order to monitor larger areas.

5.3.1 Femtocell architecture

A femtocell is a device that acts as a small base station to enhance 3G coverage and connectivity, especially inside buildings with otherwise bad coverage. Its coverage radius ranges from 10 to 50 meters. It connects mobile phones to the network of the corresponding MNO (Mobile Network Operator) using an existing wired Internet connection provided by the femtocell user, not the operator. 3G femtocells, also called Home Node B (HNB) support most of the functionalities provided by a typical 3G base station (Node B), *e.g.* physical layer (radio signalling) functions.

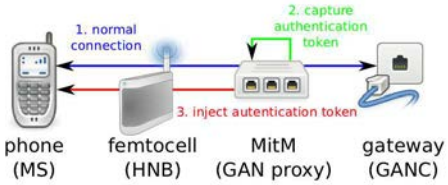

FIGURE 5.4: Experimental Attack Setup

In addition, the HNB establishes an authenticated secure tunnel over the Internet with the network of the operator. Using this encrypted connection, the femtocell forwards all radio signalling and user-generated traffic to the GANC (GAN Controller), which is connected to the core network of the operator (refer to [3GP11a] for more details of the femtocell architecture).

The communication between the femtocell and the GANC is based on the Generic Access Network (GAN) protocol. The GAN protocol, was originally designed to allow mobile communication over Wi-Fi access points. The protocol was standardised by MNOs in 2004 [Kin10] and led to the GAN specification [3GP10a, 3GP10b] in 2005. This specification has been adopted and extended to be used in femtocell environments [ZdlR09] . The femtocell uses this protocol to forward communication from a mobile station via the GANC to the network or vice versa. The MS does not need any special GAN support, it just connects to the femtocell in the same way as it connects to a standard base station. The femtocell maps all Layer-3 radio signalling to TCP/IP based GAN messages and passes them to the GANC. Thus, it transparently encapsulates all traffic generated by the phone and the network.

5.3.2 Attack Procedure

For the purpose of implementing our attacks (Chapter 4 and Section 5.2.2), we use a compromised femtocell like the one described in [GRB12]. More specifically, we reproduce the hacking performed in [GRB12] to gain root access of our femtocell

and redirect the traffic to a Man in the Middle (MitM) GAN proxy, positioned between the femtocell and the GANC. We use this MitM GAN proxy as entry point for message injection. In particular, using the MitM GAN proxy we can inject messages into the connection between the MNO and the femtocell. The femtocell forwards these messages to the mobile phone, making them appear as legitimately delivered by the MNO. To perform the attacks, we intercept, modify and inject 3G Layer-3 messages into the communication from the base station to the mobile phone in both directions, GANC-to-femtocell and femtocell-to-GANC. We redirect all the traffic between the femtocell and the GANC to our GAN proxy.

The GAN traffic is cleartext travelling over an IP Sec tunnel for which we own the key material, thanks to the initial rooting/hacking of the femtocell. Additionally, we developed a set of applications which allow us to intercept, manipulate or insert selected messages, and distinguish different types of GAN messages. This allows us, for example, to cache subscribers information used to perform the attacks. In particular, we store the random challenge $RAND$, the authentication token $AUTN$, the TMSI and the IMSI of our victim MS. This information is directly extracted from the traffic that is passed through the MitM GAN proxy.

In the next two paragraphs we give some details about the implementation of our attacks on the IMSI paging procedure and on the AKA protocol.

IMSI-Paging Procedure Attack To perform our IMSI paging attack, our software crafts a paging message encoding the necessary paging headers and parameters and a mobile station identity, *i.e.* one of the previously stored victim IMSIs. The crafted paging request is then sent by the GAN proxy to the femtocell. When the victim mobile phone receives the IMSI paging request, it readily answers with a paging response containing the victim's TMSI. Thus, by injecting a paging request, we can check whether a phone belonging to a designated victim is in the area covered by our device. In case of success, the phone generates the

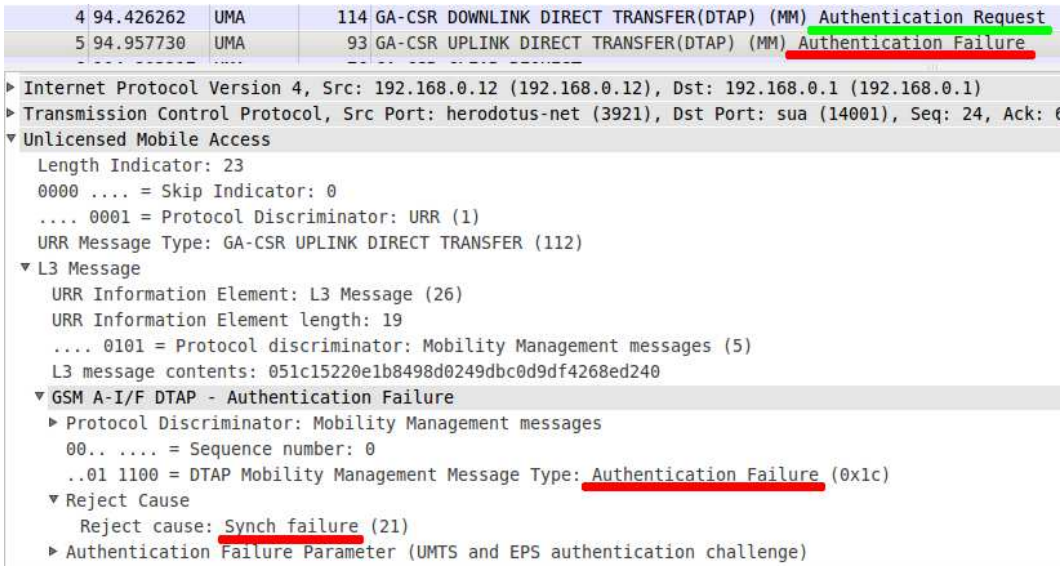

FIGURE 5.5: Linkability-Attack: Victim Found.

paging response, while a failed attempt generates no message. In general, it is possible that more than one phone replies to a paging request during the same time slot. However, one can repeat this procedure multiple times and correlate the timing and TMSI usage from the multiple replies as in [KKHK12].

AKA Protocol attack To perform our AKA attack we replay a given authentication message for a specific target for which the GAN proxy cached the legitimate authentication data, *i.e.* $RAND$, $AUTN$. This data is sent unencrypted on the radio link and could be captured with any equipment capable of sniffing it. As soon as a dedicated channel is allocated to the MS, *e.g.* after being paged or when initiating a phone call, our software crafts an authentication request AUTH_REQ using the previously cached $RAND$ and $AUTN$, *i.e.* replays a previous request. This request is encapsulated into a GAN message and sent to the femtocell. The femtocell takes care of delivering the authentication request message on the dedicated channel assigned to the MS, as illustrated in Figure 5.4. The phone performs a validation of the authentication request and answers with the authentication response. If the response to the replayed authentication is a Synchronisation Failure

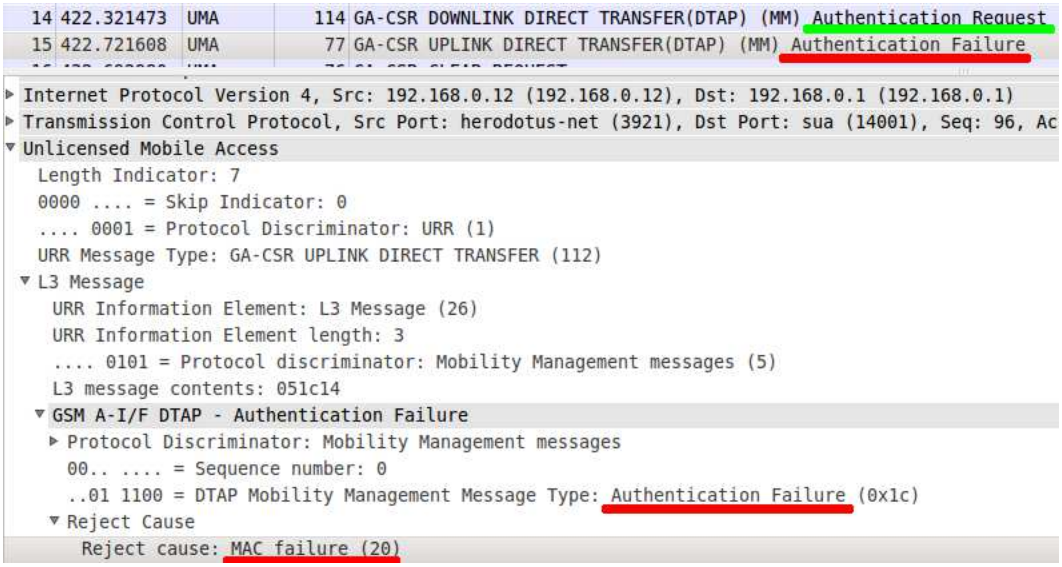

FIGURE 5.6: Linkability-Attack: Victim not Found.

(Figure 5.5), then the MS on this dedicated channel is the victim's phone, and the victim is indeed in the femtocell area. Otherwise, if the response to the replayed authentication is a MAC failure (Figure 5.6), the attacker can only deduce that this phone is not the victim phone. The attacker then needs to replay the same authentication message to possibly all the mobile stations in the monitored area. If the attacker receives only MAC failures then he can deduce that the victim is not in the monitored area.

The 3G AKA protocol is performed at each new session in the femtocell setting, this makes the caching of the authentication parameters very easy. Though, we do not have the tools to test if this applies when connecting to a typical Node B, we tested the 3G/GSM interoperability scenario by using the Osmocom-BB software and we observed that in this setting the execution of the AKA protocol can be triggered by calling for example the victim mobile phone a given number of times (by hanging up within a short time window this activity can be made non detectable by the victim [KKHK12]). For instance, our experiments showed that the execution of the AKA protocol on the UK Vodafone network can be triggered by calling six times the victim mobile phone, and hanging up before it even rings.

To illustrate the use of our attacks, consider an employer interested in tracking one of his employee's accesses to a building. He would first use the femtocell to sniff a valid authentication request. This could happen in a different area than the monitored one. Then the employer would position the device near the entrance of the building. Movements inside the building could be tracked as well by placing additional devices to cover different areas of the building. Similarly, these attacks could be used to collect large amount of data on users' movements in defined areas for profiling purposes, as an example of how mobile systems have already been exploited in this direction is available in [pat]. If devices with wider area coverage than a femtocell are used, the adversary should use triangulation to obtain finer position data.

6

Privacy Friendly Fixes

In Chapters 3, 4, 5 we have seen some breaches to the 2G/3G user's privacy. In this Chapter we will propose some privacy friendly solutions and in Chapter 7 we will show how it is possible to verify them using the `ProVerif` automatic protocol verifier. We will highlight some of the limitations of `ProVerif` which make it not possible to verify the TMSI reallocation procedure and its fix. Further, we will provide the sketch proof of the unlinkability of the fixed version of the TMSI reallocation procedure. In Chapter 8 we will present our work on an extension of the ProVerif tool in order to deal with stateful processes and observational equivalence based properties.

Despite the use of temporary identities to avoid linkability and to ensure anonymity of 3G subscribers, active attackers can rely on the paging procedure to break both anonymity and unlinkability. Moreover, the AKA protocol provides a way to trace 3G subscribers without the need to identify them in any way. As described in Chapter 5, these two attacks on privacy can be implemented using cheap devices which are widely available. This shows that the analysed procedures are a real threat for the users' privacy, and countermeasures should be promptly taken to provide an effectively privacy friendly mobile telephony system.

In this section we propose a set of countermeasures involving symmetric and public key cryptography. The public key infrastructure we propose is easy to deploy because we only require one public/private key pair per mobile network operator, and none for the mobile stations. More generally, the solutions we present require only small changes to the current security architecture and to the cryptographic functions currently used in 3G. Hence we believe our solutions may be implemented in a cost-effective way, and thus could realistically be adopted by the telecommunication operators.

In addition to the solutions proposed to fix the IMSI paging and the AKA protocol, in this section we give a privacy friendly version of the identification procedure to fix the IMSI catcher attack. Indeed, the problem of privacy is a multilayer/-multiprotocol problem [AO05] which requires all protocols at all layers to satisfy the desired properties. Even though, the analysis from the user privacy point of view of the entire set of 3G protocols cannot be tackled in a single thesis, we cannot ignore the best known privacy issue of mobile telecommunication systems. For this reason, we include a fixed version of the identification procedure in our privacy friendly solutions.

6.1 Public Key Infrastructure

We propose the adoption of a public key infrastructure (PKI) providing each MNO with a private/public key pair. The public key of a network provider can be stored in the USIM. This public key makes it possible for a mobile station to encrypt privacy related information such as the IMSI, and deliver them to the network in a confidential manner. We do not require a public/private key pair to be assigned to the mobile stations. The adoption of a PKI can also solve the problem exposed by Zhang and Fang in [ZF05] concerning the lack of serving network authentication in the current 3G infrastructure.

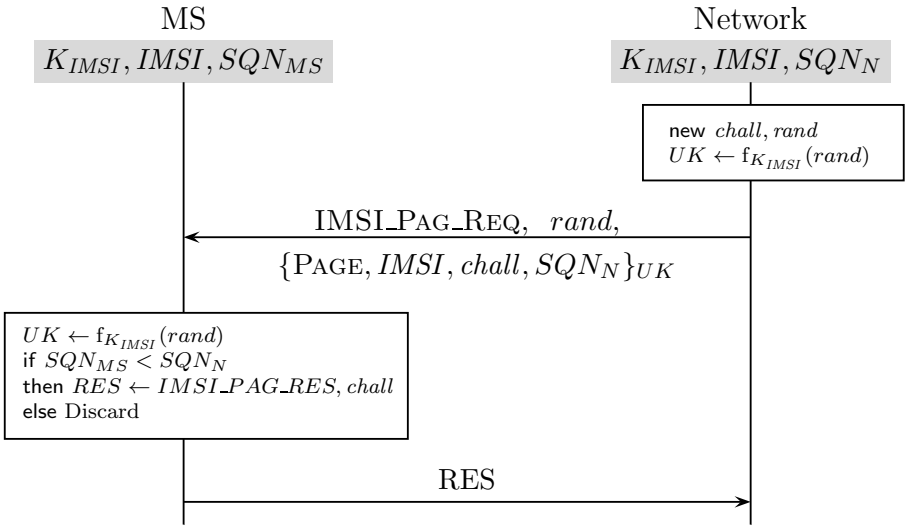

FIGURE 6.1: IMSI Paging Procedure Fix. The IMSI paging request which is sent on a common channel is encrypted with the unlinkability key UK so to ide the long term identity $IMSI$. Each MS decrypts the IMSI paging request and check if the $IMSI$ contained in it is its own. If so the MS checks the freshness of the freshness of the request ($SQN_{MS} < SQN_N$), requests a dedicated channel and sends the response RES to the network. Otherwise the MS discards the message and aborts the procedure.

Protecting the IMSI Paging Procedure To protect the paging procedure, we propose to encrypt the paging request using a shared session key UK_{IMSI}, which we call unlinkability key. This key is generated by applying a new one-way keyed function f to the long-term key K_{IMSI}, and a random number *rand* contained in the paging request. This key should be used for privacy preserving purposes only. Furthermore, we require the encrypted request message to include a random challenge *chall* and a sequence number SQN. The network stores the random challenge and checks it against the one sent by the MS in the paging response (Figure 6.1). The aim of the SQN is to ensure freshness of the paging request and avoid replay attacks. The SQN should be handled in the same way as in the AKA protocol. A MS receiving a legitimate IMSI paging request should discard it if the SQN is not in the correct range. The use of this procedure should still be kept minimal (preferring the paging with TMSI whenever possible) to avoid burdening the signalling communication with cryptographic operations. In fact,

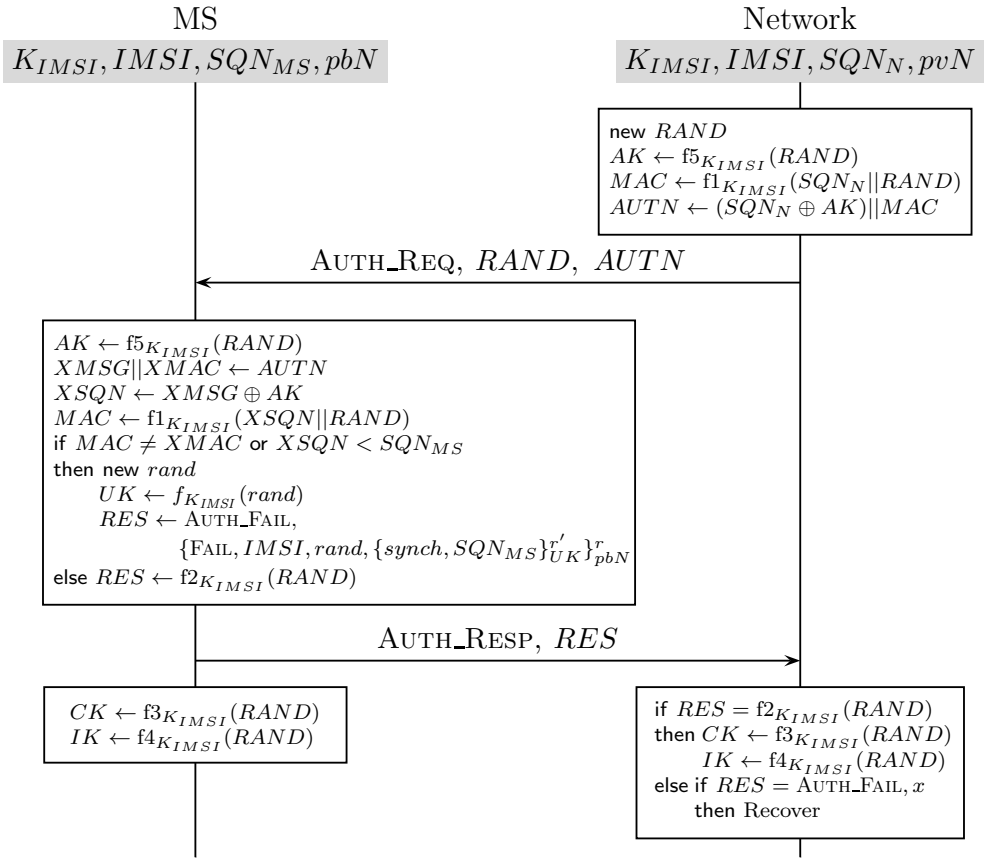

FIGURE 6.2: The fixed AKA protocol. The error messages are encrypted using the network public key.

each MS has to decrypt and check all the received encrypted IMSI paging to determine if it is the recipient (Note that TMSI paging are still sent in cleartext). To enable the IMSI paging by a serving network, the encrypted paging request may be precomputed by the home network and IMSI paging vectors containing the random, the encrypted paging and the challenge could be sent in bulk to the serving network the MS is attached to, in this way the unlinkability key UK_{IMSI} only has to be shared between the MS and the home network.

Fixing the AKA Protocol The AKA protocol is a threat for the unlinkability of 3G subscribers because the error messages sent in case of authentication

failure leak information about the identity of the subscriber. To avoid this information leakage, the error messages sent in case of any type of failure should look indistinguishable from an attacker's point of view.

Moreover, the 3G standard stipulates [3GP10d] different procedures to recover from each of the two kinds of failure, but this is a source of additional information flow that can be used to launch our privacy attack. In the solution we propose we solve this problem since error recovery can be performed within the network without the need to trigger further procedures over-the-air. Indeed, all the parameters needed for error recovery are sent in the error message allowing the recovery procedure to be carried within the network.

The fixed version of the AKA protocol (Figure 6.2) carries on as specified by the standard. The network sends $RAND$, $AUTN$ and waits for a response. The response is $RES = \mathrm{f2}_{K_{IMSI}}(RAND)$, as in the standard, in case the checks of MAC and sequence number are successful. If either of these checks fails, an error message is sent to the network. The failure message is now encrypted with the public key of the network pbN, and contains a constant FAIL, the IMSI, and the current sequence number SQN_{MS} of the MS. The IMSI sent encrypted in the error message allows the network to check the identity of the MS without triggering the identification procedure. The current sequence number of the mobile station enables the network to perform resynchronisation with the Authentication Centre (AuC, the server storing subscribers authentication data) of the operator of the mobile station, if needed. SQN_{MS} is sent encrypted with the unlinkability key (as defined in the fixed paging procedure) in order to authenticate the error message to the Network as coming from the MS with permanent identity $IMSI$. The Network can deduce the cause of the failure from the $IMSI$ and SQN_{MS} contained in the error message. Upon receipt of this authentication failure message the action performed for error recovery purposes should be the same regardless of the type of failure occurred. Indeed any difference in behaviour would be a source of additional information flows.

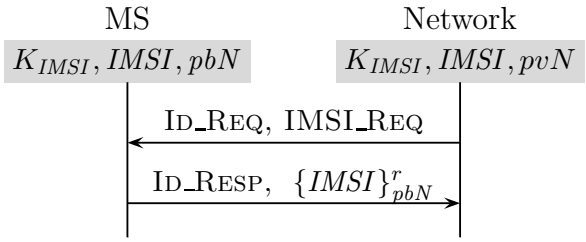

FIGURE 6.3: Identification Procedure Fix. The identity response is encrypted with the public key of the network. The r denotes randomised encryption.

Protecting the Identification Procedure The identification procedure exposes the IMSI of a MS (the IMSI is sent in cleartext upon request by the network). Hence, it breaches both anonymity and unlinkability. According to the standard, the use of the identification procedure should be limited as much as possible, to avoid a passive attacker overhearing the IMSI. However, as discussed in Chapter 3 the cost of devices allowing active attacks is constantly decreasing. As a consequence, enhancing the protocol to protect the IMSI is vital to ensure privacy.

The fixed version of the identification procedure (Figure 6.3) involves two messages: the first is sent by the network to ask for the IMSI, the second, the identity response, is the randomised encryption of the IMSI of the mobile station using the public counterpart (pbN) of the private key of the network operator (pvN).

Fixing the TMSI reallocation procedure The solution we propose to fix the TMSI reallocation procedure does not require any change in the security architecture of mobile telephony systems. We only require the standard to specify that the reuse of the encryption key is not permitted when the key is used to execute the TMSI reallocation procedure, i.e. the establishment of a new key before the execution of the TMSI reallocation procedure should be mandatory. This would avoid the possibility of replay attacks to be mounted. However, frequent executions of the authentication procedure could burden the radio communication and slow down the delivery of mobile telephony services. Alternative solutions

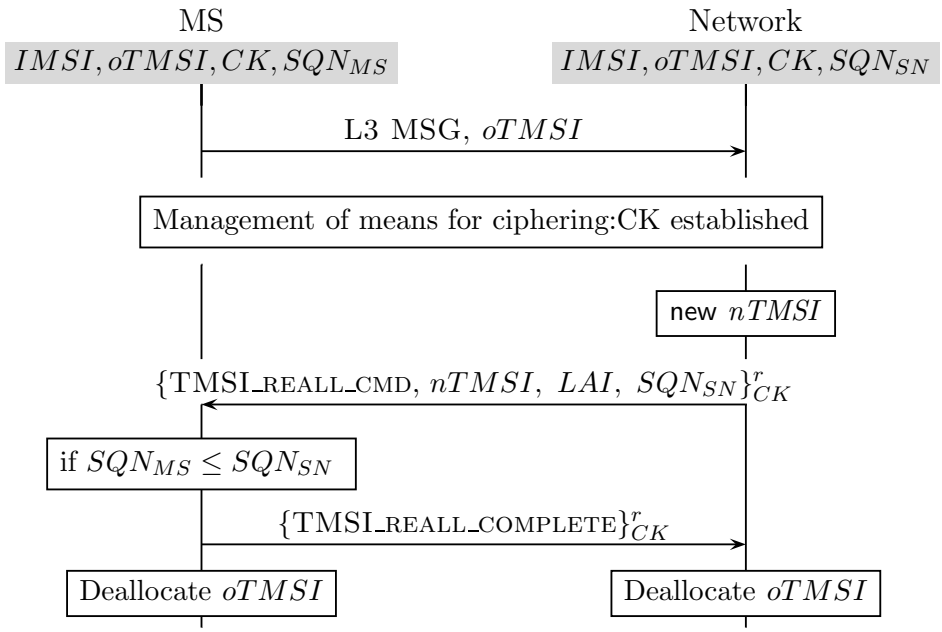

FIGURE 6.4: TMSI Reallocation Procedure Fix: this fix uses the SQN to ensure the freshness of the reallocation command.

are possible, as for example the introduction of a sequence number in the TMSI reallocation command, similarly to the one used to avoid replay attacks against the Authentication and Key Agreement protocol [3GP10d]. We illustrate this solution in Figure 6.4. The network sends a sequence number SQN_{SN} along with the TMSI reallocation command. The MS checks if the received sequence number is in the expected range ($SQN_{MS} \leq SQN_{SN}$). If so it carries on with the reallocation of the TMSI. Otherwise the MS aborts the TMSI reallocation execution, hence avoiding replay attacks.

6.2 Discussion of the Proposed Fixes

While the fix we propose for the identification procedure and the one we propose for the TMSI reallocation are intuitive and straightforward, this is not the case for the other two procedures. In particular, we take care of maintaining the style of

mobile telecommunication protocols and at the same time ensuring privacy. We introduce the unlinkability key, a new session key generated for privacy purposes, instead of using the long term key K_{IMSI} (as in the 3G AKA), and make use of the sequence number SQN for freshness purposes (this is needed to avoid user linkability caused by replay attacks). We maintain the authentication flow of the AKA and modify only the way error messages are dealt with by including error recovery information inside the error message (this avoids the triggering by the network of specific procedures in order to perform error recovery depending on the occurred error).

Our proposed fixes use public-key cryptography; intuitively, there is no way to avoid that, since if a mobile station's TMSI is unknown to the serving network (hence the need to perform the identification procedure) then there is no shared key by which they can communicate privately. The additional costs associated with deploying and using public-key cryptography are in fact small for the two following reasons.

Firstly, only mobile telephony *operators* are required to have a public/private key pair. Neither subscribers, nor mobile phone equipments nor USIMs need to have their own public/private key pair. The operator's public key could be stored in the USIM of the mobile station, as it is already the case for the IMSI and the long-term key K_{IMSI}. The Home Network can act as a certifying authority for the public key of the different Serving Networks (see below). Thus, the public key infrastructure is similar to that used on the web, where corporations (not users) have certified keys.

Secondly, the computationally expensive public-key encryption and decryption are required only for the identification protocol and when the AKA-protocol fails. The execution of the identification and the IMSI paging procedures should anyway be kept minimal according to the currently deployed standard. Moreover, failures

during the execution of the AKA-protocol rarely occur according to our experiments. Hence, the computational overhead of the public-key cryptography is not significant. Moreover, it is possible to delegate the encryption and decryption to the mobile equipment, instead of executing them on the USIM. This would not weaken the security properties of the 3G procedure, since the mobile equipment in the current architecture has already access to the IMSI, while the network public key is publicly available information.

For roaming purposes, each Home Network (HN) can act as certifying authority of the Serving Network (SN) for its own subscribers. The public key $pbHN$ of the HN could be stored in the USIM. At registration time with a SN, the MS would declare its HN, and the SN would provide the MS with its public key $pbSN$, together with a certificate from the mobile station's HN ($\mathsf{sign}_{skHN}(pbSN, idSN)$). Hence, a mobile station would only need to obtain a certified version of the SN's public key, and verify it using its own network provider public key. This would provide, in an efficient way, the MS with the necessary public keys to execute our fixed versions of the protocols.

The introduction of cryptographic operations on the mobile equipment side could be a source of Denial of Service (DoS) attacks aiming to consume the battery load of victim MSs. To mitigate the effect of such attacks, the mobile phone's software could rate limit the phone's willingness to respond to authentication, IMSI paging and identity request messages, so to guarantee a minimum battery life-time even in case of attempted DoS attacks. We have calculated that responding to such requests on average once per minute would consume an additional one tenth of battery life.

7

Formal Verification of the Fixed Protocols

Many deployed protocols have subsequently been found to be flawed [Low96b, CJS+08, ACC+08b, BCFS10]. In this perspective and in order to increase the confidence one can have in the solutions proposed in Chapter 6, we show how to formally analyse our proposed fixes *w.r.t.* privacy. In particular in this Chapter, we present the results of the automatic verification of the privacy-friendly enhancement discussed in Chapter 6. We use the formalisation of privacy-related properties as given by Arapinis et al. in [ACRR10], namely strong unlinkability and strong anonymity and adapt them to obtain definitions suitable for automatic verification using the `ProVerif` tool [Blab]. Specifically we automatically verify the unlinkability and anonymity properties of our fixes of the procedures exposing the IMSI (identification and paging) and of the 3G AKA protocol. Further, we highlight in this Chapter the reasons that make the currently available tools not adequate for the automatic verification of the TMSI reallocation procedure and present a sketch of the manual proof of the unlinkability of the fixed TMSI reallocation procedure. We report the full proof in Appendix B. In Chapter 8 we will present an extension of the theory underlying the `ProVerif` and `StatVerif`

tools in order to automatically verify the TMSI reallocation procedure and more in general to verify observational equivalence based properties of stateful protocols. However, further work is required in this direction since this first implementation is too conservative and still suffers from `ProVerif` overapproximation of fresh names.

7.1 `ProVerif` Encoding

We use the automatic verification tool `ProVerif` [Blab] which takes in input processes written in a sintax similar to the applied pi-calculus [AF01] one. In the rest of this Chapter we use the `ProVerif` sintax to describe how we modelled the fixed protocols. However, we omit the formal details [Bla09, BAF08] which are given in Chapter 8 as part of the theory of our extension of `ProVerif` for observational equivalence of stateful processes.

We here just briefly describe the `let` construct:

$$\texttt{let}\ M = D\ \texttt{in}\ P\ \texttt{else}\ Q$$

that represents a process which tries to rewrite D and matches the result with M; if this succeeds, then the variables in M are instantiated accordingly and P is executed; otherwise Q is executed.

Example 7.1. *Multiple mobile stations* `MS`*, with identity* `imsi`*, running along with the serving network,* `SN`*, with which the share a long-term symmetric key* `sk`*, can be modelled by the process:*

$$\begin{aligned} \texttt{S} \;=\; & \texttt{new pvN};\ \texttt{let pbN} = \texttt{pub(pvN)}\ \texttt{in} \\ & \texttt{out(c, pbN)}; \\ & !\texttt{new sk}; \texttt{new imsi}; !\texttt{new sqn}; (\texttt{SN}|\texttt{MS}). \end{aligned}$$

In this and in the following examples `pvN` *and* `pbN` *are network's private and public key respectively.*

The privacy related properties we verify are expressed in terms of *observational equivalence*. Intuitively, two processes P and Q are observationally equivalent denoted by $P \approx Q$, if any interaction of P with the adversary, can be matched with an interaction of Q (and vice versa, *i.e.* all interactions of Q can be matched by P) and the same input/output behaviour is observed.

The `ProVerif` tool can prove diff-equivalence, *i*.e. equivalence of a pair of processes differing by some choice of terms, but having the same structure i.e. biprocesses. The choice of terms is written $\texttt{choice}[\texttt{M}, \texttt{M}']$. For example, to test if the processes $\texttt{out}(\texttt{c}, \texttt{a})$ and $\texttt{out}(\texttt{c}, \texttt{b})$ are equivalent, one would check the following biprocess using `ProVerif`: $\texttt{out}(\texttt{c}, \texttt{choice}[\texttt{a}, \texttt{b}])$. More specifically `ProVerif`checks whether a biprocess satisfies uniformity (see definition 8.4), a property which intuitively requires the two processes component of the biprocess to evolve in parallel and exhibit the same behaviour. Uniformity is proved to imply observational equivalence (see Section 8.4 for more detail).

The encoding of the equivalence representing unlinkability is challenging since the processes to be tested do not have the same structure hence it is not straightforward to see how to build the biprocess representing them. In the next two Sections we show how we built the biprocess to test strong unlikability and the biprocess to test strong anonymity respectively.

7.1.1 Strong Unlinkability

In our mobile phone scenario, the strong unlinkability property holds when the situation where mobile stations access services multiple times looks the same as the ideal situation where each mobile station accesses the services at most once, *i.e.* where by construction unlinkability holds. Formally, we want the process `S`,

defined in Example 7.1, to be observationally equivalent to the system $\mathtt{S}_{UNLINK}$ defined as follows:

$$\begin{aligned}\mathtt{S}_{\mathtt{UNLINK}} \;=\; & \mathtt{new}\ \mathtt{pvN};\ \mathtt{let}\ \mathtt{pbN} = \mathtt{pub}(\mathtt{pvN})\ \mathtt{in}\\ & \mathtt{out}(\mathtt{c}, \mathtt{pbN});\\ & !\mathtt{new}\ \mathtt{sk}; \mathtt{new}\ \mathtt{imsi}; \mathtt{new}\ \mathtt{sqn}; (\mathtt{SN}|\mathtt{MS}).\end{aligned}$$

The absence of the replication before the `new sqn` construct means that in $\mathtt{S}_{\mathtt{UNLINK}}$ each MS executes the protocol at most once. The above mentioned observational equivalence can be verified with `ProVerif`, defining `S` and $\mathtt{S}_{\mathtt{UNLINK}}$ as the following biprocess $\mathtt{PV}_{\mathtt{UNLINK}}$, where $sk1$, $sk2$ are long term keys and $imsi1, imsi2$ are long term identities:

$$\begin{aligned}\mathtt{PV}_{UNLINK} \;=\; & \mathtt{new}\ pvN;\ \mathtt{let}\ pbN = \mathtt{pub}(pvN)\ \mathtt{in}\\ & \mathtt{out}(c, pbN);\\ & !\mathtt{new}\ sk1; \mathtt{new}\ imsi1;\\ & !\mathtt{new}\ sk2; \mathtt{new}\ imsi2; \mathtt{new}\ sqn;\\ & \mathtt{let}\ (sk, imsi)\mathtt{=choice[}(sk1, imsi1)\mathtt{,}(sk2, imsi2)\mathtt{]}\\ & \mathtt{in}\ \mathtt{(SN\ |\ MS).}\end{aligned}$$

We have that the left side of the choice represents a system where a mobile station (with identity $imsi1$ and key $sk1$) may execute the protocol many times, while the right side represents a system where mobile stations execute the protocol at most once (the identity $imsi2$ and the key $sk2$ are always different and can be used at most once for the execution of the protocol). Hence, we reduce the problem of testing strong unlinkability to the diff-equivalence of a biprocess. `ProVerif` proves that the strong unlinkability property is satisfied by our models of the fixed identification, paging and AKA protocols as described in Chapter 6 (See table 7.1).

7.1.2 Strong Anonymity

In our mobile phone scenario, strong anonymity requires a system in which a mobile station $\mathtt{MS}_V$ with publicly known identity $IMSI_V$ executes the protocol to be indistinguishable from a system in which the $\mathtt{MS}_V$ is not present at all. Such a system obviously preserves $IMSI_V$'s anonymity. Formally, we want the system S, defined as in Example 7.1 to be observationally equivalent to the system $\mathtt{S}_V$ defined as follows:

$$\begin{aligned} S_V \;=\; & \mathtt{new}\, pvN;\ \mathtt{let}\ pbN = \mathtt{pub}(pvN)\ \mathtt{in} \\ & \mathtt{out}(c, pbN); \\ & \quad !\mathtt{new}\ sk; \mathtt{new}\ imsi; (!\mathtt{new}\ sqn; \mathtt{(SN\ |\ MS))} \\ & \mid \mathtt{new}\ sk; !\mathtt{new}\ sqn;\ \mathtt{(SN\ |\ MS}_V\mathtt{)}. \end{aligned}$$

In the system $\mathtt{S}_V$ the mobile station $\mathtt{MS}_V$ with publicly known identity $imsi_V$ can run the protocol. The mentioned observational equivalence can be translated in the following ProVerif biprocess $\mathtt{PV}_{ANON}$, where $imsi_V, imsi_{ms}$ are permanent mobile station identities:

$$\begin{aligned} & \mathtt{free}\ \ imsi_V. \\ & \mathtt{PV}_{ANON} \;=\; \mathtt{new}\, pvN;\ \mathtt{let}\ pbN = \mathtt{pub}(pvN)\ \mathtt{in} \\ & \qquad \mathtt{out}(c, pbN); \\ & \qquad\quad (!\mathtt{new}\ sk;\ \mathtt{new}\ imsi; \\ & \qquad\quad \mathtt{(!new}\ sqn;\ \mathtt{(MS\ |\ SN)))} \\ & \qquad \mid\ \mathtt{(new}\ sk;\ \mathtt{new}\ imsi_{ms}; \\ & \qquad\quad \mathtt{let}\ imsi\mathtt{=choice[}imsi_V, imsi_{ms}\mathtt{]\ in} \\ & \qquad\quad !\mathtt{new}\ sqn;\ \mathtt{(SN\ |\ MS)}). \end{aligned}$$

The left side of the choice represents a system where the mobile station with public identity $imsi_V$ can run the protocol. Our fixes of the identification procedure,

paging procedure and 3G AKA protocol as described in Chapter 6 are proved by `ProVerif` to satisfy anonymity (See table 7.1).

7.2 Automatic Verification Results and Remarks

We run the `ProVerif` tool on the 2G/3G identification procedure, on the 2G/3G IMSI paging procedure and on the 3G AKA protocol, in order to confirm that the tool would have detected the breaches of the privacy properties present in the 3GPP standard procedures. Even though the coding of the protocols in `ProVerif` is straightforward, the coding of the observational equivalences defining the privacy properties in term of bi-processes is not. In fact the biprocess structure is symmetrical while the definition of anonymityand and unlinkability is not. We showed in the previous Section how we obtained a symmetrical definition in terms of bi-processes. Moreover, we had to take particular care in avoiding false attacks that could be reported by the tool due to its abstractions. Indeed, we formally define privacy properties through observational equivalence, however, `ProVerif` adopts a stronger equivalence relation called diff-equivalence ($\approx_{\texttt{diff}}$). In particular, diff-equivalence can distinguish between the execution of different branches of a conditional statement even in the following case:

$$\texttt{if } a = a \texttt{ then } P \texttt{ else } P \not\approx_{\texttt{diff}} \texttt{if } a = b \texttt{ then } P \texttt{ else } P$$

and hence, although the above processes are observationally equivalent (P is executed regardless the result of the if statement evaluation), they do not satisfy diff-equivalence. We are dealing with this issue in our code for the verification at lines 4-5, 36, 73-74, 81, and 86-87 of the code in Appendix A. As expected, the verification with the `ProVerif` tool fails to prove the anonymity of the IMSI paging procedure and the unlinkability of both IMSI paging and 3G AKA protocols (see Table 7.2). In case of the IMSI paging procedure `ProVerif` exhibits actual

Properties	Identification	IMSI Paging	3G AKA
Unlinkability	√	√	√
Anonymity	√	√	√

NA Not Applicable √ Proved to hold × Attack found

TABLE 7.1: `ProVerif` results of the on Fixed Procedures

Properties	Identification	IMSI Paging	3G AKA
Unlinkability	×	×	×
Anonymity	×	×	√

NA Not Applicable √ Proved to hold × Attack found

TABLE 7.2: `ProVerif` results on current 3GPP Procedures

attack traces. In the case of the 3G AKA protocol, the anonymity property is proved to hold, while the unlinkability property verification fails. Although, the trace provided by `ProVerif` is a false attack, it does give a hint of the real attack by highlighting the test of the MAC received from the network as the source of the problem. The modelling of unlinkability and anonymity into diff-equivalences we showed in the previous Section can in general be adopted for protocols which do not require an initialization phase preceding the main protocol procedure. Hence, our method is not specific to the analysed protocols, and shows how to automatically verify unlinkability and anonymity on a wide class of protocols. The `ProVerif` code used for the automatic verification is available online [PVs] and in part in Appendix A.

Note that for verification purposes in our models of MS and SN we use randomised symmetric encryption to conceal the sequence number SQN instead of using the exclusive-or. Indeed, even if the theory allows to write a set of reduction rules to model the `xor` function, the `ProVerif` tool cannot deal with its algebraic properties. The use of randomised encryption anyway would achieve stronger properties with respect to the secrecy of the sequence number, we hence recommend the adoption of this modification in the standard protocol.

Authentication, Secrecy, Integrity. The main purpose of the 3G AKA protocol is to provide mutual authentication and establish session keys to be used for integrity protection and secrecy. Hence, our analysis would not be complete without ensuring that our privacy preserving version of the 3G AKA protocol still achieves the goals it was originally designed for. We verify mutual authentication and integrity properties as injective correspondence properties. We prove using `ProVerif` that the original properties of the 3G AKA protocol are preserved by our fixes; the verification results are shown in Table 7.3.

Properties	Identification	Paging	AKA
Secrecy			
IMSI	√	√	√
K_{IMSI}	NA	√	√
CK, IK	NA	NA	√
confidential information	NA	NA	√
Authentication	NA	NA	√
Integrity	NA	NA	√

NA Not Applicable √ Proved to hold × Attack found

TABLE 7.3: Results of the Automatic Verification of the Fixed Procedures

7.3 Unlinkability of the TMSI Reallocation Procedure

In this Section, we model the TMSI reallocation procedure and formally prove its unlinkability when fresh ciphering keys are established before each execution. We point out some limitations of the currently available automatic verification tools and develop one of the few manual proofs of labelled bisimilarity ofand a real protocol available in literature. In our proof we combine manual and automatic

proof techniques to overcome the issues arising when trying to automatically verify protocols using persistent states.

7.3.1 Model of the TMSI Reallocation

In this Section we present a formal model of the TMSI reallocation. We first introduce the equations we use to define randomized symmetric encryption and pairing.

Let $\Sigma = \{\mathtt{senc}/3, \mathtt{sdec}/2, \mathtt{pair}/2, \mathtt{fst}/1, \mathtt{snd}/1\}$, and consider the equations:

$\mathtt{sdec}(k, \mathtt{senc}(k, r, m)) = m,\ \mathtt{fst}(\mathtt{pair}(x, y)) = x,\ \mathtt{snd}(\mathtt{pair}(x, y)) = y.$

The first equation allows to decrypt a randomized encryption of message m, given the knowledge of the encryption key k. This is the usual rule to model randomized symmetric encryption. The two last rules allow to decompose a pair and retrieve its components.

Further, we introduce MS and SN modelling respectively a mobile station and a serving network sharing a private channel *dck*. This private channel models the fact that MS and SN can "securely" establish a shared session key by executing the authentication procedure. The private channel d models a mobile station's memory (or state) recording the currently assigned TMSI. Input messages are read from the *dw* channel and output messages are sent on the *up* channel. The system of multi-sessions mobile stations executing the TMSI reallocation with a

multi-session serving network is represented by the process M

$$
\begin{array}{rcl}
Init & \stackrel{def}{=} & \overline{d}\,\langle id\rangle \\
MS & \stackrel{def}{=} & \nu\ ck.\ \nu\ mr.\ d(x).\ \overline{up}\langle x\rangle.\ \overline{dck}\langle ck\rangle.\ dw(y). \\
 & & \quad \mathit{if}\ \ \mathtt{fst}(\mathtt{sdec}(ck,y)) = \text{TMSI\_REALL}\ \ \mathit{then} \\
 & & \quad\ \overline{up}\langle \mathtt{senc}(ck, mr, \text{COMPLETE})\rangle. \\
 & & \quad\ \overline{d}\langle \mathtt{snd}(\mathtt{sdec}(ck,y))\rangle \\
 & & \quad \mathit{else}\ 0 \\
SN & \stackrel{def}{=} & \nu\ nid.\ \nu\ sr.\ dw(z).\ dck(xck). \\
 & & \quad \overline{up}\langle \mathtt{senc}(xck, sr, \mathtt{pair}(\text{TMSI\_REALL},\ nid))\rangle.\ dw(w) \\
M & \stackrel{def}{=} & \nu\ dck.\ (!(\nu\ d.\ \nu\ id.\ (Init\ |!MS))\ |!SN)
\end{array}
$$

The $Init$ process initializes the MS memory by storing in it the initial pseudonym id. The current pseudonym is stored in the memory d and is sent by the MS with its first message. The MS then establishes a session key with the network, modelled here by the communication of a new key ck over a private channel dck (note that in this model a fresh session key is established before the execution of each TMSI reallocation). The mobile station then receives a message and checks if it is a legitimate TMSI reallocation command message encrypted by the network using the agreed session key ($\mathit{if}\ \ \mathtt{fst}(\mathtt{sdec}(ck,y)) = \text{TMSI\_REALL}$). In this case it sends a TMSI reallocation complete message ($\overline{up}\langle \mathtt{senc}(ck, mr,\ \text{COMPLETE})\rangle$) and updates its own memory with the new pseudonym received in the TMSI Reallocation command ($\overline{d}\langle \mathtt{snd}(\mathtt{sdec}(ck,y))\rangle$). For simplicity, we do not model the eventual updating of the location area.

7.3.2 Proof of Unlinkability of the Fixed TMSI Reallocation

We recall that the definition of strong unlinkability given in [ACRR10] requires an ideal system, where each agent can execute the protocol at most once (and hence is unlinkable by construction) to be undistinguishable from a system, where each agent can execute the protocol an unbounded number of times. Refer to Section 7.1.1 for the formal definition.

The definition of strong unlinkability allows us to formally analyse the TMSI reallocation procedure and establish if it achieves the desired unlinkability property when a new session key is established prior to each execution of the TMSI reallocation procedure. We showed in Chapter 5 that when this does not happen a linkability attack can be mounted to trace mobile telephony users.

Let the $Init, MS$ and SN processes be as defined in Section 7.3.1,pag. 103. We define:

$$SSA \stackrel{def}{=} \nu\, d.\nu\, id.(Init \mid MS)$$
$$MSA \stackrel{def}{=} \nu\, d.\nu\, id.(Init \mid !MS)$$

The processes SSA and MSA are respectively a single-session and a multi-session mobile station agent. Single-session mobile stations can only execute one session of the TMSI reallocation procedure hence are unlinkable by construction and are part of the ideal system, while the multi-session agents represent the mobile stations of the real systems i.e. the ones we want to prove to be unlinkable, although they can execute several sessions of the procedure.
Let S and M (the single-session and the multi-session system, respectively) be two closed processes defined as follows:

$$S \stackrel{def}{=} \nu\, dck.(!SSA \,|!SN)$$
$$M \stackrel{def}{=} \nu\, dck.(!MSA \,|!SN)$$

The process S represents an unbounded number of mobile stations executing the TMSI reallocation procedure at most once. The process M represents an unbounded number of mobile stations which can execute the TMSI reallocation procedure an unbounded number of times. We want to prove that M and S are labelled bisimilar and hence that M satisfies unlinkability.

`ProVerif` is to date the only tool able to automatically verify observational equivalence based properties for unbounded processes like the ones considered in this work. However, the presence of the memory (state) for the storage of the currently assigned identity makes the use of `ProVerif` to automatically analyse this protocol not feasible. In fact, `ProVerif` cannot prove the observational equivalence of the following toy bi-process[1], which models one process sending a fresh name on a public channel and another reading a fresh name from its state (modelled by the private channel d) and then sending it on a public channel:

```
free c.

let P = (!(new n; in(d, x); out(d, n);
           out(c, choice[x, n]))).

process new d; (P | (new m; out(d, m)))
```

This happens because the abstractions `ProVerif` does for the sake of termination allow the process using the private channel to never consume the input. Hence, once a name is sent on the private channel `d`, that name can be read from it again

[1] A biprocess represents a pair of processes differing in the choice of some message, this choice is modelled by the `choice[m,n]` construct. See [Bla01, BAF05] for more details about `ProVerif`.

and again, making the two processes not observationally equivalent. This is one of the reasons that led to the development of `StatVerif` [ARR11], an extension of `ProVerif` which deals with stateful processes. However, `StatVerif` is not suitable in our case since it does not yet handle observational equivalence. For this reason we carry out a manual analysis instead. In the next section we give a sketch of the proof of the unlinkability of the TMSI reallocation procedure (Proposition 1) when performed by establishing a fresh session key prior to each execution. The full proof is given in Appendix B. Note that Abadi and Rogaway [AR00] showed that for what they call type-0 security encryption schemes, i.e. repetition concealing, which-key concealing, and message-length concealing encryption schemes observational equivalence is computationally sound.

7.3.3 Unlinkability Proof Sketch

To be able to describe the relation $\mathcal{R}$ witnessing that $S \approx_l M$ we define partial execution steps of the multi (resp. single)-session process components as specified below. The process $MMS^k_{i,j}$ represents the i^{th} mobile station executing the k^{th} step of its j^{th} session of the TMSI reallocation protocol, while the process $SMS^k_{i,j}$ represents the $(i+j)^{th}$ mobile station executing the k^{th} step of its unique session. The process SN^l_m represents the l^{th} step of the m^{th} session of the serving network. The key point of the proof is to show that processes $MMS^k_{i,j}$ and $SMS^k_{i,j}$ simulate each other. We now give an outline of how this simulation works, by explaining how to match transitions in the multi-session and single-session processes.

1. Any transition within a session of some mobile station is a transition from $MMS^k_{i,j}$ to $MMS^{k'}_{i,j}$, with $k' > k$. There is always a matching transition within the single session of the corresponding mobile station from $SMS^k_{i,j}$ to $SMS^{k'}_{i,j}$, and vice versa.

2. The transitions for the serving network are the same in the multi-session and the single-session process, hence they match trivially.

3. The start of a new session for the same mobile station is modelled by a transition from $MMS^6_{i,j} \mid MS$ to $MMS^7_{i,j} \mid MMS^0_{i,j+1}$. The corresponding transitions in the single-session process, which are $SMS^6_{i,j}$ to $SMS^7_{i,j}$ and $Init \mid MS$ to $SMS^0_{i,j+1}$, model the use of an additional mobile station to simulate this extra-session.

4. The use of an additional mobile station in the multiple session process is modelled by a transition from $Init|MS$ to $MMS^0_{i+1,1}$. There is always a matching transition from $Init|MS$ to $SMS^0_{i+1,1}$ in the single-session process, and vice versa.

So far, this produces a perfect match between transitions for the multiple-session process and the single-session processes in cases 1, 2 and 4. In case 3, we still have to find a matching transition for the transition from $Init \mid MS$ to $SMS^0_{i,j+1}$ without $SMS^6_{i,j}$ being present. In this case we use the fact that $SMS^0_{i,j+1}$ and $SMS^0_{i+1,1}$ are α-equivalent and use case 4 to find a matching transition in the multi-session process. This point is the key part of the proof and shows that the single-session system really models several sessions of the same mobile station by using several mobile stations.

We now present this proof in more details. We start by defining matching pairs of multi and single-session mobile stations for each evolution step k. Note that each single session mobile station uses a different secret channel, modelling a memory cell, $d_{i,j}$ while each multi-session mobile station uses the same secret channel across all the sessions $d_{i,1}$.

Let $i,\ j \in \mathbb{N}$. We denote:

$$Init_{i,j} \stackrel{def}{=} \overline{d_{i,j}}\langle id_{i,j}\rangle$$

$$\begin{aligned} MChk_{i,j} \stackrel{def}{=}\ & \textit{if}\ \texttt{fst}(\texttt{sdec}(ck_{i,j},\ y_{i,j})) = \text{TMSI\_REALL}\ \textit{then} \\ & \quad \overline{up}\langle \texttt{senc}(ck_{i,j},\ mr_{i,j},\ \text{COMPLETE})\rangle. \\ & \quad \overline{d_{i,1}}\langle \texttt{snd}(\texttt{sdec}(ck_{i,j},\ y_{i,j}))\rangle \\ & \textit{else}\ 0 \end{aligned}$$

$$\begin{aligned} SChk_{i,j} \stackrel{def}{=}\ & \textit{if}\ \texttt{fst}(\texttt{sdec}(ck_{i,j}, y_{i,j})) = \text{TMSI\_REALL}\ \textit{then} \\ & \quad \overline{up}\langle \texttt{senc}(ck_{i,j},\ mr_{i,j},\ \text{COMPLETE})\rangle. \\ & \quad \overline{d_{i,j}}\langle \texttt{snd}(\texttt{sdec}(ck_{i,j},\ y_{i,j}))\rangle \\ & \textit{else}\ 0 \end{aligned}$$

$$\begin{aligned} MMS^0_{i,j} &\stackrel{def}{=} d_{i,1}(x_{i,j}).\overline{up}\langle x_{i,j}\rangle.\overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).MChk_{i,j} \\ SMS^0_{i,j} &\stackrel{def}{=} d_{i,j}(x_{i,j}).\overline{up}\langle x_{i,j}\rangle.\overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).SChk_{i,j} \\ MMS^1_{i,j} &\stackrel{def}{=} \overline{up}\langle M_{i,j}\rangle.\overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).MChk_{i,j} \\ SMS^1_{i,j} &\stackrel{def}{=} \overline{up}\langle id_{i,j}\rangle.\overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).SChk_{i,j} \\ MMS^2_{i,j} &\stackrel{def}{=} MX_{i,j} \mid \overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).MChk_{i,j} \\ SMS^2_{i,j} &\stackrel{def}{=} SX_{i,j} \mid \overline{dck}\langle ck_{i,j}\rangle.dw(y_{i,j}).SChk_{i,j} \\ MMS^3_{i,j} &\stackrel{def}{=} MX_{i,j} \mid dw(y_{i,j}).MChk_{i,j} \\ SMS^3_{i,j} &\stackrel{def}{=} SX_{i,j} \mid dw(y_{i,j}).SChk_{i,j} \\ MMS^4_{i,j} &\stackrel{def}{=} MX_{i,j} \mid MChk_{i,j}\{^{N_{i,j}}/_{y_{i,j}}\} \\ SMS^4_{i,j} &\stackrel{def}{=} SX_{i,j} \mid SChk_{i,j}\{^{N_{i,j}}/_{y_{i,j}}\} \end{aligned}$$

$$
\begin{array}{lcl}
MMS^5_{i,j} & \stackrel{def}{=} & MX_{i,j} \mid \overline{up}\langle \mathtt{senc}(ck_{i,j},\ mr_{i,j},\ \textsc{Complete})\rangle. \\
 & & \quad \overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle \\
SMS^5_{i,j} & \stackrel{def}{=} & SX_{i,j} \mid \overline{up}\langle \mathtt{senc}(ck_{i,j},\ mr_{i,j},\ \textsc{Complete})\rangle. \\
 & & \quad \overline{d_{i,j}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle \\
MMS^6_{i,j} & \stackrel{def}{=} & MX_{i,j} \mid MK_{i,j} \mid \overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle \\
SMS^6_{i,j} & \stackrel{def}{=} & SX_{i,j} \mid SK_{i,j} \mid \overline{d_{i,j}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle \\
 & & \\
MMS^7_{i,j} & \stackrel{def}{=} & MX_{i,j} \mid MK_{i,j} \mid 0 \\
SMS^7_{i,j} & \stackrel{def}{=} & SX_{i,j} \mid SK_{i,j} \mid \overline{d_{i,j}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle \\
MMS^8_{i,j} & \stackrel{def}{=} & MX_{i,j} \mid 0 \\
SMS^8_{i,j} & \stackrel{def}{=} & SX_{i,j} \mid 0 \\
MX_{i,j} & \stackrel{def}{=} & \{^{M_{i,j}}/_{x_{i,j}}\} \\
SX_{i,j} & \stackrel{def}{=} & \{^{id_{i,j}}/_{x_{i,j}}\} \\
 & & \\
SK_{i,j}, MK_{i,j} & \stackrel{def}{=} & \{^{\mathtt{senc}(ck_{i,j},\ mr_{i,j},\ \textsc{Complete})}/_{k_{i,j}}\} \\
 & & \\
RMS_i & \stackrel{def}{=} & \nu\ ck.\nu\ mr.(d_{i,1}(x).\overline{up}\langle x\rangle.\overline{dck}\langle ck\rangle.dw(y). \\
 & & \quad \mathit{if}\ \mathtt{fst}(\mathtt{sdec}(ck,\ y)) = \text{TMSI\_Reall}\ \mathit{then} \\
 & & \quad\ \overline{up}\langle \mathtt{senc}(ck,\ mr,\ \textsc{Complete})\rangle. \\
 & & \quad\ \overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck,\ y))\rangle \\
 & & \quad \mathit{else}\ 0 \\
 & & \\
M_{i,j} & \stackrel{def}{=} & \begin{cases} id_{i,j} & \text{if } j = 1 \\ nid_{i,j-1} & \text{otherwise} \end{cases}
\end{array}
$$

$$
\begin{array}{lcl}
\widetilde{ss}_{i,j} & \stackrel{def}{=} & id_{i,1},\ d_{i,1},\ ck_{i,1},\ mr_{i,1}, \ldots, id_{i,j},\ d_{i,j},\ ck_{i,j},\ mr_{i,j} \\
\widetilde{ms}_{i,j} & \stackrel{def}{=} & id_{i,1},\ d_{i,1},\ ck_{i,1},\ mr_{i,1}, \ldots,\ ck_{i,j},\ mr_{i,j}
\end{array}
$$

Note that a full execution of the TMSI reallocation procedure by a multi (resp. single)-session mobile station goes through the first 6 evolution steps. In particular, a new session of the TMSI reallocation protocol can be executed (by the multi-session MS) only after the mobile station fully completed the previous session ending up at step $k = 6$ where the synchronization on the memory channel d is enabled by the output of the newly allocated temporary identity $\overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j},\ N_{i,j}))\rangle$. In case the *if* condition is not satisfied both the multi and the single-session mobile stations end up in a deadlock state ($k = 8$). This simplification is not fundamental for the unlinkability of the procedure to be satisfied. However, it simplifies an already long and error prone manual proof.

$$SN_i^0 \stackrel{def}{=} \nu\ nid_i.\nu\ sr_i.dw(z_i).dck(xck_i).\ \overline{up}\langle \mathtt{senc}(xck_i,\ sr_i,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid_i))\rangle.\ dw(w_i)$$

$$SN_i^1 \stackrel{def}{=} \nu\ nid_i.\nu\ sr_i.dck(xck_i).\ \overline{up}\langle \mathtt{senc}(xck_i,\ sr_i,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid_i))\rangle.\ dw(w_i)$$

$$SN_i^2 \stackrel{def}{=} \overline{up}\langle \mathtt{senc}(xck_i,\ sr_i,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid_i))\rangle.\ dw(w_i)$$

$$SN_i^3 \stackrel{def}{=} \{^{\mathtt{senc}(ck_i,\ sr_i,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid_i))}/_{y_i}\} \mid dw(w_i)$$

$$SN_i^4 \stackrel{def}{=} \{^{\mathtt{senc}(ck_i,\ sr_i,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid_i))}/_{y_i}\}$$

$$SN_{i,j}^k \stackrel{def}{=} SN_l^k\{^{ck_{i,j}}/_{xck_l},\ ^{y_{i,j}}/_{y_l},\ ^{w_{i,j}}/_{w_l},\ ^{nid_{i,j}}/_{nid_l},\ ^{sr_{i,j}}/_{sr_l}\},\ k \geq 2$$

$$\widetilde{nid}_{i,j} \stackrel{def}{=} nid_{i,1},\ sr_{i,1}, \ldots nid_{i,j}\ sr_{i,j}$$

$MX_{i,j}, MK_{i,j}$, and SN_i^4 (resp. $SX_{i,j}, SK_{i,j}$ and SN_i^4) are the possible active substitutions resulting from one full execution of the TMSI reallocation procedure by the j^{th} session of the i^{th} mobile station in the multi-session system (resp. by the $i+j^{th}$ mobile station in the single-session system). RMS_i is the replicated part of the multi-session mobile station agent. Note that we group the name restrictions and we bring them in front of the process.

We define the grouped multi-session system component $GMS_{i,j}[\_]$ representing the leftovers after the execution of j sessions of the i^{th} mobile station and the simulating grouped single-session system component $GSS_{i,j}[\_]$ representing the leftovers after the execution of j single session mobile stations simulating the j sessions of the i^{th} mobile station of the multi-session system, as follows:

$$GMS_{i,j}[\_] \stackrel{def}{=} \nu\ \widetilde{ms}_{i,j}.\nu\ \widetilde{nid}_{i,l}.(MMS_{i,1}^7 \mid \cdots \mid MMS_{i,j-1}^7 \mid \_ \mid !RMS_i)$$

$$GSS_{i,j}[\_] \stackrel{def}{=} \nu\ \widetilde{ss}_{i,j}.\nu\ \widetilde{nid}_{i,l}.(SMS_{i,1}^7 \mid \cdots \mid SMS_{i,j-1}^7 \mid \_)$$

$$\text{where} \quad l \in \{j-1, j\}$$

The grouped multi (resp. single)-session system components are the building blocks of the bisimulation relation. They basically define how the grouped single-session MSs can mimic the structure resulting by the evolution of a multi-session mobile station. We define the symmetric relation between the single-session and the multi-session system to be:

$$\mathcal{R} \stackrel{def}{=} \{ \;\; (C,\ D),\ (D,\ C) : \exists\ n,\ m \geq 0,$$

$$A \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN),$$
$$B \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN),$$

where $\forall i,\ 1 \leq i \leq n,\ \exists l_i,\ k_{l_i},\ l_i \geq 0, 1 \leq k_{l_i} \leq 8$ such that

$$C_i = GSS_{i,l_i}[SMS_{i,l_i}^{k_{l_i}} \mid SSN_{i,l_i}] = \nu\ \widetilde{ss}_{i,l_i}.\nu\ \widetilde{nid}_{i,j}.(SMS_{i,1}^{7} \mid \cdots \mid SMS_{i,l_i-1}^{7} \mid SMS_{i,l_i}^{k_{l_i}} \mid SSN_{i,l_i})$$

$$D_i = GMS_{i,l_i}[MMS_{i,l_i}^{k_{l_i}} \mid MSN_{i,l_i}] = \nu\ \widetilde{ms}_{i,l_i}.\nu\ \widetilde{nid}_{i,j}.(MMS_{i,1}^{7} \mid \cdots \mid MMS_{i,l_i-1}^{7} \mid MMS_{i,l_i}^{k_{l_i}} \mid MSN_{i,l_i} \mid !RMS_i)$$

$$SSN_{i,l_i} = MSN_{i,l_i} = SN_{i,1}^{h_1} \mid \cdots \mid SN_{i,l_i-1}^{h_{l_i-1}} \mid L^{h_{l_i}},\ h_1, \ldots, h_{l_i-1} \geq 2$$

$$L^{h_{l_i}} = \begin{cases} 0 & \text{if } k_{l_i} \in \{1,\ 2\} \\ SN_{i,l_i}^{h_{l_i}} & \text{otherwise} \end{cases} \qquad j = \begin{cases} l_i - 1 & \text{if } L^{h_{l_i}} = 0 \\ l_i & \text{otherwise} \end{cases}$$

$$PSN_m = SN_{j_1}^1 \mid \cdots \mid SN_{j_m}^1, \text{ for some } j_1, \ldots,\ j_m \in \{0,\ 1\}$$

$\}$

We want to prove that $\mathcal{R}$ is a bisimulation. To ease this proof, we define a lemma dealing with the bisimulation part of the proof and a lemma dealing with static equivalence. Informally, Lemma 7.1 states that the actions of a system can be mimicked by actions of the other system and vice versa. Formally:

Lemma 7.1. *Let* $C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN)$, $D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !\overline{SA} \mid !SN)$ *such that* $SA = SSA$ *(resp.* $SA = MSA$*) and* $(\overline{SA} = MSA$ *(resp.* $\overline{SA} = SSA)$ *and* $(C, D) \in \mathcal{R}$
if $C \stackrel{\ell}{\rightarrow} C'$ *with* $fv(\ell) \subseteq dom(C)$ *and* $bn(\ell) \cap fn(D) = \emptyset$ *then* $D \stackrel{\ell}{\rightarrow} D'$ *and* $(C', D') \in \mathcal{R}$ *for any* $\ell \in \{\tau, \alpha\}$.

The proof of Lemma 7.1 relies on the proof of two extra lemmas (Lemma 7.3 and Lemma 7.4 in the Appendix B.2) which informally state that if the single (resp. multi)-session system can do a transition then either one of the grouped single (resp. multi)-session system components (i.e. one of the C_i, respectively D_i) can do the transition, possibly synchronizing with one of the SN_j^1 components of the PSN_m process (i.e. the MS synchronizes with the SN. This step models the establishment of means for ciphering of the TMSI reallocation protocol); or one of the components under replication is unrolled and does the transition; or one of the mobile stations starts a new session (in case of the multi-session system). The formal statements and proofs of Lemma 7.1, 7.3, 7.4 are given in Appendix B.2.

To complete the proof of Proposition 1 we have to prove that the processes obtained after each simulation step are statically equivalent. This is stated by Lemma 7.2.

Lemma 7.2. *Let* $(C, D) \in \mathcal{R}$ *then* $C \approx_s D$

In order to ease this proof we define the following substitutions:

$$\sigma_{i,j}^{id} \stackrel{def}{=} \{^{id_{i,1}}/_{x_{i,1}}, {}^{id_{i,2}}/_{x_{i,2}}, \ldots, {}^{id_{i,j}}/_{x_{i,j}}\}$$

$$\sigma_{i,j}^{M} \stackrel{def}{=} \{^{id_{i,1}}/_{x_{i,1}}, {}^{M_{i,2}}/_{x_{i,2}}, \ldots, {}^{M_{i,j}}/_{x_{i,j}}\}$$

$$\sigma_{i,j}^{K} \stackrel{def}{=} \{^{\mathtt{senc}(ck_{i,1},\ mr_{i,1},\ \textsc{Complete})}/_{k_{i,1}}, \ldots, {}^{\mathtt{senc}(ck_{i,j},\ mr_{i,j},\ \textsc{Complete})}/_{k_{i,j}}\}$$

$$\sigma_{i,j}^{nid} \stackrel{def}{=} \{^{\mathtt{senc}(ck_{i,1},\ sr_{i,1},\ \mathtt{pair}(\text{TMSI\_}\textsc{Reall},\ nid_{i,1}))}/_{y_{i,1}}, \ldots, {}^{\mathtt{senc}(ck_{i,j},\ sr_{i,j},\ \mathtt{pair}(\text{TMSI\_}\textsc{Reall},\ nid_{i,j}))}/_{y_{i,j}}\}$$

Moreover, we prove in Lemma 7.5 that the structure of the frame of a single (resp. multi)-session system is as follows:

Let $(C, D) \in \mathcal{R}, C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN), D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ then $\varphi(C) \equiv \nu\ dck.(\varphi(C_1) \mid \cdots \mid \varphi(C_n))$, $\varphi(D) \equiv$

$\nu\ dck.(\varphi(D_1) \mid \cdots \mid \varphi(D_n))$ where $\forall i,\ l_i\ 1 \leq i \leq n,\ l_i \geq 0$,

$$\begin{aligned}
\varphi(C_i) &\equiv \varphi(GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}]) \\
&\equiv \nu\ \widetilde{ss}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.(\sigma^{id}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}}) \\
\varphi(D_i) &\equiv \varphi(GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}]) \\
&\equiv \nu\ \widetilde{ms}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.(\sigma^{M}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})
\end{aligned}$$

We then use the obtained frame structure to define a `ProVerif` bi-process that generates the frame of the multi-session and single-session processes. This allows us to automatically prove the static equivalence. Hence, the full proof combines manual and automatic techniques. The full proofs of Lemma 7.5 and Lemma 7.2 are detailed in the Appendix B.3.

We can now easily prove that the TMSI reallocation procedure preserves unlinkability if a new session key is established before each execution. This is formalised in the following proposition.

Proposition 1. *We show that* $S \approx_l M$.

Proof. We show that $(M,\ S) \in \mathcal{R}$.
Let $C = S$ and $D = M$, let $n = 0,\ m = 0$ then $C \equiv \nu\ dck.(!SSA\ |!SN) \equiv S$ and $D \equiv \nu\ dck.(!MSA\ |!SN) \equiv M$ and $(S,\ M) \in \mathcal{R}$.
We show that $\mathcal{R}$ is a bi-simulation.
We show that $\forall\ (C, D) \in \mathcal{R},\ C \approx_s D$: trivially follows by Lemma 7.2.
We show that if $C \xrightarrow{\tau} C'$ then $\exists\ D'$ such that $D \Rightarrow D'$ and $(C',\ D') \in \mathcal{R}$: trivially follows by Lemma 7.1.
We show that if $C \xrightarrow{\alpha} C'$ and $fv(\alpha) \subseteq dom(C),\quad bn(\alpha) \cap fn(D) = \emptyset$ then $\exists\ D'$ such that $D \Rightarrow\xrightarrow{\alpha}\Rightarrow D'$ and $(C',\ D') \in \mathcal{R}$: trivially follows by Lemma 7.1 □

7.4 Remarks

Our analysis of the TMSI reallocation procedure only considers a simplified version of the protocol and abstracts away the establishment of CK through the execution of the key agreement protocol. Hence, we do not claim any guarantees provided by this proof. However, our formal analysis of TMSI reallocation still allows to establish that the TMSI reallocation procedure as we modeled it in this Chapter is not subject to linkability attacks. This shows that the establishment of new session key before the execution of the TMSI reallocation procedure could be used as countermeasure to the attack we propose in Section 5.1.1.

We formally proved that if new session keys are established for each TMSI reallocation execution then unlinkability is preserved[2]. Our proof of unlinkability is one of the few examples in the literature of a proof of labelled bisimilarity of a real-sized protocol. Such manual proofs give useful insights on the way one could automate them, and thus pave the way to automating labelled bisimilarity proofs. In particular this proof shows how the bisimulation and the static equivalence part of the proof can be dealt with separately and how to take advantage of automatic verification tools to prove static equivalence of unbounded protocols.

[2]In our model no action is taken by either the network nor the MS in case of failure of the reallocation procedure. This reflects our understanding of the procedure from the standard specification. Hence, our proof does not give any guarantees in case further messages are sent over-the-air as a result of a failure in executing the TMSI reallocation.

8

Automatic Verification of Equivalences for Stateful Processes: A StatVerif Extension

Our aim in this Chapter is to extend the range of protocols and protocol properties verifiable using automatic verification tools. Recent studies [ARR11, DKRS11, Gut12, Mod10, Her06] pointed out the lack of automatic support for the verification of protocols with persistent states. Persistent states are useful when agents need to maintain some information across consecutive sessions in order to take sensible action according to the protocol specifications. Protocols which may require the encoding of states or memory cells are for example:

- anonymous or pseudonymous based protocols, which involve the use of periodically changing identities, such as mobile phone protocols and vehicular network protocols;

- protocols that involve secure hardware such as smart cards, Radio Frequency Identification (RFID) tags, USB tokens and Trusted Platform Modules (TPMs);
- protocols involving databases such as electronic voting, file sharing, conference management, web transactions, fair exchange and contract signing protocols.

Persistent state can be encoded in applied pi-calculus using private channels. However, this encoding makes reasoning with stateful protocols more difficult and less intuitive. In fact, one usually needs to input and output on the private channel to model read and write of the state value, and because these operations are synchronous an auxiliary process should be modelled to repeatedly input and output on the channel so to make read and write of the state content always available and avoid deadlocks. Moreover, the use of private channels to simulate state cells is not adequate for automatic verification using the `ProVerif` tool since it introduces false attacks. This is due to the fact that a value sent on a private channel (and hence the one of a state) is always available even if a new value is subsequently sent, i.e. older values of the state can be used even after the state is updated.

The wide application of stateful protocols together with the lack of automatic support for the verification of their security properties led to the development of the `StatVerif` tool. The `StatVerif` tool is an automatic verifier for the verification of reachability properties of stateful protocols. `StatVerif` does not handle equivalence based properties. However, many interesting properties of stateful protocols such as anonymity, unlinkability, and strong secrecy are equivalence-based properties. We aim to close this gap. For this reason we extend the automatic verification tool `StatVerif` [ARR11] to the verification of observational equivalence based properties. In fact, the `StatVerif` tool can only deal with reachability properties of Stateful protocols.

8.1 Related Work

Guttman [Gut12] enriches the strand space model with multiset rewriting in order to deal with protocol's states. This technique is then used to verify a fair exchange protocol, however there is no automatic support for it. Multiset rewriting is used in combination with constraint solving by the tool *tamarin* [SMCB12] to automatically support the verification of Authentication and Key Exchange (AKE) protocols based on Diffie-Hellman. The underlying model makes tamarin suitable for the verification of stateful protocols and it has been successfully used to verify hardware password token protocols in interactive (i.e. with human intervention) mode. In [Mod10] the IF (Intermediate Format) language of AVISPA (Automated Validation of Internet Security Protocols and Applications) is extended in order to explicitly deal with fresh data and databases, using the set-membership abstraction and resulting in the AIF language. The author implements an automatic translation tool from AIF to Horn clauses. The Horn clauses can be generated both in the `ProVerif` syntax and in the SPASS syntax, hence allowing the automatic verification with the respective tools. They automatically verify reachability properties of non-trivial stateful protocols. States are manually encoded into Horn clauses in [DKRS11] to model TPM's registers. The method used is strictly tied to the TPM configuration registers scenario. The closest work to ours is the one presented in [ARR11] where the `StatVerif` tool is developed. In this Chapter we extend the `StatVerif` calculus, semantics and translation into Horn clauses in order to deal with the verification of equivalence-based properties. For this purpose we rely as well on the theory developed in [BAF05] for the automatic verification of equivalence of bi-processes. We also take advantage of the theory for a stateful applied pi-calculus as developed in [ALRR14]. The authors define observational equivalence and labelled bisimilarity for stateful processes with and without public cells. In particular, the definition of observational equivalence with public cells is the one we use in this work.

8.2 Our Contribution

Firstly, we extend the `StatVerif` calculus to support the modelling of bi-processes and equational theories. Secondly, we develop a translation of the `StatVerif` bi-processes into Horn clauses. Finally, we implement `StatVerif` for observational equivalence and test it on some examples.

8.3 Background: `StatVerif` Process Language

We first introduce the `StatVerif` process language as presented in [ARR11] to model and automatically verify stateful processes. This language extends the `ProVerif` process language presented in [BAF05]. It allows to model concurrent processes which communicate by sending terms built over a signature including names and variables. These terms model the messages that are exchanged during the execution of a protocol.

8.3.1 Syntax and Informal Semantics

Figure 8.1 summarizes the syntax of the basic `StatVerif` calculus as presented in [ARR11]. We assume an infinite set of names $a, b, c, k, s, \ldots$, an infinite set of variables $x, y, z, \ldots$, and a signature E (a set of function symbols, with arities). We distinguish function symbols in constructors f and destructors g. We write h for a constructor or a destructor. Destructors represent primitives that can visibly succeed or fail, while equations model primitives that always succeed even though they may return junk terms in some cases. Terms $M, N, \ldots$ are built over variables, names, and constructor applications of the form $f(M_1, \ldots, M_n)$. Terms are subject to an equational theory, we write $\Sigma \vdash M = N$ for an equality modulo the equational theory, and $\Sigma \vdash M \neq N$ for an inequality modulo the equational theory.

(We write $M = N$ and $M \neq N$ for syntactic equality and inequality, respectively.) The equational theory is defined by a finite set of equations $\Sigma \vdash M_i = N_i$, where M_i and N_i are terms that contain only constructors and variables. The equational theory is then obtained from this set of equations by reflexive, symmetric, and transitive closure, closure by substitution, and closure by context application (if $\Sigma \vdash M = N$ then $\Sigma \vdash M'\{M/x\} = M'\{N/x\}$, where $\{M/x\}$ is the substitution that replaces x with M).

As implemented in `ProVerif`, destructors are partial, non-deterministic operations that processes can apply on terms. The semantics of a destructor g is defined by a finite set $\mathtt{def}_\Sigma(g)$ of rewrite rules of the form $g(M'_1, \ldots, M'_n) \to M'$, where $M'_1, \ldots, M'_n, M'$ are terms that contain only constructors and variables, the variables of M' are bound in $M'_1, \ldots, M'_n$, and variables are subject to renaming. The destructor application $g(M_1, \ldots, M_n)$ is defined if and only if there exists a substitution σ and a rewrite rule $g(M'_1, \ldots, M'_n) \to M'$ in $\mathtt{def}_\Sigma(g)$ such that $M_i = \sigma M'_i$ for all $i \in \{1, \ldots, n\}$, and $g(M_1, \ldots, M_n) \to \sigma M'$. In order to avoid distinguishing constructors and destructors in the definition of term evaluation, we let $\mathtt{def}_\Sigma(f)$ be $\{f(x_1, \ldots, x_n) \to f(x_1, \ldots, x_n)\}$. The definition of term evaluation is not present in [ARR11], we define term evaluation as in [BAF05] (see Figure 8.2). Moreover, as in [BAF05], the prefix `eval` is here introduced in order to indicate when terms are evaluated. However, it can be ignored for the moment, since `eval` f and f have the same semantics when f is a constructor, and destructors are used only with `eval`.

Processes $P, Q, R, \ldots$ include the empty process 0 which does nothing; the input process $\mathtt{in}(M, x); P$ which models the input of a term on a channel M, the term is then substituted for x in the process P. The process $\mathtt{out}(M, N); P$ outputs a term N on a channel M and then executes P. $P \mid Q$ models the concurrent execution of P and Q. The replication $!P$ models the concurrent execution of an unbounded number of copies of the process P. The name restriction `new` $a; P$ is used to model that a is a fresh random number or key. The let construct

$M, N ::=$	terms
x, y, z	variables
a, b, c, k, s	names
$f(M_1, \ldots, M_n)$	constructor
$D ::=$	term evaluation
M	term
`eval` $h(D_1, \ldots, D_n)$	evaluation
$P, Q ::=$	processes
0	nil
`out`$(M, N); P$	output
`in`$(M, x); P$	input
$P \mid Q$	parallel
$!P$	replication
`new` $a; P$	restriction
`let` $x = D$ `in` P `else` Q	destructor
$[s \mapsto M]$	state
`read` $s_1, \ldots, s_n$ `as` $x_1, \ldots, x_n; P$	read
$s_1, \ldots, s_n := M_1, \ldots, M_n; P$	assign
`lock` $s_1, \ldots, s_n; P$	lock
`unlock` $s_1, \ldots, s_n; P$	unlock

FIGURE 8.1: StatVerif syntax

`let` $x = D$ `in` P `else` Q behaves as P where x is substituted by the evaluation of the term D, if the evaluation of D succeeds, and behaves as Q otherwise. The state declaration $[s \mapsto M]$ represents a cell s that has the initial value M. The process `read` $s_1, \ldots, s_n$ `as` $x_1, \ldots, x_n; P$ binds the variables $x_1, \ldots, x_n$ to the values stored in the cells $s_1, \ldots, s_n$ respectively and then continues as P. The assignment $s_1, \ldots, s_n := M_1, \ldots, M_n; P$ assigns the values $M_1, \ldots, M_n$ to the cells $s_1, \ldots, s_n$ respectively and then continues as P. The process `lock` $s_1, \ldots, s_n; P$ locks the state cells $s_1, \ldots, s_n$ for the exclusive access of P and `unlock` $s_1, \ldots, s_n; P$ releases the lock on the state cells $s_1, \ldots, s_n$, and continues as P. The full syntax of `StatVerif` is subject to the additional restrictions requiring the state declaration $[s \mapsto M]$ to occur only once for a given cell name s, and only within the scope of new, a parallel. It may not be in the scope of an input, output, conditional,

let, assignment, lock, or unlock. Note that a process that executes a parallel or a replication after locking one or more cells, but before unlocking them, will block according to the semantics. Such a syntactic construction is therefore not useful. The conditional `if` $M = N$ `then` P `else` Q is not included in the syntax and can be defined as `let` $x =$ `equals`(M, N) `in` P `else` Q where x is a fresh variable and `equals` is a binary destructor such that `equals`$(x, x) \rightarrow x$. This destructor is always included in Σ. The sets of free names and free variables in P are denoted by $fn(P)$ and $fv(P)$ respectively, and are defined as usual. We write $cells(P)$ to denote the free cells in P. Note that $cells(P) \subseteq fn(P)$. A process is closed if it has no free variables. We identify processes up to renaming of bound names and variables. As usual, `new` a binds a, `in`(M, x) and `let` $x = D$ `in` P bind x, moreover `read` $s_1, \ldots, s_n$ `as` $x_1, \ldots, x_n$ binds $x_1, \ldots, x_n$. An evaluation context C is a closed context built from $[\_], C \mid P, P \mid C$, and `new` $a; C$, and not containing the state construct $[s \mapsto M]$.

Example 8.1. *As an example of a simple stateful process we can model an RFID tag. The state of the tag is its identity "id" which is stored in a private cell. The tag reads from the cell its identity and sends it to the reader when starting an interaction.*

`new` s; `new` id;
($[s \mapsto id] \mid$ `lock` s; `read` s `as` x; `out`(c, x); `unlock` s)

8.3.2 Operational Semantics

The `StatVerif` semantics is defined over *"stateful"* processes. A `StatVerif` stateful process is a tuple $(\mathcal{E}, \mathcal{S}, \mathcal{P})$, where $\mathcal{E}$ is a finite set of names called environment, the state $\mathcal{S}$ maps state cells to their values, and $\mathcal{P}$ is a finite multiset of pairs of the form (P, λ) where P is a process and λ is the set of the indices of cell names locked by P. In a stateful process $(\mathcal{E}, \mathcal{S}, \mathcal{P})$, a cell index appears in at

most one of the λs. We denote with $locks(\mathcal{P})$ the set of indices of cells locked by the processes in $\mathcal{P}$, i.e. $locks(\mathcal{P}) = \{i | i \in \lambda, (P, \lambda) \in \mathcal{P}\}$. The environment $\mathcal{E}$ must contain at least the free names of $\mathcal{S}$ and $\mathcal{P}$.

Example 8.2. *The stateful process*
$(\{a_1, \ldots, a_l\} \cup \{fn(P_i) | 1 \leq i \leq m\} \cup fn(\{s_i | 1 \leq i \leq n\}) \cup \{fn(M_i) | 1 \leq i \leq n\}, \mathcal{S}, \{(P_1, \emptyset), \ldots, (P_m, \emptyset)\})$
intuitively corresponds to the process
$\texttt{new}\ a_1, \ldots, a_l; ([s_1 \mapsto M_1] \mid \cdots \mid [s_n \mapsto M_n] \mid P_1 \mid \cdots \mid P_m)$ where $\mathcal{S}(s_i) = M_i,\ i \in \{1, \ldots, n\}$.

The semantics of `StatVerif` is defined by a reduction relation $\rightarrow$ on stateful processes, shown in Figure 8.2. It extends the semantics of [ARR11] to take into account equational theories. Auxiliary rules define term evaluation $\Downarrow_\Sigma$ as in [BAF05]. We write $\rightarrow^*$ for the reflexive and transitive closure of $\rightarrow$.

Notice that the reduction relation preserves the invariant that at most one of the processes in P can have a given cell name locked. The cell name s is added to λ by lock, and only one process $(P, \lambda) \in \mathcal{P}$ can satisfy $s \in \lambda$. If a process has locked a cell, the other running processes cannot use the cell until the corresponding unlock. $s_1, \ldots, s_n := M_1, \ldots, M_n$ and $\texttt{read}\ s_1, \ldots, s_n\ \texttt{as}\ x_1, \ldots, x_n$ update and read the store $\mathcal{S}$ in the expected way.

A stateful context C is a stateful process with holes $(\mathcal{E}\_, \mathcal{S}\_, \mathcal{P}\_)$. Let $A = (\mathcal{E}_a, \mathcal{S}_a, \mathcal{P}_a)$ be a closed stateful process, and $C = (\mathcal{E}\_, \mathcal{S}\_, \mathcal{P}\_)$ be a stateful context such that $bn(A) \cap (bn(C) \cup fn(\mathcal{S}) \cup fn(\mathcal{P})) = dom(\mathcal{S}) \cap dom(\mathcal{S}_a) = \emptyset$. The application of the stateful context C to A is the stateful process $C[A] = (\mathcal{E} \cup \mathcal{E}_a, \mathcal{S} \cup \mathcal{S}_a, \mathcal{P} \cup \mathcal{P}_a)$. An evaluation stateful context is a stateful context that does not have its own cells $C = (\mathcal{E}_c\_, \_, \mathcal{P}_c\_)$ and $\mathcal{P}_c$ does not contain the state construct.

term evaluation
$M \Downarrow_\Sigma M$
$\texttt{eval}\ h(D_1, \ldots, D_n) \Downarrow_\Sigma \sigma N$
  if $h(N_1, \ldots, N_n) \to N \in \texttt{def}_\Sigma(h)$, and
σ is such that $\forall\, i,\ D_i \Downarrow_\Sigma M_i$ and $\Sigma \vdash M_i = \sigma N_i$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(0, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P}) \qquad \text{(Red Nil)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!P, \emptyset)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P \mid !P, \emptyset)\}) \qquad \text{(Red Repl)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P \mid Q, \emptyset)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \emptyset), (Q, \emptyset)\}) \qquad \text{(Red Par)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{new}\ a; P, \lambda)\}) \to (\mathcal{E} \cup \{a'\}, \mathcal{S}, \mathcal{P} \cup \{(P\{^{a'}/_{a}\}, \lambda)\}) \quad \text{if } a' \text{ fresh} \qquad \text{(Red New)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ P\ \texttt{else}\ Q, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P\{^{M}/_{x}\}, \lambda)\}) \quad \text{if } \exists M \text{ such that } D \Downarrow_\Sigma M \qquad \text{(Red Fun1)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ P\ \texttt{else}\ Q, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(Q, \lambda)\}) \quad \text{if } \nexists M \text{ such that } D \Downarrow_\Sigma M \qquad \text{(Red Fun2)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{out}(M, N); P, \lambda_1), (\texttt{in}(M', x); Q, \lambda_2)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \lambda_1), (Q\{^{N}/_{x}\}, \lambda_2)\}) \quad \text{if } \Sigma \vdash M = M' \qquad \text{(Red Comm)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{([s \mapsto M], \emptyset)\}) \to (\mathcal{E}, \mathcal{S} \cup \{s \mapsto M\}, \mathcal{P}) \quad \text{if } s \in dom(\mathcal{E}) \text{ and } s \notin dom(\mathcal{S}) \qquad \text{(Red State)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \lambda \cup \{i_1, \ldots, i_m\})\}) \quad \text{if } \forall\, (Q, \lambda') \in \mathcal{P}.\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset \qquad \text{(Red Lock)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{unlock}\ s_{i_1}, \ldots, s_{i_m}; P, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \lambda \setminus \{i_1, \ldots, i_m\})\}) \qquad \text{(Red Unlock)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_{i_1}, \ldots, x_{i_n}; P, \lambda)\}) \to (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P\{^{\mathcal{S}(s_{i_1})}/_{x_{i_1}}, \ldots, ^{\mathcal{S}(s_{i_m})}/_{x_{i_m}}, \lambda)\}) \quad \text{if } s_{i_1}, \ldots, s_{i_m} \in dom(\mathcal{S}) \text{ and } \forall\, (Q, \lambda') \in \mathcal{P}.\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset \qquad \text{(Red Read)}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(s_{i_1}, \ldots, s_{i_m} := M_{i_1}, \ldots, M_{i_m}; P, \lambda)\}) \to (\mathcal{E}, \mathcal{S} \cup \{[s_{i_j} \mapsto M_{i_j} \mid 1 \le j \le m]\}, \mathcal{P} \cup \{P, \lambda)\}) \quad \text{if } s_{i_1}, \ldots, s_{i_m} \in dom(\mathcal{S}) \text{ and } \forall\, (Q, \lambda') \in \mathcal{P}.\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset \qquad \text{(Red Assign)}$$

Figure 8.2: StatVerif semantics

8.3.3 Observational Equivalence

We introduce the definition of observational equivalence for stateful processes.

Definition 8.1. A stateful process A emits on M, denoted $A \downarrow_M$, if and only if $A = C[(\mathcal{E}, \mathcal{S}, \{(\mathtt{out}(M', N); R, \lambda)\})]$ for some evaluation stateful context $C = (\mathcal{E}_C \_, \_, \mathcal{P}_C \_)$ that does not bind any name in $fn(M)$ and $\Sigma \vdash M = M'$.

Definition 8.2. Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ on closed stateful processes such that $A \mathcal{R} B$ implies:

1. $locks(A) = locks(B)$ and $cells(A) = cells(B)$
2. if $A \rightarrow^* A'$, $A' \downarrow_M$ then $B \rightarrow^* B'$, $B' \downarrow_M$;
3. if $A \rightarrow^* A'$ then $B \rightarrow^* B'$ and $A' \mathcal{R} B'$ for some B';
4. $C[A] \mathcal{R} C[B]$ for all evaluation stateful contexts C.

This definition is similar to the observational equivalence definition given in [BAF05] but introduces some constraints specific to stateful processes as proposed in [ALRR14]. In particular, we require the set of locked cells in the two processes to be the same, as well we require the set of free cells in the two processes to be the same. In fact, the presence of free cells gives the attacker the possibility to tell apart two processes just by holding the lock on the free cell, as described in the following example.

Example 8.3. *Let*

$$\begin{aligned} A &= (\emptyset, \{s \mapsto 0\}, \{(\overline{c}\langle b\rangle, \emptyset)\}) \\ B &= (\emptyset, \{s \mapsto 0\}, \{(\mathtt{read}\ s\ \mathtt{as}\ x.\overline{c}\langle b\rangle, \emptyset)\}) \end{aligned}$$

We have that A and B are not observationally equivalent in fact the attacker represented by the evaluation context $C = (\_, \_, \{(0, \{s\})\}, \_)$ can lock the unlocked

public state cells. By holding the lock the adversary blocks the process B which will never output on the channel c while A can execute the output of b on the channel c. Hence $C[A] \Downarrow_c$ but $C[B] \not\Downarrow_c$.

8.4 StatVerif Extension to Observational Equivalence

We extend the syntax of `StatVerif` with a construct that allows to represent pairs of processes having the same structure but possibly differing by the terms and term evaluations that they contain. Such a pair of processes is called a biprocess and was first introduced in [BAF05]. As in [BAF05], the syntax of Figure 8.1 is enriched so that $\texttt{diff}[M, M]$ is a term and $\texttt{diff}[D, D]$ is a term evaluation. Contexts may contain `diff` terms and `diff` term evaluations. We call plain terms, processes and contexts, that do not contain `diff`. Given a biprocess P, the process $\texttt{fst}(P)$ is the process obtained by replacing all occurrences of $\texttt{diff}[M, M']$ with M and $\texttt{diff}[D, D']$ with D. Similarly, the process $\texttt{snd}(P)$ is obtained by replacing $\texttt{diff}[M, M']$ with M' and $\texttt{diff}[D, D']$ with D'; $\texttt{fst}(D)$, $\texttt{fst}(M)$, $\texttt{snd}(D)$, and $\texttt{snd}(M)$ are defined recursively. We extend the biprocess construct to stateful processes. A stateful biprocess is a tuple $(\mathcal{E}, \mathcal{S}, \mathcal{P})$, where the environment $\mathcal{E}$ is a finite set of names that contains at least the free names of $\mathcal{S}$ and $\mathcal{P}$. The state $\mathcal{S}$ maps state cells to their values, and values can be terms containing the `diff` construct. $\mathcal{P}$ is a finite multiset of pairs of the form (P, λ) where P is a biprocess and λ is the set of cell indices locked by P. As before, in a stateful process $(\mathcal{E}, \mathcal{S}, \mathcal{P})$, a cell name appears in at most one of the λs. Given a stateful biprocess A, the stateful process $\texttt{fst}(A)$ is the stateful process obtained by replacing all occurrences of $\texttt{diff}[M, M']$ with M and $\texttt{diff}[D, D']$ with D in (P, λ) for all $(P, \lambda) \in \mathcal{P}$ and in V_i for all $s_i \mapsto V_i \in \mathcal{S}$. Similarly, the stateful process $\texttt{snd}(P)$ is obtained by replacing $\texttt{diff}[M, M']$ with M' and $\texttt{diff}[D, D']$ with D' in (P, λ) for all $(P, \lambda) \in \mathcal{P}$ and in

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ P\ \texttt{else}\ Q, \lambda)\}) \rightarrow (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P\{^{\texttt{diff}[M,M']}/_{x}\}, \lambda)\}) \quad \text{if } \exists M, M' \text{ such that } \texttt{fst}(D) \Downarrow_{\Sigma} M \text{ and } \texttt{snd}(D) \Downarrow_{\Sigma} M' \quad \textsc{Red Fun1}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ P\ \texttt{else}\ Q, \lambda)\}) \rightarrow (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(Q, \lambda)\}) \quad \text{if } \nexists M, M' \text{ such that } \texttt{fst}(D) \Downarrow_{\Sigma} M \text{ and } \texttt{snd}(D) \Downarrow_{\Sigma} M' \quad \textsc{Red Fun2}$$

$$(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{out}(M, N); P, \lambda_1), (\texttt{in}(M', x); Q, \lambda_2)\}) \rightarrow (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \lambda_1), (Q\{^{N}/_{x}\}, \lambda_2)\}) \quad \text{if } \Sigma \vdash \texttt{fst}(M) = \texttt{fst}(M') \text{ and } \Sigma \vdash \texttt{snd}(M) = \texttt{snd}(M') \quad \textsc{Red Comm}$$

FIGURE 8.3: RED FUN1, RED FUN2 and RED COMM reduction rules for `StatVerif` for observational equivalence

V_i for all $s_i \mapsto V_i \in \mathcal{S}$. The semantics of `StatVerif` for observational equivalence is defined as before but ranges over stateful biprocess and the RED FUN1, RED FUN2 and COMM reduction rules are now defined as in Figure 8.3.

Definition 8.3. Let A be a closed stateful biprocess. We say that A satisfies observational equivalence when $\texttt{fst}(A) \approx \texttt{snd}(A)$.

Reductions for stateful biprocesses always imply reductions of the first and second components *i.e.* if $A \rightarrow B$ then $\texttt{fst}(A) \rightarrow \texttt{fst}(B)$ and $\texttt{snd}(A) \rightarrow \texttt{snd}(B)$, however the converse is not always true. When the converse is true we say that the stateful biprocess is *uniform*. As in [BAF05] we want to show that if a stateful biprocess A is uniform then $\texttt{fst}(A) \approx \texttt{snd}(A)$. First, we extend the definition of uniform biprocess given in [BAF05] to stateful biprocesses, then we prove that uniformity implies observational equivalence (Theorem 8.5) and establish the necessary condition for uniformity (and hence observational equivalence, Corollary 8.6).

Definition 8.4. We say that the stateful biprocess A is uniform when $\texttt{fst}(A) \rightarrow B_1$ implies that $A \rightarrow B$ for some stateful biprocess B with $\texttt{fst}(B) = B_1$, and symmetrically for $\texttt{snd}(A) \rightarrow B_2$.

Example 8.4. *Let*

$A = (\emptyset, \{s \mapsto 0\}, \{(\texttt{if}\ a = \texttt{diff}[a, b]\texttt{then}\ P\texttt{else}\ P, \emptyset)\})$ *and*

$B = (\emptyset, \{s \mapsto 0\}, \{(P, \emptyset)\})$.

We have that A is not uniform because $\texttt{fst}(A) \rightarrow B$ *and* $\texttt{snd}(A) \rightarrow B$ *but* $A \not\rightarrow B$.

Theorem 8.5. *Let A_0 be a closed stateful biprocess. If for all plain evaluation stateful contexts C and reductions $C[A_0] \to^* A$, the stateful biprocess A is uniform, then A_0 satisfies observational equivalence.*

A proof of theorem 8.5 is given in Appendix C

Corollary 8.6. *Let A_0 be a closed stateful biprocess. Suppose that, for all plain evaluation stateful contexts C, all evaluation contexts C', and all reductions $C[A_0] \to^* A$,*

1. *if $A = C'[(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup$*
   $\{(\mathbf{out}(N, M); P, \emptyset), (\mathbf{in}(N', x); Q, \emptyset)\})]$, then
   $\Sigma \vdash \mathbf{fst}(N) = \mathbf{fst}(N')$ if and only if $\Sigma \vdash \mathbf{snd}(N) = \mathbf{snd}(N')$,
2. *if $A = C'[(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{let}\ x = D\ \mathtt{in}\ P\ \mathtt{else}\ Q, \lambda)\})]$, then*
   there exists M_1 such that $\mathbf{fst}(D) \Downarrow M_1$ if and only if there exists M_2 such that $\mathbf{snd}(D) \Downarrow M_2$.

Then A_0 satisfies observational equivalence.

A proof of Corollary 8.6 is given in Appendix C.2.

As in [BAF05], we consider an auxiliary rewriting system on terms, T, that defines partial normal forms. The terms manipulated by T do not contain `diff`, but they may contain variables. The rules of T do not contain names and do not have a single variable on the left-hand side. We say that a term is irreducible by T when none of the rewrite rules of T applies to it; we say that the set of terms $\mathcal{M}$ is in normal form relatively to T and Σ, and write $\mathtt{nf}_{T,\Sigma}(\mathcal{M})$, if and only if all terms of $\mathcal{M}$ are irreducible by T and, for all subterms N_1 and N_2 of terms of $\mathcal{M}$, if $\Sigma \vdash N_1 = N_2$ then $N_1 = N_2$. Intuitively, we allow for the possibility that terms may have several irreducible forms requiring that $\mathcal{M}$ uses irreducible forms consistently. This requirement implies, for instance, that if the rewrite

rule $f(x,x) \to x$ applies modulo the equational theory to a term $f(N_1, N_2)$ then N_1 and N_2 are identical and the rule $f(x,x) \to x$ also applies without invoking the equational theory. We extend the definition of $\mathtt{nf}_{T,\Sigma}()$ to sets of processes: $\mathtt{nf}_{T,\Sigma}(\mathcal{P})$ if and only if the set of terms that appear in processes in $\mathcal{P}$ is in normal form.

We recall that as in [BAF05], for an equational theory (without equations) Σ' of a signature E', the evaluation on open terms is defined as a relation $D \Downarrow' (M, \sigma)$, where σ collects instantiations of D obtained by unification:

$$M \Downarrow' (M, \emptyset)$$

$$\begin{aligned}&\mathtt{eval}\ h(D_1, \ldots, D_n) \Downarrow' (\sigma_u N, \sigma_u \sigma')\\ &\quad \text{if } (D_1, \ldots, D_n) \Downarrow' ((M_1, \ldots, M_n), \sigma'),\\ &\quad h(N_1, \ldots, N_n) \to N \text{ is in } \mathtt{def}_{\Sigma'}(h) \text{ and}\\ &\quad \sigma_u \text{ is a most general unifier of } (M_1, N_1), \ldots, (M_n, N_n)\end{aligned}$$

$$\begin{aligned}&(D_1, \ldots, D_n) \Downarrow' ((\sigma_n M_1, \ldots, \sigma_n M_{n-1}, M_n), \sigma_n \sigma)\\ &\quad \text{if } (D_1, \ldots, D_{n-1}) \Downarrow' ((M_1, \ldots, M_{n-1}), \sigma) \text{ and}\\ &\quad \sigma D_n \Downarrow' (M_n, \sigma_n)\end{aligned}$$

As in [BAF05], we let $\mathtt{addeval}(M_1, \ldots, M_n)$ be the tuple of term evaluations obtained by adding $\mathtt{eval}$ before each function symbol of $M_1, \ldots, M_n$. Using these definitions, we describe when an equational theory Σ' with rewrite rules models another equational theory Σ with equations.

Definition 8.7. Let E and E' be signatures on the same function symbols. We say that the equational theory, without equations, Σ' of E' models the equational theory Σ of E if and only if

1. The equational theory of E' is syntactic equality: $\Sigma' \vdash M = N$ if and only if $M = N$.

2. The constructors of E' are the constructors of E; their definition $\mathtt{def}_{\Sigma'}(f)$ contains the rule $f(x_1, \ldots, x_n) \rightarrow f(x_1, \ldots, x_n)$, plus perhaps other rules such that there exists a rewriting system T on terms that satisfies the following properties:

   S1. If $M \rightarrow N$ is in T, then $\Sigma \vdash M = N$.

   S2. If $\mathtt{nf}_{T,\Sigma}(\mathcal{M})$, then for any term M there exists M' such that $\Sigma \vdash M = M'$ and $\mathtt{nf}_{T,\Sigma}(\mathcal{M} \cup \{M'\})$.

   S3. If $f(N_1, \ldots, N_n) \rightarrow N$ is in $\mathtt{def}_{\Sigma'}(f)$, then $\Sigma \vdash f(N_1, \ldots, N_n) = N$.

   S4. If $\Sigma \vdash f(M_1, \ldots, M_n) = M$ and $\mathtt{nf}_{T,\Sigma}(\{M_1, \ldots, M_n, M\})$, then there exist σ and $f(N_1, \ldots, N_n) \rightarrow N$ in $\mathtt{def}_{\Sigma'}(f)$ such that $M = \sigma N$ and $M_i = \sigma N_i$ for all $i \in \{1, \ldots, n\}$.

3. The destructors of E' are the destructors of E, with a rule $g(M'_1, \ldots, M'n) \rightarrow M'$ in $\mathtt{def}_{\Sigma'}(g)$ for each $g(M_1, \ldots, M_n) \rightarrow M$ in $\mathtt{def}_{\Sigma}(g)$ and each $\mathtt{addeval}(M_1, \ldots, M_n, M) \Downarrow' ((M'_1, \ldots, M'_n, M'), \sigma)$.

In the rest of this work, we assume that E' (without equations) models E. The equational theory without equations Σ' of the signature E' can be obtained using one of the algorithms presented in [BAF05]. The equality modulo Σ is extended to stateful biprocesses and term evaluations: $\Sigma \vdash A = A'$ if and only if A can be obtained from A' by replacing some of its subterms M (not containing `diff` or `eval`) with subterms equal modulo Σ and $\Sigma \vdash \mathcal{S}(s_i) = \mathcal{S}'(s_i) \, \forall i \in \{1, \ldots, n\}$ where $\mathcal{S}$ and $\mathcal{S}'$ are the mappings from cells to their values in A and A' respectively. We define $\Sigma \vdash D = D'$ similarly. Let $\mathcal{P}$ and $\mathcal{P}'$ be multisets of pairs (P, λ), we write $\Sigma \vdash \mathcal{P} = \mathcal{P}'$ when $\Sigma \vdash \overline{P} = \overline{Q}$ where $\overline{P} = \{P | (P, \lambda) \in \mathcal{P}\}$, $\overline{Q} = \{Q | (Q, \lambda) \in \mathcal{P}'\}$. We define $A \rightarrow_{\Sigma,\Sigma'} A'$ as $A \rightarrow_{\Sigma} A'$ except that the equational theory Σ' is used for reduction rules (Comm) and (Red Fun1), while the equational theory Σ is still used for (Red Fun2).

An *initial* stateful biprocess A is a stateful biprocess of the form $(\mathcal{E}, \emptyset, \{(\texttt{new}\ \tilde{m};$ $([s_1 \mapsto M_1] \mid \cdots \mid [s_n \mapsto M_n] \mid P), \emptyset)\})$ such that: P has no state declarations in it; each name and variable is bound at most once in P; and each name and variable in P is either bound or free but not both. The tuple $\tilde{m}$ contains cell names and ordinary names. Note that any process with a bounded number of cell names can be converted into one of the prescribed form.

We say that a stateful biprocess A is unevaluated when every term in A (a part from cell names in `lock`, `unlock`, `read`, and assignment statements) is either a variable or $\texttt{diff}[a, a]$ for some name a. Hence, every function symbol in A must be in a term evaluation and prefixed by `eval`. For any stateful biprocess A, we can build an unevaluated stateful biprocess $\texttt{uneval}(A)$ by introducing a term evaluation for every non-trivial term and a `diff` for every name (with $A \approx \texttt{uneval}(A)$).

The following Lemma states that the transitions of the original process are preserved by its unevaluated version.

Lemma 8.8. *Let A_0 be a closed, unevaluated stateful biprocess. If $A_0 \to^*{}_{\Sigma} A'_0$, $\Sigma \vdash A'_0 = A'$, and $\texttt{nf}_{S,\Sigma}(A')$, then $A_0 \to^*{}_{\Sigma',\Sigma} A'$. Conversely, if $A_0 \to^*{}_{\Sigma',\Sigma} A$ then there exists A'_0 such that $\Sigma \vdash A'_0 = A'$ and $A_0 \to^*{}_{\Sigma} A'_0$.*

Hence, as stated in the next Lemma, we can consider unevaluated processes to check the conditions of Corollary 8.6.

Lemma 8.9. *A closed stateful biprocess A_0 satisfies the conditions of Corollary 8.6 if and only if, for all plain evaluation contexts C, all evaluation contexts C', and all reductions $\texttt{uneval}(C[A_0]) \to^* A$, we have:*

1. *if $A = C'[(\mathcal{E}_a, \mathcal{S}_a, \mathcal{P}_a$*
   $\cup\{(\texttt{out}(N, M); Q, \lambda_1), (\texttt{in}(N', x); R, \lambda_2)\})]$ *and*
   $\texttt{fst}(N) = \texttt{fst}(N')$, *then $\Sigma \vdash \texttt{snd}(N) = \mathit{snd}(N')$*

2. *if* $A = C'[(\mathcal{E}_a, \mathcal{S}_a, \mathcal{P}_a \cup \{(\texttt{let}\ x = D\ \texttt{in}\ Q\ \texttt{else}\ R, \lambda)\})]$ *and* $\texttt{fst}(D) \Downarrow_{\Sigma'} M_1$ *for some* M_1*, then* $\texttt{snd}(D) \Downarrow_{\Sigma} M_2$ *for some* M_2

as well as the symmetric properties where we swap `fst` *and* `snd`*.*

A proof of Lemma 8.8 and Lemma 8.9 is given in Appendix C.4 and C.5 respectively.

Thanks to Lemma 8.9 we can concentrate on unevaluated biprocesses and check if condition 1 and 2 are satisfied. If so the stateful biprocess satisfies observational equivalence.

However, the `ProVerif` tool requires our stateful biprocesses to be translated into Horn clauses in order to verify them. In the next section, we describe the translation procedure from unevaluated biprocesses into Horn clauses. The translation into Horn clauses is such that if either condition 1 or condition 2 is not satisfied then the fact *bad* is derivable. This means that our translation is sound and is formalised by Theorem 8.11.

8.5 Clause Generation

In the following sections we describe how to translate an applied pi-calculus process into a set of horn clauses which can be verified by the `ProVerif` tool, or better an extension of it programmed to take into account stateful processes.

Given a closed initial stateful biprocess A_0, `StatVerif` builds a set of Horn clauses. In the clauses, terms are represented by patterns, with the following grammar:

$p ::=$	patterns
x, y, z, i	variable
$f(p_1, \ldots, p_n)$	constructor
$a[p_1, \ldots, p_n]$	name
$\mathtt{g}$	element of GVar

A fresh session identifier variable i is assigned to each replication of A_0. A distinct value for i will be used for each copy of the replicated process (this value is recorded in the pattern sequence used to parametrise bound names). To each name a of A_0 we assign a pattern $a[p_1, \ldots, p_n]$ *i.e.* each name a is a function of the patterns $p_1, \ldots, p_n$. If a is bound by a restriction `new` a in A_0 then $p_1, \ldots, p_n$ are the patterns for the results of term evaluations, the session identifiers of replications, the terms received as inputs and the terms read from states in the context that encloses the restriction. If a is a free name the pattern of a is $a[\,]$. We assume that each restriction `new` a in A_0 has a different name a, distinct from any free name of A_0. Moreover, session identifiers enable us to distinguish names created in different copies of processes. Hence, each name created in the process calculus is represented by a different pattern in the verifier.

Patterns include an infinite set of constants $GVar$. These constants are basically universally quantified variables, and occur only in arguments of the predicate $nounif$, defined in Definition 8.10 below. We write $GVar(M)$ for the term obtained from M by replacing the variables of M with new constants in the set $GVar$. Clauses are built from the following predicates:

$F ::=$	facts
$att(\tilde{s_1}, p, \tilde{s_2}, p')$	attacker knowledge
$msg(\tilde{s_1}, p_1, p_2, \tilde{s_2}, p'_1, p'_2)$	output message
	p2 on p1 (resp. p2' on p1')
$input(\tilde{s_1}, p, \tilde{s_2}, p')$	input on p (resp. p')
$nounif(p, p')$	impossible unification
bad	bad

The predicate $att(\tilde{s_1}, p, \tilde{s_2}, p')$ means that the attacker may obtain p in $\texttt{fst}(P)$ with state $\tilde{s_1}$ and p' in $\texttt{snd}(P)$ in the state $\tilde{s_2}$ by the same operations; $msg(\tilde{s_1}, p_1, p_2, \tilde{s_2}, p'_1, p'_2)$ means that message p_2 can be sent on channel p_1 in $\texttt{fst}(P)$ in the state $\tilde{s_1}$ and that message p'_2 can be sent on channel p'_1 in $\texttt{snd}(P)$ in the state $\tilde{s_2}$ after the same reductions; $input(\tilde{s_1}, p, \tilde{s_2}, p')$ means that an input may be executed in the state $\tilde{s_1}$ on channel p in $\texttt{fst}(P)$ and in the state $\tilde{s_2}$ on channel p' in $\texttt{snd}(P)$; $nounif(p, p')$ means that p and p' cannot be unified modulo Σ by substituting elements of $GVar$ with patterns; finally, *bad* serves in detecting violations of observational equivalence: when *bad* is not derivable, we have observational equivalence. The predicate *nounif* is defined as in [BAF05]:

Definition 8.10. Let p and p' be closed patterns. The fact $nounif(p, p')$ holds if and only if there is no closed substitution σ with domain $GVar$ such that $\Sigma \vdash \sigma p = \sigma p'$.

The *nounif* predicate is not implemented by clauses but by special simplification steps in the `ProVerif` solver.

8.5.1 Clauses for the Protocol

The translation $[\![P]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ of a biprocess P is a set of clauses, where:

- ρ is an environment, that is a function mapping names and variables of the process language to pair of patterns of the clause language;
- ω_1 and ω_2 are sequences of patterns. They accumulates the set of variables that have been input or read so far and the session identifiers respectively for the components $\texttt{fst}(Q)$ and $\texttt{snd}(Q)$ of the biprocess Q being translated. This set is used to parametrise the names created by `new`. The empty sequence is written $\emptyset$; the concatenation of a pattern p to the sequence ω is written $p :: \omega$.
- H is a sequence obtained by conjunction of facts;
- φ_1 is a tuple of terms $(M_1, \ldots, M_n)$, not containing `diff`, representing the last known values of the state cells in the component $\texttt{fst}(Q)$ of the biprocess Q being translated, similarly φ_2 is a tuple of terms not containing `diff`, $(M'_1, \ldots, M'_n)$ representing the last known values of the state cells in the component $\texttt{snd}(Q)$ of the biprocess Q being translated.
- We store in λ the set of the indices of the state cells locked by the currently processed thread.

We write $\varphi[k]$ for the $k-th$ element of the tuple corresponding to the term stored in the cell s_k, and $\varphi[k \mapsto M]$ for the assignment of the term M to the $k-th$ element of the tuple. We extend ρ to a substitution i.e. $\rho(f(M_1, \ldots, M_n)) = (f(p_1, \ldots, p_n), f(p'_1, \ldots, p'_n))$ where $\rho(M_i) = (p_i, p'_i)$ for all $i \in \{1, \ldots, n\}$. We denote by $\rho_1(M)$ and $\rho_2(M)$ the left and the right components of the pair $\rho(M)$, respectively.
We let $\rho(\texttt{diff}[M, M']) = (\rho_1(M), \rho_2(M'))$ and
$\rho_1(\varphi_1) = (\rho_1(M_1), \ldots, \rho_1(M_n))$,
$\rho_2(\varphi_2) = (\rho_2(M_1), \ldots, \rho_2(M_n))$

Let $A_0 = (\mathcal{E}, \emptyset, \{(\texttt{new}\ \tilde{m}; ([s_1 \mapsto M_1] \mid \cdots \mid [s_n \mapsto M_n] \mid P), \emptyset)\})$ be a closed initial stateful biprocess and let $\rho 0$ be the function $\{a \to a[], s_i \to s_i[] | a \in fn(P), 1 \leq i \leq n\}$ and let

$\varphi 0_1 = (\texttt{fst}(M_1), \ldots, \texttt{fst}(M_n))$,
$\varphi 0_2 = (\texttt{snd}(M_1), \ldots, \texttt{snd}(M_n))$. The stateful biprocess above is translated into the union of the following sets of clauses:

- clauses corresponding to the translation of the protocol $[\![P]\!]\ \rho 0\ \emptyset\ \emptyset\ \mathit{true}\ \varphi 0_1\ \varphi 0_2\ \emptyset$ where the function $[\![]\!]\rho H \omega_1 \omega_2 \varphi_1 \varphi_2 \lambda$ is given in Figure 8.4, 8.5;
- clauses encoding the capabilities of the attacker defined in Section 8.5.2;
- clauses encoding the mutability of public states defined in Section 8.5.3;

The `StatVerif` compiler that performs the translation maintains the variables $\rho, H, \omega_1, \omega_2, \varphi_1\varphi_2$ and λ.

8.5.2 Clauses for the Attacker

The following clauses represent the capabilities of the attacker.

The attacker knows all the free names of A_0, rule (Rinit), can generate fresh names, rule (Rn).

$$\begin{array}{ll} \text{For each } a \in fn(A_0), & \\ att(\rho 0_1(\varphi 0_1), a[], \rho 0_2(\varphi 0_2), a[]); & \text{(Rinit)} \\ \text{For some } b \text{ that does not occur in } A_0, & \\ att(\rho 0_1(\varphi 0_1), b[x], \rho 0_2(\varphi 0_2), b[x]) & \text{(Rn)} \end{array}$$

$$
\begin{array}{lcl}
[\![0]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & \emptyset \\
[\![!Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\emptyset & = & [\![Q]\!]\rho(i :: \omega_1)(i :: \omega_2)H\varphi_1\varphi_2\emptyset \quad i \text{ fresh variable} \\
[\![Q_1 \mid Q_2]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\emptyset & = & [\![Q_1]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\emptyset \ \cup \ [\![Q_2]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\emptyset \\
[\![\mathtt{new}\ a; Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & [\![Q]\!]\rho \cup \{a \mapsto (a[\omega_1], a[\omega_2])\}\omega_1\omega_2 H\varphi_1\varphi_2\lambda \\
[\![\mathtt{in}(M, x); Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & \{H \to input(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)))\} \\
& \cup & [\![Q]\!]\rho'(x' :: \omega_1)(x'' :: \omega_2)H'\varphi'_1\varphi'_2\lambda \\
& & \quad \text{let } \{y_k, y'_k, y''_k | 1 \le k \le n, k \notin \lambda\} \text{ be fresh variables} \\
& & \quad \varphi'_1 = \varphi_1[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
& & \quad \varphi'_2 = \varphi_2[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
& & \quad \rho' = \rho \cup \{x \mapsto (x', x'') \mid x', x'' \text{ fresh}\} \\
& & \qquad \cup \{y_k \mapsto (y'_k, y''_k) \mid 1 \le k \le n\} \\
& & \quad H' = (H \wedge msg(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), x', \\
& & \qquad \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)), x'')) \\
[\![\mathtt{out}(M, N); Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & \{H \to msg(\rho_1(\varphi_1), \rho_1(\mathtt{fst}(M)), \rho_1(\mathtt{fst}(N)), \\
& & \quad \rho_2(\varphi_2), \rho_2(\mathtt{snd}(M)), \rho_2(\mathtt{snd}(N)))\} \\
& \cup & [\![Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda \\
[\![\mathtt{let}\ x = D & & \\
\quad \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & \{[\![Q_1]\!](\sigma\rho \cup \{x \mapsto (p, p')\})(p :: \sigma\omega)(p' :: \sigma\omega')(\sigma H) \\
& & (\sigma\varphi_1)(\sigma\varphi_2)\lambda \mid (\rho_1(\mathtt{fst}(D)), \rho_2(\mathtt{snd}(D))) \Downarrow' ((p, p'), \sigma)\} \\
& \cup & [\![Q_2]\!]\rho\omega_1\omega_2 H'\varphi_1\varphi_2\lambda \\
& & \quad H' = H \wedge \rho_1(fails(\mathtt{fst}(D))) \wedge \rho_2(fails(\mathtt{snd}(D))) \\
& \cup & \{\sigma H \wedge \sigma\rho_2(fails(\mathtt{snd}(D))) \to bad | \\
& & \qquad \rho_1(\mathtt{fst}(D)) \Downarrow' (p, \sigma)\} \\
& \cup & \{\sigma H \wedge \sigma\rho_1(fails(\mathtt{fst}(D))) \to bad | \\
& & \qquad \rho_2(\mathtt{snd}(D)) \Downarrow' (p', \sigma)\} \\
& & \quad \text{where } fails(D) =_{\sigma | D \Downarrow' (p,\sigma)} \bigwedge nounif(D, GVar(\sigma D))
\end{array}
$$

FIGURE 8.4: Translation of the protocol: null, replication, parallel, restriction, input, output and let

The attacker can read and write on channels that he knows:

$$msg(xs, y, z, xs', y', z') \wedge att(xs, y, xs', y') \to att(xs, z, xs', z') \qquad (\text{RL})$$

$$att(xs, y, xs', y') \wedge att(xs, z, xs', z') \to msg(xs, y, z, xs', y', z') \qquad (\text{RS})$$

$$att(xs, y, xs', y') \to input(xs, y, xs', y') \qquad (\text{RI})$$

$$
\begin{array}{lcl}
[\![\texttt{lock } s_{i_1},\ldots,s_{i_m};Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & [\![Q]\!]\rho'\omega_1\omega_2 H\varphi_1'\varphi_2'\lambda' \\
 & & \quad \text{let } \{y_k, y_k', y_k'' | 1 \le k \le n, k \notin \lambda\} \text{ be fresh variables} \\
 & & \quad \varphi_1' = \varphi_1[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \varphi_2' = \varphi_2[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \rho' = \rho \cup \{y_k \mapsto (y_k', y_k'') \mid 1 \le k \le n, k \notin \lambda\} \\
 & & \quad \lambda' = \lambda \cup \{i_1,\ldots,i_m\} \\
[\![\texttt{unlock } s_{i_1},\ldots,s_{i_m};Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & [\![Q]\!]\rho'\omega_1\omega_2 H\varphi_1'\varphi_2'\lambda' \\
 & & \quad \text{let } \{y_k, y_k', y_k'' | 1 \le k \le n, k \notin \lambda\} \text{ be fresh variables} \\
 & & \quad \varphi_1' = \varphi_1[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \varphi_2' = \varphi_2[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \rho' = \rho \cup \{y_k \mapsto (y_k', y_k'') \mid 1 \le k \le n, k \notin \lambda\} \\
 & & \quad \lambda' = \lambda \setminus \{i_1,\ldots,i_m\} \\
[\![\texttt{read } s_{i_1},\ldots,s_{i_m} \texttt{ as} & & \\
\quad x_1,\ldots,x_m;Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & [\![Q]\!]\rho'(\rho_1(\varphi_1'[i_1]) :: \cdots :: \rho_1(\varphi_1'[i_m]) :: \omega_1) \\
 & & \qquad (\rho_2(\varphi_2'[i_1]) :: \cdots :: \rho_2(\varphi_2'[i_m]) :: \omega_2) H\varphi_1'\varphi_2'\lambda \\
 & & \quad \text{let } \{y_k, y_k', y_k'' | 1 \le k \le n, k \notin \lambda\} \text{ be fresh variables} \\
 & & \quad \varphi_1' = \varphi_1[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \varphi_2' = \varphi_2[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \rho' = \rho \cup \{x_j \mapsto \\
 & & \qquad\qquad (\rho_1(\varphi_1'[i_j]), \rho_2(\varphi_2'[i_j])) | 1 \le j \le m\} \\
 & & \qquad \cup \{y_k \mapsto (y_k', y_k'') \mid 1 \le k \le n, k \notin \lambda\} \\
 & & \qquad \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\} \\
 & & \quad H' = H \wedge msg(\rho_1(\varphi_1'), vc', vm', \rho_2(\varphi_2'), vc'', vm'') \\
 & & \quad vc, vc', vc'', vm, vm', vm'' \text{ fresh} \\
[\![s_{i_1},\ldots,s_{i_m} := & & \\
\quad M_1,\ldots,M_m;Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda & = & \{H \wedge msg(\rho_1(\varphi_1'), vc', vm', \rho_2(\varphi_2'), vc'', vm'') \rightarrow \\
 & & \qquad msg(\rho_1(\varphi_1''), vc', vm', \rho_2(\varphi_2''), vc'', vm'')\} \\
 & \cup & \{H \wedge att(\rho_1(\varphi_1'), vm', \rho_2(\varphi_2'), vm'') \rightarrow \\
 & & \qquad att(\rho_1(\varphi_1''), vm', \rho_2(\varphi_2''), vm'')\} \\
 & \cup & [\![Q]\!]\rho'\omega_1\omega_2 H\varphi_1''\varphi_2''\lambda \\
 & & \quad \text{let } \{y_k, y_k', y_k'' | 1 \le k \le n, k \notin \lambda\} \text{ be fresh variables} \\
 & & \quad \varphi_1' = \varphi_1[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \varphi_2' = \varphi_2[k \mapsto y_k | 1 \le k \le n, k \notin \lambda] \\
 & & \quad \rho' = \rho \cup \{y_k \mapsto (y_k', y_k'') \mid 1 \le k \le n, k \notin \lambda\} \\
 & & \qquad \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\} \\
 & & \quad \varphi_1'' = \varphi'[i_j \mapsto \texttt{fst}(M_j) | 1 \le j \le m] \\
 & & \quad \varphi_2'' = \varphi'[i_j \mapsto \texttt{snd}(M_j) | 1 \le j \le m] \\
 & & \quad vc, vc', vc'', vm, vm', vm'' \text{ fresh}
\end{array}
$$

FIGURE 8.5: Translation of the protocol: lock, unlock, read and assign

The attacker can listen on any channel he knows by rule (RL) and can send all the messages he knows on channels he knows by rule (RS). If the attacker knows y (resp. y') then he can try to input on it and hence test if y (resp. y') is a channel used for output, rule (RI).

The attacker can apply functions on terms he knows by rule (RF).

For each function h, for each pair of rewrite rules
$h(M_1, \ldots, M_n) \to M,\ h(M'_1, \ldots, M'_n) \to M'$
in $def_{\Sigma'}(h)$ (after renaming of variables),

$$att(xs, M_1, xs', M'_1) \wedge \cdots \wedge att(xs, M_n, xs', M'_n) \to att(xs, M, xs', M') \qquad \text{(RF)}$$

$$input(xs, y, xs', y') \wedge msg(xs, y, z, xs', w', z') \wedge nounif(y', w') \to bad \qquad \text{(RCOM)}$$

For each destructor g,
for each rewrite rule $g(M_1, \ldots, M_n) \to M \in def_{\Sigma'}(g)$

$$\bigwedge_{g(M'_1,\ldots,M'_n)\to M' \in def_{\Sigma'}(g)} nounif((x'_1, \ldots, x'_n), GVar((M'_1, \ldots, M'_n))) \wedge att(xs, M_1, xs', x_1) \wedge \cdots \wedge att(xs, M_n, xs', x_n) \to bad \qquad \text{(RT)}$$

Moreover, we define the symmetric clauses (RCOM') and (RT') obtained from (RCOM) and (RT) by swapping the first and second arguments of *input* and *att* and the first and third arguments of *msg*. By clauses (RCOM) and (RCOM') the attacker can test when a communication can happen in one variant of the biprocess

but not on the other. This is a violation of condition 1 of Lemma 8.9. Hence, clauses (Rt) and (Rt') check whether the application of a destructor succeeds in one of the variant of the biprocess but not on the other, and hence check that condition 2 of lemma 8.9 (and its converse) is true.

8.5.3 Clauses for the Mutability of Public States

The following clauses take into account the mutability of public states, where n be the number of cells in the process to be translated and $s_i[], s_j[]$ are patterns of some public cell s_i, s_j.

$$att((xs_1, \ldots, xs_n), s_i[], (xs'_1, \ldots, xs'_n), s_j[]) \rightarrow \\ att((xs_1, \ldots, xs_n), xs_i, (xs'_1, \ldots, xs'_n), xs'_j); \\ \text{(Rm1)}$$

$$att((xs_1, \ldots, xs_n), s_i[], (xs'_1, \ldots, xs'_n), s_j[]) \wedge \\ att((xs_1, \ldots, xs_n), y, (xs'_1, \ldots, xs'_n), z) \wedge \\ msg((xs_1, \ldots, xs_n), xc', xm', \\ (xs'_1, \ldots, xs'_n), xc'', xm'') \rightarrow \\ msg((xs_1, \ldots, xs_{i-1}, y, xs_{i+1}, \ldots, xs_n), xc', xm', \\ (xs'_1, \ldots, xs'_{i-1}, z, xs'_{i+1}, \ldots, xs_n), xc'', xm''); \\ \text{(Rm2)}$$

$$att((xs_1, \ldots, xs_n), s_i[], (xs'_1, \ldots, xs'_n), s_j[]) \wedge \\ att((xs_1, \ldots, xs_n), y, (xs'_1, \ldots, xs'_n), z) \wedge \\ att((xs_1, \ldots, xs_n), xm', (xs'_1, \ldots, xs'_n), xm'') \rightarrow \\ att((xs_1, \ldots, xs_{i-1}, y, xs_{i+1}, \ldots, xs_n), xm', \\ (xs'_1, \ldots, xs'_{i-1}, z, xs'_{i+1}, \ldots, xs_n), xm''); \\ \text{(Rm3)}$$

RM1 states that the adversary can always read the value of a public cell. Rule RM2 (resp. RM3) states that if the adversary knows a message was sent on a channel (resp. knows a value) at a given state than he can propagate this knowledge to a new state where he modifies the content of a public cell.

8.5.4 Soundness

The set of clauses corresponding to an initial stateful biprocess A_0 is

$$\begin{aligned} \mathcal{C}_{A_0} &= [\![\texttt{uneval}(P)]\!]\rho 0\emptyset\emptyset H\varphi 0_1\varphi 0_2\emptyset \\ &\cup \{(\text{RINIT}), (\text{RN}), \ldots, (\text{RT}), \\ &\quad (\text{RT'}), (\text{RM1}), (\text{RM2}), (\text{RM3})\} \end{aligned}$$

The soundness of our translation is stated by the following theorem.

Theorem 8.11. *If bad is not a logical consequence of $\mathcal{C}_{A_0}$ then A_0 satisfies observational equivalence.*

Proof. To prove that if bad is not derivable from $\mathcal{C}_{A_0}$ then the condition of Lemma 8.9 are satisfied, we use the technique developed in [AB05, Bla02], i.e. we use a type system to express the invariant that correspond to the soundness of the clauses and a subject reduction property to show that the invariant is indeed preserved.

We first define instrumented biprocesses in which we associate a pattern with each name. Let $A_0 = (\mathcal{E}, \emptyset, \{(\texttt{new}\ \tilde{m}; ([s_1 \mapsto M_1] \mid \cdots \mid [s_n \mapsto M_n] \mid P), \emptyset)\})$ be a closed initial stateful biprocess with pairwise distinct bound variables. The instrumented stateful biprocess $instr(A)$ is obtained by replacing all $!Q \in P$ with $!^i Q$ where i is a fresh variable, and all $\texttt{new}\ a; Q, \in P$ with $\texttt{new}\ a : \ a_0[x_1, \ldots, x_n]; Q$ where a_0 is a function symbol and it is not subject to α-conversion and $x_1, \ldots, x_n$ are variables and session identifiers bound above $\texttt{new}\ a$ in $instr(A)$. We denote with $delete(A)$ the stateful biprocess obtained by erasing the instrumentation from an

$$\begin{aligned}
[\![!^i Q]\!]\rho H\varphi_1\varphi_2\emptyset &= [\![Q]\!](\rho[i \mapsto (i,i)])H\varphi_1\varphi_2\emptyset \\
&\quad \text{where } i \text{ is a fresh variable} \\
[\![\texttt{new}\ a : a[x_1,\ldots,x_n]; Q]\!]\rho H\varphi_1\varphi_2\lambda &= [\![Q]\!](\rho[a \mapsto (a[\rho_1(x_1),\ldots,\rho_1(x_n)], \\
&\quad a[\rho_2(x_1),\ldots,\rho_2(x_n)])])\ H\varphi_1\varphi_2\lambda
\end{aligned}$$

FIGURE 8.6: Translation of the instrumented protocol

instrumented stateful biprocess A. Let Λ be a countable set of constant session identifiers, intuitively Λ is the set of session identifiers not yet used in the reduction of A. The semantics of the (RED REPL) rule for instrumented stateful biprocesses is defined as follows:

$$\begin{aligned}
&\Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!^i Q, \emptyset)\}) \rightarrow \\
&\quad \Lambda \setminus \{l\}; (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!^i Q \mid Q\{^l/_i\}, \emptyset)\}) \text{ if } l \in \Lambda
\end{aligned}$$

This rule takes one of the fresh session identifiers in Λ and uses it for the new copy of Q. Note that i is a fresh variable. The other rules for the semantics of instrumented stateful biprocesses are lifted from $A \rightarrow A'$ to $\Lambda; A \rightarrow \Lambda; A'$. Instrumented biprocesses include by construction the variables that were collected by ω_1 and ω_2 in the definition of the translation $[\![]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$. Hence, the clauses $[\![\texttt{uneval}(P)]\!]\rho 0\emptyset\emptyset\emptyset\varphi 0_1\varphi 0_2\emptyset$ can be computed from $instr(\texttt{uneval}(P))$ as follows: $[\![\texttt{uneval}(P)]\!]\rho 0\emptyset\emptyset\emptyset\varphi 0_1\varphi 0_2\emptyset = [\![instr(\texttt{uneval}(P))]\!]\rho 0\emptyset\varphi 0_1\varphi 0_2\emptyset$ where the translation function is defined as in Figure 8.6 for what concerns the translation of $!^i Q$ and of $\texttt{new}\ a : a[x_1,\ldots,x_n]; Q$ and is defined as in Figure 8.4, 8.5 but ignoring ω_1, and ω_2 in the other cases.

Let C be a closed stateful context. For each reduction $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma,\Sigma'} A$ there is a reduction $\Lambda_0; instr(\texttt{uneval}(C[A_0])) \rightarrow \Lambda; A'$ such that $delete(A') = A$ and conversely. Let $A'_0 = instr(\texttt{uneval}(A_0))$ there exists an unevaluated evaluation context C' such that $\texttt{diff}$ occurs only in terms $\texttt{diff}[a, a]$ for some name a in C' and $instr(\texttt{uneval}(C[A_0])) = C'[A'_0]$. Let $\mathcal{C}_{C'A'_0}$ be the set of clauses obtained

by adding to $\mathcal{C}_{A_0}$ the clauses

$$att(xs, a[x_1, \ldots, x_n], xs', a[x_1, \ldots, x_n]) \quad \text{(Rn')}$$

such that either $\texttt{new}\ a : a[x'_1, \ldots, x'_n]$ occurs in C'
or $n = 0, a \in fn(C'), a \notin fn(A'_0)$.
The fact *bad* is derivable from $\mathcal{C}_{C'A'_0}$ if and only if is derivable from $\mathcal{C}_{A_0}$ since we can replace all patterns $a[\ldots]$ of names created by the context C' with patterns $b[i]$, and as long as different names have different images then we can replace the clause Rn' with clause Rn. Hence the definition of $\mathcal{C}_{A_0}$ is sufficient to represent the facts derivable from $[\![instr(\texttt{uneval}(C[A_0]))]\!]$.

Let $\mathcal{S} = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$ we define the ordered representation of $\mathcal{S}$ to be $\bar{\mathcal{S}} = (M_1, \ldots, M_n)$, we denote with $\bar{\mathcal{S}}_1, \bar{\mathcal{S}}_2$ the first and the second projection of the ordered representation of $\mathcal{S}$, respectively, *i.e.*
$\bar{\mathcal{S}}_1 = (\texttt{fst}(M_1), \ldots, \texttt{fst}(M_n)), \bar{\mathcal{S}}_2 = (\texttt{snd}(M_1), \ldots, \texttt{snd}(M_n))$. We say that a states $\mathcal{R}$ is a predecessor of the states $\mathcal{S}$ and we write $\mathcal{R} \leq \mathcal{S}$ if

- $\forall\ M, N, M', N'\ msg(\rho_1(\bar{\mathcal{R}}_1), M, N, \rho_2(\bar{\mathcal{R}}_2), M', N')$
  $\in \mathcal{F}_A \Rightarrow msg(\rho_1(\bar{\mathcal{S}}_1), M, N, \rho_2(\bar{\mathcal{S}}_2), M', N') \in \mathcal{F}_A$ where M, N, M', N' are patterns;
- $\forall\ M, M'\ att(\rho_1(\bar{\mathcal{R}}_1), M, \rho_2(\bar{\mathcal{R}}_2), M') \in \mathcal{F}_A \Rightarrow$
  $att(\rho_1(\bar{\mathcal{S}}_1), M, \rho_2(\bar{\mathcal{S}}_2), M') \in \mathcal{F}_A$ where M, M' are patterns; and
- $\forall\ M, M'\ input(\rho_1(\bar{\mathcal{R}}_1), M, \rho_2(\bar{\mathcal{R}}_2), M') \in \mathcal{F}_A \Rightarrow$
  $input(\rho_1(\bar{\mathcal{S}}_1), M, \rho_2(\bar{\mathcal{S}}_2), M') \in \mathcal{F}_A$ where M, M' are patterns

We define a type system. Types are pairs of closed patterns, E is a function from variables and names to patterns and it is extended to terms as a substitution, so that term M has type $E(M)$. We write $(E, \mathcal{S}, \lambda) \vdash Q$ when Q is well typed with respect to the environment E the states $\mathcal{S}$ and the set of locked cells λ. Let $\mathcal{F}_{C'A'_0}$ be

the set of closed facts derivable from $\mathcal{C}_{C'A'_0}$. Let $\mathcal{S}0 = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$ be the initial state (i.e. the mapping between cells and their initial values), let $E0$ be the environment such that $fn(A_0) \cup fn(Att) = dom(E0)$, $E0(b) = (b[x], b[x])\ \forall\ b \in fn(A_0) \cup fn(Att)$. The type system is defined by the rules given in Figure 8.7.

Let $M_1, \ldots, M_n$ be a sequence of terms and variable or constant session identifiers as found in labels of restriction, we define $Last(M_1, \ldots, M_n) = M_i$ if M_i is the last session identifier appearing in $M_1, \ldots, M_n$, $Last(M_1, \ldots, M_n) = 0$ if no session identifier appears in $M_1, \ldots, M_n$. We define $Label(P)$ as follows:
$Label((\texttt{new}\ a : a_0[M_1, \ldots, M_n]); P) = \{(a_0, Last(M_1, \ldots, M_n))\} \cup Label(P)$,
$Label(!^i P) = 0$,
$Label(P) = \bigcup_{P' \text{subprocess of } P} Label(P')$ and
$Label(\mathcal{P}) = \bigcup_{(P',\lambda)\in\mathcal{P}} Label(P')$.
Let E be a mapping from names and variables to closed patterns, we define $Label(E) = \{(a_0, Last(M_1, \ldots, M_n)) | a \mapsto a_0[M_1, \ldots, M_n] \in E\}$, $Label(\Lambda) = \{(a, l) | l \in \Lambda\}$. We say that $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ is well-labelled when the multisets $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates (E_1 and E_2 are the left and right projection of E respectively). $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ when $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ is well-labelled and for all $(P, \lambda) \in \mathcal{P}$, P is well-typed i.e. $(E, \mathcal{S}, \lambda) \vdash P$. Showing that $Label(E_1)$ and $Label(E_2)$ contain no duplicates guarantees that different terms have different types. More precisely, if E maps names to closed patterns $a[\ldots]$, E is extended to terms as a substitution, and $Label(E)$ contains no duplicates, then we have the following properties:

E1. E is an injection (if $E(M) = E(N)$ then $M = N$) and also an injection modulo Σ (if $\Sigma \vdash E(M) = E(N)$ then $\Sigma \vdash M = N$).

E2. Let N be a term not containing names; if $E(M)$ is an instance of N , then M is an instance of N ; if $E(M)$ is an instance of N modulo Σ, then M is an instance of N modulo Σ.

E3. If $D \Downarrow_{\Sigma'} M$, then $E(D) \Downarrow_{\Sigma'} E(M)$. (This is proved by induction on D.)

$$\frac{}{(E,\mathcal{S},\lambda)\vdash 0}\ T_{nil} \qquad \frac{\forall\ l,\ (E[i\mapsto(l,l)],\mathcal{S},\emptyset)\vdash Q}{(E,\mathcal{S},\emptyset)\vdash !^i Q}\ T_{repl}$$

$$\frac{(E,\mathcal{S},\emptyset)\vdash Q_1 \quad (E,\mathcal{S},\emptyset)\vdash Q_2}{(E,\mathcal{S},\emptyset)\vdash Q_1 \mid Q_2}\ T_{par}$$

$$\frac{(E[a\mapsto(a_0[E_1(M_1),\ldots,E_1(M_n)],a_0[E_2(M_1),\ldots,E_2(M_n)])],\mathcal{S},\lambda)\vdash Q}{(E,\mathcal{S},\lambda)\vdash \texttt{new}\ a:a_0[M_1,\ldots,M_n];Q}\ T_{new}$$

$$\frac{\begin{array}{c}\forall\mathcal{R}\ (\mathcal{S}\leq\mathcal{R}\ \wedge\ \mathcal{R}=\mathcal{R}[j\mapsto\mathcal{S}(j)\mid\ j\in\lambda]\ \wedge\\ \forall\ p_1,p_2\ msg(E_1(\bar{\mathcal{R}}_1),E_1(\texttt{fst}(M)),p_1,E_2(\bar{\mathcal{R}}_2),E_2(\texttt{snd}(M)),p_2)\in\mathcal{F}\\ input(E_1(\bar{\mathcal{R}}_1),E_1(\texttt{fst}(M)),E_2(\bar{\mathcal{R}}_2),E_2(\texttt{snd}(M)))\in\mathcal{F})\\ (E[x\mapsto(p1,p2)],\mathcal{R},\lambda)\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{in}(M,x);Q}\ T_{in}$$

$$\frac{\begin{array}{c}msg(E_1(\bar{\mathcal{S}}_1),E_1(\texttt{fst}(M)),E_1(\texttt{fst}(N)),E_2(\bar{\mathcal{S}}_2),E_2(\texttt{snd}(M)),E_2(\texttt{snd}(N)))\in\mathcal{F}\\ (E,\mathcal{S},\lambda)\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{out}(M,N);Q}\ T_{out}$$

$$\frac{\begin{array}{c}(\forall\ p_1,p_2\ E_1(\texttt{fst}(D))\Downarrow_{\Sigma'} p_1 \text{ and } E_2(\texttt{snd}(D))\Downarrow_{\Sigma'} p_2\\ (E[x\mapsto(p1,p2)],\mathcal{S},\lambda)\vdash Q_1)\\ (\text{if } \nexists\, p_1,\ E_1(\texttt{fst}(D))\Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ E_2(\texttt{snd}(D))\Downarrow_{\Sigma'} p_2 \text{ then } (E,\mathcal{S},\lambda)\vdash Q_2)\\ (\text{if } \exists\, p_1,\ E_1(\texttt{fst}(D))\Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ E_2(\texttt{snd}(D))\Downarrow_{\Sigma'} p_2 \text{ then } bad\in\mathcal{F})\\ (\text{if } \nexists\, p_1,\ E_1(\texttt{fst}(D))\Downarrow_{\Sigma'} p_1 \text{ and } \exists\, p_2,\ E_2(\texttt{snd}(D))\Downarrow_{\Sigma'} p_2 \text{ then } bad\in\mathcal{F})\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{let}\ x=D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2}\ T_{let}$$

$$\frac{\begin{array}{c}\forall\mathcal{R}\ (\mathcal{S}\leq\mathcal{R}\ \wedge\ \mathcal{R}=\mathcal{R}[j\mapsto\mathcal{S}(j)\mid\ j\in\lambda]\\ (E,\mathcal{R},\lambda\cup\{i_1,\ldots,i_m\})\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{lock}\ s_{i_1},\ldots,s_{i_m};Q}\ T_{lock}$$

$$\frac{\begin{array}{c}\forall\mathcal{R}\ (\mathcal{S}\leq\mathcal{R}\ \wedge\ \mathcal{R}=\mathcal{R}[j\mapsto\mathcal{S}(j)\mid\ j\in\lambda]\\ (E,\mathcal{R},\lambda\setminus\{i_1,\ldots,i_m\})\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{unlock}\ s_{i_1},\ldots,s_{i_m};Q}\ T_{unlock}$$

$$\frac{\begin{array}{c}\forall\mathcal{R}\ (\mathcal{S}\leq\mathcal{R}\ \wedge\ \mathcal{R}=\mathcal{R}[j\mapsto\mathcal{S}(j)\mid\ j\in\lambda]\\ (E\cup\{x_k\mapsto(E_1(\texttt{fst}(\mathcal{R}[i_k])),E_2(\texttt{snd}(\mathcal{R}[i_k])))\mid 1\leq k\leq m\},\mathcal{R},\lambda)\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash \texttt{read}\ s_{i_1},\ldots,s_{i_m}\ \texttt{as}\ x_1,\ldots,x_m;Q}\ T_{read}$$

$$\frac{\begin{array}{c}\forall\mathcal{R}\ (\mathcal{S}\leq\mathcal{R},\mathcal{R}=\mathcal{R}[k\mapsto\mathcal{S}(k)\mid k\in\lambda]\wedge\\ \mathcal{R}\leq\mathcal{R}'=\mathcal{R}[j_k\mapsto M_k],1\leq k\leq m),\\ (E,\mathcal{R}[j_k\mapsto M_k],\lambda)\vdash Q\end{array}}{(E,\mathcal{S},\lambda)\vdash s_{i_1},\ldots,s_{i_m}:=M_1,\ldots,M_m;Q}\ T_{assign}$$

FIGURE 8.7: Type system

E4. If $\Sigma \vdash D' = E(D)$ and $D' \Downarrow_\Sigma p'$ then there exists M such that $\Sigma \vdash p' = E(M)$ and $D \Downarrow_\Sigma M$. (This is proved by induction on D, using **E2**).

Let $E0 = a \mapsto (a[], a[]) | a \in fn(C'[A'_0])$. Let $\mathcal{S}0 = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$. The type system has the following properties:

1. Typability of the Adversary (this is stated in Lemma C.23 and proved in Appendix C.7.3).
2. Typability of the Protocol (this is stated in Lemma C.24 and proved in Appendix C.7.4).
3. Typability of the Protocol in the Adversarial context (this is stated in Lemma C.25 and proved in Appendix C.7.5).
4. Type preservation with respect to substitutions of terms (this is stated in Lemma C.21 and proved in Appendix C.7.1).
5. Type propagation throughout successor states (this is stated in Lemma C.22 and proved in Appendix C.7.2).
6. Subject reduction, i.e. type preservation under reduction (this is stated in Lemma C.26 and proved in Appendix C.7.6).

Proof of the second hypothesis of Lemma 8.9. Let $A_0 = (\mathcal{E}_0, \mathcal{S}_0, \{(P_0, \emptyset)\})$ and $C = (\mathcal{E}_c, \_, \{(C_0, \emptyset)\})$ We show that if $E_0 \vdash \Lambda_0; (\mathcal{E}_0, \mathcal{S}_0, \{(instr(\texttt{uneval}(C_0 \mid P_0)), \emptyset)\})$ and $bad \notin \mathcal{F}_{CA_0}$ then the second hypothesis of Lemma 8.9 holds. Assume that $(\mathcal{E}_0 \cup E_c, \mathcal{S}_0, \{(\texttt{uneval}(C_0), \emptyset)), (\texttt{uneval}(P_0), \emptyset)\}) \rightarrow^*_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(C_1, \lambda_1)), (\texttt{let } x = D \texttt{ in } Q \texttt{ else } Q', \lambda_2)\})$ and $\exists M_1$ such that $\texttt{fst}(D) \Downarrow_{\Sigma'} M_1$ then
$\Lambda_0; (\mathcal{E}_0 \cup E_c, \mathcal{S}_0, \{instr(\texttt{uneval}(C_0 \mid P_0)), \emptyset)\}) \rightarrow^*_{\Sigma',\Sigma}$
$\Lambda; (\mathcal{E}, \mathcal{S}, \{(C'_1, \lambda_1), (\texttt{let } x = D \texttt{ in } Q_1 \texttt{ else } Q'_1, \lambda_2)\})$ where $delete(C'_1) = C_1$ and $delete(\texttt{let } x = D \texttt{ in } Q_1 \texttt{ else } Q'_1) = \texttt{let } x = D \texttt{ in } Q \texttt{ else } Q'$.
By hypothesis $E_0 \vdash \Lambda_0; (\mathcal{E}_0, \mathcal{S}_0, \{(instr(\texttt{uneval}(C_0 \mid P_0)), \emptyset)\})$ and by subject

reduction
$E_0 \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(C'_1, \lambda_1), (\texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q'_1, \lambda_2)\})$ hence $(E_0, \mathcal{S}, \lambda_2) \vdash \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q'_1$. Since $\texttt{fst}(D) \Downarrow_{\Sigma'} M_1$, by property **E3** we have that $E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} E_1(M_1)$ since $(E_0, \mathcal{S}, \lambda_2) \vdash \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q'_1$ has been derived by rule $\mathcal{T}_{let}$ and $bad \notin \mathcal{F}_{C'A_0}$ we have that $\exists p_2$ such that $E_2(\texttt{snd}(D)) \Downarrow_{\Sigma} p_2$ and by property **E4** $\exists M_2$ such that $\texttt{snd}(D) \Downarrow_{\Sigma} M_2$. Therefore the second hypothesis of Lemma 8.9 holds.

Proof of the first hypothesis of Lemma 8.9. Assume that $(\mathcal{E}_0 \cup E_c, \mathcal{S}_0, \{(\texttt{uneval}(C_0), \emptyset)\}), (\texttt{uneval}(P_0), \emptyset)\}) \rightarrow^*_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{C_1, \lambda_1)\}), (\texttt{out}(M, N); Q \mid \texttt{in}(M', x); Q', \lambda_2)\})$ and $\texttt{fst}(M) = \texttt{fst}(M')$. As above, there exists E such that $(E, \mathcal{S}, \lambda_2) \vdash \texttt{out}(M, N); Q_1 \mid \texttt{in}(M', x); Q'_1$. Since $(E, \mathcal{S}, \lambda_2) \vdash \texttt{out}(M, N); Q_1 \mid \texttt{in}(M', x); Q'_1$ has been derived by type rules $\mathcal{T}_{par}, \mathcal{T}_{out}, \mathcal{T}_{in}$ we have that
$msg(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(M)), E_1(\texttt{fst}(N)), E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(M')), E_2(\texttt{snd}(N))) \in \mathcal{F}_{C'A_0}$
and
$input(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(M)), E_2(\bar{(\mathcal{S})}_2), E_2(\texttt{snd}(M'))) \in \mathcal{F}_{C'A_0}$. Since $\texttt{fst}(M) = \texttt{fst}(M')$, $E_1(\texttt{fst}(M)) = E_1(\texttt{fst}(M'))$. Since bad is not derivable from $C_{C'A_0}$, $nounif(E_2(\texttt{snd}(M)), E_2(\texttt{snd}(M')))$ does not hold, hence $\Sigma \vdash E_2(\texttt{snd}(M)) = E_2(\texttt{snd}(M'))$. By property **E1**, E_2 is injective modulo Σ and hence $\Sigma \vdash \texttt{snd}(M) = \texttt{snd}(M')$.

The symmetric hypotheses of Lemma 8.9 follows by symmetry. To conclude the proof of Theorem 8.11, we apply Lemma 8.9 and Corollary 8.6 □

8.6 Implementation and Examples

We extended the `ProVerif` tool in order to accept as input stateful protocols, produce the horn clauses according to the translation described in Section 8.5 and finally verify if the equivalence relation is satisfied. In this section, we present a few examples of stateful protocols and equivalence based security properties and

we discuss the results we obtained when using our `ProVerif` extension to verify them. Note that our tool is built onto the typed version of `ProVerif` and hence the code we describe in this session has some type annotation which is not present in the syntax introduced in Section 8.3.1.

We firstly checked our tool against a very simple example. We implement a very simple RFID protocol where the RFID tag reads from the memory its own identity and then sends it in clear over the air to the reader. We ask our tool to verify if this protocol satisfies unlinkability of the tags. As expected, our tool finds a linkability attack which can be found as well by encoding states using private channels and running the `ProVerif` tool.

```
(* State initial value and state declaration *)
free init: bitstring [private].
cell s: bitstring := (init).

(* Public channel *)
channel c.

(* Simple pair of systems consisting in
- one system where RFID tags execute at most once and send
their identity
- one system where RFID tags execute more than once and each time
send their identity
*)

process
    !new id1: bitstring;!(new id2:bitstring;
      lock s;
      s:= choice[id1,id2];
```

```
      read s as x;
      out(c, x);
      unlock s)
```

We next ask the tool to verify the unlinkability of a simple unlinkable RFID protocol. In this protocol RFID tags send a random number and then send the hash of their identity to the reader using the chosen random number to conceal it. Our tool can verify that this protocol satisfies the unlinkability property. `ProVerif` can verify the unlinkability of this protocol as well when using private channels to encode states.

```
 (* State initial value and state declaration *)
free init: bitstring [private].
cell s: bitstring := (init).

(* Public channel *)
channel c.

(* Hash functions *)
fun h(bitstring,bitstring): bitstring.

(* Simple RFID tag unlinkable biprocess consisting in
- one system where RFID tags execute at most once and send
their identity (id2)
- one system where RFID tags execute more than once and each
time send the hash  with a random nonce of their identity (id1)
*)

process
    !new id1: bitstring;
```

```
lock s; s:=id1; unlock s;
!(new id2: bitstring;
  lock s;
  read s as x;
  new r: bitstring;
  out(c, r);
  out(c, (r, h(choice[x, id2],r)));
  unlock s)
```

Another RFID protocol aimed at preserving tags' unlinkability is the OSK protocol. In this case the tag reads its current identifier, x from the memory, uses the function $g()$ to hash it and sends it to the reader. The reader sends the hashed value $g(x)$ to the back-end database in order to check if the tag is a legitimate one and identify it. Finally, the tag updates its identifier by hashing it using the function $h()$ and stores $h(x)$ into the memory.

```
 (* State initial value and state declaration *)
free init: bitstring [private].
cell s: bitstring := (init).


(* Public channel *)
channel c.


(* Hash functions *)
fun g(bitstring): bitstring.
fun h(bitstring): bitstring.

(* Simple pair of systems consisting in
```

```
- one system where an unbounded number of RFID tags execute the OSK
protocol at most once, each tag sends the hash g(id2) of its
identifying token (id2) and then updates the stored identifying token
with the hash h(id2)
- one system where an unbounded number of RFID tags execute the OSK
protocol more than once; each tag reads its currently stored
identifying token x from the memory s and sends the hash g(x) to the
reader finally it updates the identifying token with the hash of the
current identifying token h(x) *)

let Tag =
      new id2:bitstring;
      lock s;
      read s as x;
      out(c,g(choice[x,id2]));
      s:= choice[h(x), h(id2)];
      unlock s.



process
    !(new id1: bitstring;
    lock s;
    s:=id1;
    unlock s;
    !(Tag))
```

Our tool fails to prove that this protocol satisfies unlinkability and reports a false attack. The same attack is reported by `ProVerif` when using private channel to model states. The false attack is caused by the abstraction used in `ProVerif` to generate nonces. In particular, `id1` and `id2` are two nonces generated by the

translation into horn clauses with a different degree of freshness, in fact `id2` is parametrised by both replications while `id1` is parametrised by only the first replication. The same problem arises when verifying our motivating example, the fixed TMSI reallocation procedure.

```
(* Initialisation value *)
free init: bitstring [private].

(* Memory cells *)
cell s: bitstring := (init).
cell s1: bitstring := (init).

(* Symmetric encryption *)
fun senc(bitstring, bitstring, bitstring): bitstring.
fun sdec(bitstring, bitstring): bitstring.
equation forall xk: bitstring, xr:bitstring, xm: bitstring;
sdec(xk, senc(xk, xr, xm)) = xm.

(* Protocol specific constants *)
fun loc_up():bitstring.
fun tmsi_reall_cmd():bitstring.
fun tmsi_reall_compl():bitstring.
fun sn_lai():bitstring.

(* Public channel *)
channel c.

(* Mobile station process representing pair of mobile station
processes. The process using memory cell s executes the TMSI
reallocation protocol many times wile the process using memory
```

```
cell s1 represents multiple mobile stations executing the TMSI
reallocation protocol at most once.
Each MS reads its TMSI from the memory and then sends it over
the air and wait for a reply from the SN. If the reply is a TMSI
reallocation command then the MS decrypts the reallocation
message using the ciphering key ck, retrieves the new TMSI
and stores it in the memory to be used in the next session *)

let MS(ck:bitstring) =
new tmsi2: bitstring;
new mr:bitstring;
lock s;lock s1;
    s1:=tmsi2;
    read s as otmsi1;
    read s1 as otmsi2;
    out(c,(loc_up, choice[otmsi1,otmsi2]));
    in(c,y:bitstring);
    let (msgtype:bitstring, msg:bitstring)=sdec(ck,y) in
       (if msgtype = tmsi_reall_cmd then
          (let (newtmsi:bitstring,lai:bitstring)=msg in
     s:= newtmsi;
     s1:=newtmsi;
          out(c, senc(ck, mr, tmsi_reall_compl));
          unlock s;
          unlock s1)).

(* Serving network process *)

let SN(ck:bitstring) =
```

```
      in(c, z:bitstring);
      new r:bitstring;
      new ntmsi: bitstring;
      out(c, senc(ck, r, (tmsi_reall_cmd,(ntmsi,sn_lai))));
      in(c, w: bitstring).

process
    !(new tmsi1:bitstring;
    lock s;
    s:=tmsi1;
    unlock s;
    !(new ck: bitstring;(MS(ck)|SN(ck))))
```

However, knowing the source of the problem we can encode this protocol keeping in mind that we want our new names `ntmsi` and `tmsi2` to have the same degree of randomness, and we can obtain an encoding of the TMSI reallocation procedure free from the false attack.

```
 (* Initialisation value *)
free init: bitstring [private].

(* Memory cells *)
cell s: bitstring := (init).

(* Symmetric encryption *)
fun senc(bitstring, bitstring, bitstring): bitstring.
fun sdec(bitstring, bitstring): bitstring.
equation forall xk: bitstring, xr:bitstring, xm: bitstring;
                              sdec(xk, senc(xk, xr, xm)) = xm.
```

```
(* Protocol specific constants *)
fun loc_up():bitstring.
fun tmsi_reall_cmd():bitstring.
fun tmsi_reall_compl():bitstring.
fun sn_lai():bitstring.

(* Public channel *)
channel c.

(* Mobile station process representing pair of mobile station
processes.
Each MS reads its TMSI from the memory and then sends it over
the air and wait for a reply from the SN. If the reply is a TMSI
reallocation command then the MS decrypts the reallocation
message using the ciphering key ck, retrieves the new TMSI.
One process (the left one) is a multi session MS and stores the
new TMSI in the memory to be used in the next session,
the other MS process (the right one) is a single session MS and
stores a new id (tmsi2) in the memory this emulate a new MS *)

let MS(ck:bitstring) =
new mr:bitstring;
lock s;
new tmsi2: bitstring;
    read s as otmsi;
    out(c,(loc_up, otmsi));
in(c,y:bitstring);
    let (msgtype:bitstring, msg:bitstring)=sdec(ck,y) in
(if msgtype = tmsi_reall_cmd then
```

```
(let (newtmsi:bitstring,lai:bitstring)=msg in
    s:= choice[newtmsi,tmsi2];
             out(c, senc(ck, mr, tmsi_reall_compl));
             unlock s)).

(* Serving network process *)

let SN(ck:bitstring) =
      new ntmsi: bitstring;
      in(c, z:bitstring);
      new r:bitstring;
      out(c, senc(ck, r, (tmsi_reall_cmd,(ntmsi,sn_lai))));
      in(c, w: bitstring).

process
    !(new tmsi1:bitstring;
    lock s;
    s:=tmsi1;
    unlock s;
    !(new ck: bitstring;(MS(ck)|SN(ck))))
```

The same results can be obtained by encoding states using private channels. As a last example we show how our tool can be used to model and verify the strong secrecy property of a stateful protocol. We model the AKA protocol and we use states to store the currently used ciphering key in order to be able to use it across multiple session to encrypt the communication before a new one is established through the AKA protocol.

```
(* Public communication channel *)
channel c.
```

```
(* State cells *)
free init: bitstring [private].
cell s: bitstring := (init).

(* Constant values *)
fun macFail(): bitstring.
fun syncFail(): bitstring.
fun reject(): bitstring.
fun auth_req(): bitstring.
fun ciph_cmd(): bitstring.
fun ksi(): bitstring.
fun true1(): bitstring.
fun false1(): bitstring.

(* UMTS AKA protocol specific mac and key generation functions *)
fun f1(bitstring, bitstring): bitstring.
fun f2(bitstring, bitstring): bitstring.
fun f3(bitstring, bitstring): bitstring.
fun f4(bitstring, bitstring): bitstring.
fun f5(bitstring, bitstring): bitstring.

(* Symmetric key encryption function *)
fun senc(bitstring, bitstring, bitstring): bitstring.
fun sdec(bitstring, bitstring): bitstring.
equation forall k: bitstring, m: bitstring, r: bitstring;
                        sdec(k, senc(k, r, m)) = m.

(* Mobile station process executing the AKA protocol *)
```

```
let AKA_MS(k: bitstring, osqn: bitstring, x:bitstring) =
    new r_ms: bitstring;
    let (xrand: bitstring, xautn: bitstring) = x in (
    let (msg: bitstring, xmac: bitstring) = xautn in (
    let ak: bitstring = f5(k, xrand) in (
    let xsqn: bitstring = sdec(ak, msg) in (
    let mac: bitstring = f1(k, (xrand, xsqn)) in (
    if xmac = mac then (
       if xsqn = osqn then (
          let res: bitstring = f2(k, xrand) in (
          let ck: bitstring = f3(k, xrand) in (
          let ik: bitstring = f4(k, xrand) in (
          s:=ck;
          out(c, res);
          in(c, xmsg: bitstring)))))
          else (out(c, syncFail)))
    else (out(c, macFail)))))))).

(* Serving network process executing the AKA protocol *)
let AKA_SN(k: bitstring,osqn: bitstring) =
    new rand: bitstring;
    new r_sn: bitstring;
    new s: bitstring;
    new r: bitstring;
    let mac: bitstring = f1(k, (rand, osqn)) in (
    let res: bitstring = f2(k, rand) in (
    let ck: bitstring = f3(k, rand) in (
    let ik: bitstring = f4(k, rand) in (
    let ak: bitstring = f5(k, rand) in (
```

```
    let autn: bitstring = (senc(ak, r_sn, osqn), mac) in (
    let av: bitstring = (rand, res, ck, ik, ak) in (
    out(c, (auth_req, (rand, autn)));
    in(c, xres: bitstring);
    if xres = res then (
       out(c, senc(ck, r, s))
    )
    else (
       out(c, reject))))))))).

(* Mobile station process receiving a message from the SN.
If the message is an AKA request the MS executes the AKA
protocol, if the message is a ciphering mode command it
sends and encrypted message *)
let MS(k: bitstring,osqn: bitstring, secret:bitstring) =
  new r:bitstring;
  (lock s;
     in(c, xmsg:bitstring);
     let (msgtype:bitstring, msg:bitstring)= xmsg in
     (if msgtype = ciph_cmd then
          (
           read s as ck;
           out(c, senc(ck,r, secret))
           )
     else if msgtype = auth_req then AKA_MS(k,osqn,msg))).

(* Serving network process that either starts the AKA protocol
or sends a ciphering mode command *)
let SN(k: bitstring,osqn: bitstring) =
```

```
     ((AKA_SN(k,osqn)|out(c,(ciph_cmd, ksi)))).

process (!new k:bitstring;
        new osqn:bitstring;
        !((MS(k, osqn, choice[true1(),false1()])) | (SN(k,osqn))))
```

Our tool can prove that strong secrecy is preserved. The same result can be obtained by using `ProVerif` and encoding states using private channels.

8.7 Discussion and Future Work

Our `ProVerif` extension is a first step towards the development of a tool to verify stateful process. However, we adopted a very conservative approach which turned out not to be more successful than `ProVerif` itself in the verification of observational equivalence based properties for stateful processes. Our tool suffers from the problems caused by the abstraction of `ProVerif` and exhibit the same false attacks, which are due to the nonce generation abstraction and the consequent lack of freshness. For this reason we are exploring further possible translation into Horn clauses and considering the introduction of explicit state sequentiality facts.

9

Conclusions

In this thesis, we contributed to the formal verification of privacy-related properties using a process calculus, namely the applied pi-calculus, to reason about security protocols. In particular, we focused on strong anonymity and strong unlinkability as defined in [ACRR10]. Our contribution is both practical and theoretical, in fact we applied our methods to real world protocols used in mobile telephony systems. For what concerns our experimental analysis and real world case study, we showed that:

- mobile telephony protocols are vulnerable to several privacy threats;
- these threats lead to attacks that can be mounted in practice at low cost;
- the efficiency of the pseudonym change strategy depends on many factors which the 3GPP standard leaves as implementation choices;
- the implementation choices made by real network operators do not provide a satisfying level of privacy and leave space for different kinds of tracking attacks.

Currently, our demonstration of the attacks presented in Chapter 5 relies on particular hardware/software using closed source implementation of the 3G protocol

stack and radio signalling functions. It would be interesting and beneficial for further research in the area of mobile telephony systems to investigate the possibility of implementing open source testing equipment, such as a 3G base station and mobile phone, using low cost hardware, e.g. USRP, and the GNU radio software. Further work is needed in order to confirm experimentally the replay attack presented in Section 5.1.1. This would allow to check if there are or not mechanisms in place (not stated in the standard) to thwart this attack by preventing replayed messages from being accepted by the Mobile Station. Also, a thorough and methodical analysis of the level of privacy achieved by different privacy policies would be of great interest. However, this would possibly require collecting further data about user mobility, aggregation areas, population density, network coverage and user base per geographical area. The impact of the adoption of the proposed TMSI reallocation policies on the network performances should be studied and related to the level of achieved user's privacy in order to carefully balance these equally important aspects of mobile telephony systems. The overall privacy of mobile telephony systems and at each layer of the protocol stack requires further investigation and would possibly offer interesting challenges for the development of formal methods as well. In particular, the proliferation of location based services makes the analysis of application layer privacy highly desirable from a user's point of view.

The analysis of our case study gave us the drive and motivation to further investigate the development of formal methods for the automatic verification of privacy-related properties and for the automatic verification of equivalence-based properties for stateful protocols. We used formal methods to:

- show that the privacy vulnerabilities that can be found in currently used mobile telephony systems could have been detected at design time using automatic verification tools such as `ProVerif`;

- show that lightweight privacy friendly solutions can be developed and automatically verified. In particular, we show how to build biprocesses for the automatic verification of anonymity and unlinkability using the `ProVerif` tool;

- present one of the few examples in the literature of a proof of labelled bisimilarity of a real-sized protocol. Such manual proofs give useful insights on the way one could automate them, and thus pave the way to automating labelled bisimilarity proofs;

- extend the existing automated tools (`ProVerif` and `StatVerif`) to the verification of equivalence based properties of stateful protocols.

In Chapter 6, we propose solutions to thwart the privacy breaches we found in some of the procedures used by currently deployed mobile telephony systems. These solutions show that privacy friendly measures could be adopted by the next generation of mobile telephony standards while keeping low the computational and economical cost of implementing them. Formal analysis of the protocols under development for the next generation of mobile telephony systems, should be considered as future work and would give stronger guarantees on the security properties of the future mobile communication systems. `StatVerif` and our extension for equivalence based properties ease the reasoning about the security of stateful protocols. However, the tool we implemented inherits `ProVerif` limitations in dealing with the generation of names. In future, we would like to explore alternative translations into Horn clauses and possibly modification of the resolution algorithm in order to overcome these limitations and extend the verification of equivalence-based properties to a larger class of stateful protocols. Furthermore, we would like to investigate other privacy-related properties recurring in pervasive systems. For example, a new primitive "open" (described in [ALRR14]) could be implemented in our `ProVerif` extension, so to take into account corrupted memory cells and enable the modelling of forward privacy properties.

A

ProVerif Code

We report the most relevant parts of the `ProVerif` scripts used for the verification of the fixed protocols. We omit the declaration of constants, any name which is not under the scope of a new statement as public name and hence as part of the adversary knowledge. Note that the identity of the victim mobile for the anonymity property is public.

Fixed IMSI paging procedure in `ProVerif`*.*

```
 1 let PAGING_MS = in(c, x);
 2  let (msgtype, xrand, xblob) = x in (
 3  if msgtype = pagingReq then (
 4     let (xpage, ximsi, =sqn, xchall) =
 5      sdec(f(k, xrand), xblob) in (
 6      if xpage = page then (
 7         if imsi = ximsi then (
 8            out(c, (pagingResp, xchall))))))).
 9 let PAGING_SN = new rand; new chall;
10   new r_sn1; new r_sn2;
11   let UK = f(k, rand) in (
```

```
12   out(c, (pagingReq, rand, senc(UK, r_sn2,
13                  (page, imsi, sqn, chall))));
14   in(c, pres)).
```

Biprocess for unlinkability of IMSI paging.

```
15 process new pvN; let pbN = pub(pvN) in (
16   out(c, pbN);
17   (! (new sk1; new imsi1; new otmsi1;
18   (! (new sk2; new imsi2; new otmsi2; new sqn;
19       let imsi = choice[imsi1, imsi2] in (
20       let k = choice[sk1, sk2] in (
21       let otmsi = choice[otmsi1, otmsi2] in (
22       (PAGING_MS) | (PAGING_SN)))))))))
```

Biprocess for anonymity of IMSI paging.

```
23 process new pvN; let pbN = pub(pvN) in (
24   out(c, pbN);
25   ((! (new k; new imsi; new otmsi;
26               (! ((PAGING_MS) | (PAGING_SN)))))
27 | (new k; new id; new otmsi;
28    let imsi = choice[id, imsi_V] in
29    (! ((PAGING_MS) | (PAGING_SN)))))
```

Fixed AKA procedure in ***ProVerif****.*

```
30 let AKA_MS = new r_ms; in(c, x);
31   let (xrand, xautn) = x in (
32   let (msg, xmac) = xautn in (
```

```
33   let ak = f5(k, xrand) in (
34   let xsqn = sdec(ak, msg) in (
35   let mac = f1(k, (xrand, xsqn)) in (
36   if (xmac, xsqn) = (mac, osqn) then (
37       let res = f2(k, xrand) in (
38       let ck = f3(k, xrand) in (
39       let ik = f4(k, xrand) in (
40       out(c, res);
41       in(c, xmsg)))))
42   else (out(c, aenc(pbN, r_ms,
43              (Fail, imsi, osqn)))))))))).
44 let AKA_SN =
45   new rand; new r_sn; new s; new r;
46   let mac = f1(k, (rand, osqn)) in (
47   let res = f2(k, rand) in (
48   let ck = f3(k, rand) in (
49   let ik = f4(k, rand) in (
50   let ak = f5(k, rand) in (
51   let autn = (senc(ak, r_sn, osqn), mac) in (
52   let av = (rand, res, ck, ik, ak) in (
53   out(c, (rand, autn));
54   in(c, xres);
55   if xres = res then (
56      out(c, senc(ck, r, s)))
57   else (out(c, reject))))))))).
```

Biprocess for unlinkability of AKA.

```
58 process new pvN; let pbN = pub(pvN) in (
59   out(c, pbN);
```

```
60   (! (new sk1; new imsi1;new otmsi1;
61   (! (new sk2; new imsi2; new osqn; new otmsi2;
62       let imsi = choice[imsi1, imsi2] in (
63       let k = choice[sk1, sk2] in (
64       let otmsi = choice[otmsi1,otmsi2] in (
65       (AKA_MS) | (AKA_SN)))))))))
```

Biprocess for anonymity of AKA.

```
66 process new pvN; let pbN = pub(pvN) in (
67   out(c, pbN);
68   ((! (new k; new imsi; new otmsi;
69        (!new osqn;  ((AKA_MS) | (AKA_SN)))))
70  | (new k; new id; new otmsi;
71     let imsi = choice[id, imsi_V] in (
72     !new osqn;  ((AKA_MS) | (AKA_SN))))))
```

Original AKA procedure in ***ProVerif****.* We check the MAC and the sequence number (line 81) in the same conditional statement, so to avoid false attacks due to the evaluation of the conditional. For the same reason we introduce the functions `err` and `geterr` (lines 73-74) to determine the error message (lines 86-87) and avoid the use of an if statement.

```
73 reduc geterr(err(x,z,y,y))=macFail;
74       geterr(err(x,x,y,z))=synchFail.
75 let AKA_MS = new r_ms; in(c, x);
76   let (xrand, xautn) = x in (
77   let (msg, xmac) = xautn in (
78   let ak = f5(k, xrand) in (
79   let xsqn = sdec(ak, msg) in (
```

```
80   let mac = f1(k, (xrand, xsqn)) in (
81   if (xmac, xsqn) = (mac,sqn) then (
82      let res = f2(k, xrand) in (
83      let ck = f3(k, xrand) in (
84      let ik = f4(k, xrand) in (
85      out(c, res); in(c, xmsg)))))
86   else (let err_msg =
87         geterr(err(mac, xmac, sqn, xsqn)) in
88         out(c, err_msg))))))).
```

The following code was used to verify secrecy authentication and integrity properties of the fixed AKA protocol.

```
query attacker: s.      (*secrecy of communication following AKA*)
query attacker: k.       (*secrecy of long term shared key*)
query attacker: f3(k, rand). (*secrecy of session ciphering key*)
query attacker: f4(k, rand). (*secrecy of session integrity key*)
query attacker: imsi.      (*secrecy of long term identity*)

(* verify the mutual authentication property *)
query ev: completedC(x,y)==> evinj: startedS(x,y).
query ev: completedS(x,y)==> evinj: startedC(x,y).
(* For sanity *)
query ev: completedF(x,y)==> evinj: startedAKA(x,y).
(* For sanity *)
query ev: completedAKA(x,y)==> evinj: startedAKA(x,y).

(*verify integrity property *)
query ev: completedI(x)==> evinj: startedI(x).
```

```
(* public communication channel *)
free c.

(* constant values *)
fun Fail/0.
fun reject/0.
fun Sqn/0.

(* UMTS AKA protocol specific mac and key generation functions *)
fun f0/2.
fun f1/2.
fun f2/2.
fun f3/2.
fun f4/2.
fun f5/2.
fun f9/2.

(* symmetric key encryption function *)
fun senc/3.
fun sdec/2.
equation sdec(k, senc(k, r, m)) = m.

(* public key generation function *)
fun pub/1.

(* public key encryption function *)
fun aenc/3.
reduc adec(k, aenc(pub(k), r, m))= m.
```

```
let AKA_MS =
    new r_ms;
    in(c, x);
    event startedC(k, osqn);
    let (xrand, xautn) = x in (
    let (msg, xmac) = xautn in (
    let ak = f5(k, xrand) in (
    let xsqn = sdec(ak, msg) in (
    let mac = f1(k, (xrand, xsqn)) in (
    if (xmac, xsqn) = (mac, osqn) then (
       let res = f2(k, xrand) in (
       let ck = f3(k, xrand) in (
       let ik = f4(k, xrand) in (
       out(c, res);
       event completedAKA(xrand,xautn);
       event completedC(k,xsqn);
       in(c, (ymsg, xmsg));
       if f9(ik, sdec(ck, ymsg)) <> xmsg
       then 0
       else event completedI(sdec(ck,ymsg))))))
    else (
       new rrand;
       new r;
       out(c, aenc(pbN, r_ms, (Fail, imsi, rrand,
                   senc(f0(k, rrand), r, (Sqn, osqn)))));
       event completedF(xrand, xautn))))))).


let AKA_SN =
```

```
    new rand;
    new r_sn;
    new s;
    new r;
    let mac = f1(k, (rand, osqn)) in (
    let res = f2(k, rand) in (
    let ck = f3(k, rand) in (
    let ik = f4(k, rand) in (
    let ak = f5(k, rand) in (
    let autn = (senc(ak, r_sn, osqn), mac) in (
    let av = (rand, res, ck, ik, ak) in (
    event startedAKA(rand, autn);
    event startedS(k, osqn);
    out(c, (rand, autn));
    in(c, xres);
    if xres = res then (
       event completedS(k, osqn);
       event startedI(s);
       out(c, (senc(ck, r, s), f9(ik, s)))
    )
    else (
       out(c, reject))))))))).

let MS =  (AKA_MS).
let SN =  (AKA_SN).

process new pvN; let pbN = pub(pvN) in out(c, pbN);
(! (new k; new imsi; new otmsi;
   (!new osqn;  ((MS) | (SN)))))
```

B

Proof of the Unlinkability of the TMSI Reallocation Procedure

The following Appendices are devoted to the proof of Proposition 1 which establishes that using new session keys at each TMSI reallocation provides unlinkability to MSs. In Appendix B.1 we introduce the necessary definitions and notations. Appendix B.2 provides the proof of Lemma 7.1 which establishes the bisimulation part of Proposition 1, while Appendix B.3 details the proof of Lemma 7.2 which establishes the static part of Proposition 1.

In our quest to understand how proofs of labelled bisimilarity work, and which are the key arguments on which they usually rely, we took the party to detail all the cases even those that look mechanical. This is particularly true for the proof of Lemma 7.1.

B.1 Definitions and Notation

The processes MS and SN model respectively a mobile station and a serving network sharing a private channel dck (the latter models the fact that MS and SN can "securely" establish a shared session key by executing the AKA protocol) are defined as follows:

$$
\begin{aligned}
MS \stackrel{def}{=}\ & \nu\, ck.\nu\, mr.d(x).\overline{up}\langle x\rangle.\overline{dck}\langle ck\rangle.dw(y).\\
& \textit{if}\ \ \mathtt{fst}(\mathtt{sdec}(ck,y)) = \text{TMSI\_REALL}\ \ \textit{then}\\
& \quad \overline{up}\langle \mathtt{senc}(ck, mr,\ \text{COMPLETE})\rangle.\overline{d}\langle \mathtt{snd}(\mathtt{sdec}(ck,y))\rangle\\
& \textit{else}\ 0\\
SN \stackrel{def}{=}\ & \nu\, nid.\nu\, sr.dw(z).dck(xck).\\
& \quad \overline{up}\langle \mathtt{senc}(xck,\ sr,\ \mathtt{pair}(\text{TMSI\_REALL},\ nid))\rangle.dw(w)
\end{aligned}
$$

We also have the following processes, defined earlier:

$$
\begin{aligned}
Init &\stackrel{def}{=} \overline{d}\langle id\rangle\\
SSA &\stackrel{def}{=} \nu\, d.\nu\, id.(Init \mid MS)\\
MSA &\stackrel{def}{=} \nu\, d.\nu\, id.(Init \mid !MS)
\end{aligned}
$$

$Init$ is the memory initialization process, SSA and MSA are respectively a single-session and a multi-session mobile station agent.

Let S and M be two closed processes defined as follows:

$$\begin{aligned} S &\stackrel{def}{=} \nu\, dck.(!(\nu\, d.\nu\, id.(\overline{d}\langle id\rangle \mid MS) \mid !SN)) \\ &\stackrel{def}{=} \nu\, dck.(!SSA \mid !SN) \end{aligned}$$

$$\begin{aligned} M &\stackrel{def}{=} \nu\, dck.(!(\nu\, d.\nu\, id.(\overline{d}\langle id\rangle \mid !MS) \mid !SN)) \\ &\stackrel{def}{=} \nu\, dck.(!MSA \mid !SN) \end{aligned}$$

The process S represents an unbounded number of mobile stations executing the TMSI reallocation procedure at most once. The process M represents an unbounded number of mobile stations which can execute the TMSI reallocation procedure an unbounded number of times.

In the following, the process $MMS^k_{i,j}$ represents the i^{th} mobile station ready to execute the k^{th} step of the j^{th} session of the TMSI reallocation protocol and the process $SMS^k_{i,j}$ represents the $i+j^{th}$ single session mobile station ready to execute the k^{th} step of the TMSI reallocation procedure. They are defined as in Chapter 7.

We define the grouped multi-session system $GMS_{i,j}[\_]$ representing j sessions of the i^{th} mobile station and the simulating grouped single-session system $GSS_{i,j}[\_]$ representing j single session mobile stations simulating the j sessions of the i^{th} mobile station of the multi-session system, as follows:

$$\begin{aligned} GMS_{i,j}[\_] &\stackrel{def}{=} \nu\, \widetilde{ms}_{i,j}.\nu\, \widetilde{nid}_{i,l}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,j-1} \mid \_ \mid !RMS_i) \\ GSS_{i,j}[\_] &\stackrel{def}{=} \nu\, \widetilde{ss}_{i,j}.\nu\, \widetilde{nid}_{i,l}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,j-1} \mid \_) \\ &\quad\ l \in \{j-1, j\} \end{aligned}$$

We define a symmetric relation between the single session and the multiple session system. Let

$$
\begin{aligned}
\mathcal{R} \stackrel{def}{=} \{ \quad & (C,\ D),\ (D,\ C) : \exists\ n,\ m \geq 0, \\
& A \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN), \\
& B \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN), \\
& \text{where } \forall i,\ 1 \leq i \leq n,\ \exists l_i,\ k_{l_i},\ l_i \geq 0, 1 \leq k_{l_i} \leq 8 \text{ such that} \\
& C_i = GSS_{i,l_i}[SMS_{i,l_i}^{k_{l_i}} \mid SSN_{i,l_i}] = \nu\ \widetilde{ss}_{i,l_i}.\nu\ \widetilde{nid}_{i,j}.(SMS_{i,1}^{7} \mid \cdots \mid \\
& \qquad SMS_{i,l_i-1}^{7} \mid SMS_{i,l_i}^{k_{l_i}} \mid SSN_{i,l_i}) \\
& D_i = GMS_{i,l_i}[MMS_{i,l_i}^{k_{l_i}} \mid MSN_{i,l_i}] = \nu\ \widetilde{ms}_{i,l_i}.\nu\ \widetilde{nid}_{i,j}.(MMS_{i,1}^{7} \mid \cdots \mid \\
& \qquad MMS_{i,l_i-1}^{7} \mid MMS_{i,l_i}^{k_{l_i}} \mid MSN_{i,l_i} \mid !RMS_i) \\
& SSN_{i,l_i} = MSN_{i,l_i} = SN_{i,1}^{h_1} \mid \cdots \mid SN_{i,l_i-1}^{h_{l_i-1}} \mid L^{h_{l_i}},\ h_1, \ldots, h_{l_i-1} \geq 2 \\
& L^{h_{l_i}} = \begin{cases} 0 & \text{if } k_{l_i} \in \{1,\ 2\} \\ SN_{i,l_i}^{h_{l_i}} & \text{otherwise} \end{cases} \qquad j = \begin{cases} l_i - 1 & \text{if } L^{h_{l_i}} = 0 \\ l_i & \text{otherwise} \end{cases} \\
& PSN_m = SN_{j_1}^{1} \mid \cdots \mid SN_{j_m}^{1}, \text{ for some } j_1, \ldots,\ j_m \in \{0,\ 1\} \\
\} &
\end{aligned}
$$

and let $SSN_{i,0} = MSN_{i,0} = GSS_{i,0}[\_] = GMS_{i,0}[\_] = SMS_{i,0}^{k} = MMS_{i,0}^{k} = 0$. Moreover, we define the set of $\mathcal{S}$ (resp. $\mathcal{M}$) to be the set of single (resp. multi)-session systems:

$$
\begin{aligned}
\mathcal{S} &\stackrel{def}{=} \{C \mid (C,D) \in \mathcal{R} \wedge \\
&\qquad C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN)\} \\
\mathcal{M} &\stackrel{def}{=} \{D \mid (C,D) \in \mathcal{R} \wedge \\
&\qquad D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)\}
\end{aligned}
$$

B.2 Proof of Lemma 7.1

In order to prove Lemma 7.1 we first establish some auxiliary lemmas. Intuitively, Lemma 7.3 and Lemma 7.4 states that if the single (respectively multi)-session system can do a transition then either one of the grouped single (respectively multi)-session system components did the transition (possibly synchronizing with one of the SN_j^1 components of the PSN_m process (i.e. the MS synchronizes with the SN network, this steps model the establishment of means for ciphering of the TMSI reallocation protocol) or one of the components under replication was unrolled and did the transition. In particular, Lemma 7.3 deals with the possible internal transitions of the single (resp. multi)-session system.

Lemma 7.3. *Let* $C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ *where either* $SA = SSA$ *or* $SA = MSA$ *and either* $\mathcal{X} = \mathcal{S}$ *or* $\mathcal{X} = \mathcal{M}$ *accordingly. We have that if* $C \xrightarrow{\tau} C'$ *then* $C' \equiv \nu\ dck.(C'_1 \mid \cdots \mid C'_{n'} \mid PSN'_{m'} \mid !SA \mid !SN) \in \mathcal{X}$ *and*

- *either* $\exists i\ C_i \xrightarrow{\tau} C'_i \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n \wedge PSN'_{m'} = PSN_m$
- *or* $\exists i\ C_i \mid PSN_m \xrightarrow{\tau} C'_i \mid PSN'_{m'} \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n$
- *or* $n' = n + 1 \wedge C'_j = C'\ \forall j \neq n+1,\ C'_{n+1} = GSS_{n+1,1}[SMS^1_{n+1,1} \mid 0]$ *if* $\mathcal{X} = \mathcal{S}$, $C'_{n+1} = GMS_{n+1,1}[MMS^1_{n+1,1} \mid 0]$ *if* $\mathcal{X} = \mathcal{M}$ *and* $PSN'_{m'} = PSN_m$

Proof. Let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ where $SA = SSA$ and $\mathcal{X} = \mathcal{S}$, by definition we have that $C_i \equiv GSS_i[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}],\ 1 \leq i \leq n,\ PSN_m \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_m}$. If $C \equiv C'' \xrightarrow{\tau} C''' \equiv C'$ then we have that $!SN$ cannot do a silent transition, while $!SSA$ can do a silent transition (case labelled **New MS**), and $C_i \mid PSN_m$ can do a silent transition on the channel dck if $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}],\ k_i = 2,\ m \geq 1$ (case labelled **MS/SN**

synch) and C_i can do a silent transition evaluating the conditional statement if $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}],\ k_i = 4$. Hence we have 3 cases:

(i) **(New MS)** $SSA \equiv \nu\, \widetilde{ss}_{n+1,1}.(Init_{n+1,1} \mid SMS^0_{n+1,1}) \xrightarrow{\tau} \nu\, \widetilde{ss}_{n+1,1}.SMS^1_{n+1,1} \equiv \nu\, \widetilde{ss}_{n+1,1}.(SMS^1_{n+1,1} \mid 0) \equiv GSS_{n+1,1}[SMS^1_{n+1,1} \mid 0] = C'_n + 1$ then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid SSA \mid !SSA \mid !SN) \equiv \nu\, dck.(SSA \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_n + 1 \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid C'_n + 1 \mid PSN_m \mid !SSA \mid !SN) = C'$

(ii) **(MS/SN synch)** Let $1 \leq i \leq n$ such that $C_i \equiv GSS_{i,l_i}[SMS^2_{i,l_i} \mid SSN_{i,l_i}]$ and let $m \geq 1$, we have that $C_i \mid PSN_m \equiv GSS_{i,l_i}[SMS^2_{i,l_i} \mid SSN_{i,l_i}] \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^2_{i,l_i} \mid SSN_{i,l_i-1}) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\ \nu\, nid_{j_h}.\nu\, sr_{j_h}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^2_{i,l_i} \mid SN^1_{j_h} \mid SSN_{i,l_i-1}) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \xrightarrow{\tau} \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\nu\, nid_{j_h}.\nu\, sr_{j_h}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^3_{i,l_i} \mid SSN_{i,l_i-1} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}\}) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i}.(SMS^{k_1}_{i,1} \mid \cdots \mid SMS^{k_{l_i}-1}_{i,l_i-1} \mid SMS^3_{i,l_i} \mid SSN_{i,l_i-1} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}, ^{nid_{i,l_i}}/_{nid_{j_h}}, ^{sr_{i,l_i}}/_{sr_{j_h}}, ^{w_{i,l_i}}/_{w_{j_h}}\}) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \equiv GSS_{i,l_i}[SMS^3_{i,l_i} \mid SSN_{i,l_i}] \mid PSN'_{m-1} = C'_i,\ PSN'_{m-1} = SN^1_{j_1} \mid \cdots \mid SN^1_{j_{h-1}} \mid SN^1_{j_{h+1}} \mid \cdots \mid SN^1_{j_m}$ then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid PSN_m \mid \cdots \mid C_n \mid !SSA \mid !SN) \equiv \nu\, dck.(C_i \mid PSN_m \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid PSN_{m-1} \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_{m-1} \mid !SSA \mid !SN) = C'$

(iii) Let $1 \leq i \leq n$ such that $C_i \equiv GSS_{i,l_i}[SMS^4_{i,l_i} \mid SSN_{i,l_i}]$ we have 2 cases:

– **(Conditional-then)** if $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, y_{i,l_i})) =_E$ TMSI_REALL we have that $C_i \xrightarrow{\tau} GSS_{i,l_i}[SMS^5_{i,l_i} \mid SSN_{i,l_i}] = C'_i$, then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) = C'$

– **(Conditional-else)** if $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, y_{i,l_i})) \neq_E$ TMSI_REALL we have that $C_i \xrightarrow{\tau} GSS_{i,l_i}[SMS^8_{i,l_i} \mid SSN_{i,l_i}] = C'_i$, then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) = C'$

Let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ where $SA = MSA$ and $\mathcal{X} = \mathcal{M}$, by definition we have that $C_i \equiv GMS_i[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}],\ 1 \leq i \leq n,\ PSN_m \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_m}$. If $C \equiv C'' \xrightarrow{\tau} C''' \equiv C'$ then we have that $!SN$ cannot do a silent transition, while $!MSA$ can do a silent transition (case labelled **New MS**), and $C_i \mid PSN_m$ can do a silent transition on the channel *dck* if $C_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}],\ k_i = 2,\ m \geq 1$ (case labelled **MS/SN synch**), and C_i can do a silent transition either evaluating the conditional statement if $C_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}],\ k_i = 4$ or creating a new session (case labelled **New Session**) if $C_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}],\ k = 6$. Hence we have 4 cases:

(i) **(New MS)** $MSA \equiv \nu\, \widetilde{ms}_{n+1,1}.(Init_{n+1,1} \mid MMS^0_{n+1,1}) \xrightarrow{\tau} \nu\, \widetilde{ms}_{n+1,1}.$ $MMS^1_{n+1,1} \equiv \nu\, \widetilde{ms}_{n+1,1}.(MMS^1_{n+1,1} \mid 0) \equiv GMS_{n+1,1}[MMS^1_{n+1,1} \mid 0] = C'_n + 1$ then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid MSA \mid !MSA \mid !SN) \equiv \nu\, dck.(MSA \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_n + 1 \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid$

$!SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid C'_n + 1 \mid PSN_m \mid !MSA \mid !SN) = C'$

(ii) **(MS/SN synch)** Let $1 \leq i \leq n$ such that $C_i \equiv GMS_{i,l_i}[MMS^2_{i,l_i} \mid MSN_{i,l_i}]$ and let $m \geq 1$, we have that $C_i \mid PSN_m \equiv GMS_{i,l_i}[MMS^2_{i,l_i} \mid MSN_{i,l_i}] \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^2_{i,l_i} \mid MSN_{i,l_i-1} \mid !RMS_i) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\nu\, nid_{j_h}.\nu\, sr_{j_h}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^2_{i,l_i} \mid SN^1_{j_h} \mid MSN_{i,l_i-1} \mid !RMS_i) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \xrightarrow{\tau} \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\nu\, nid_{j_h}.\nu\, sr_{j_h}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^3_{i,l_i} \mid MSN_{i,l_i-1} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}\} \mid !RMS_i) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i}.(MMS^{k_1}_{i,1} \mid \cdots \mid MMS^{k_{l_i-1}}_{i,l_i-1} \mid MMS^3_{i,l_i} \mid MSN_{i,l_i-1} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}, ^{nid_{i,l_i}}/_{nid_{j_h}}, ^{sr_{i,l_i}}/_{sr_{j_h}}, ^{w_{i,l_i}}/_{w_{j_h}}\} \mid !RMS_i) \mid SN^1_{j_1} \mid \cdots \mid SN^1_{j_h-1} \mid SN_{j_h+1} \mid \cdots \mid SN^1_{j_m} \equiv GMS_{i,l_i}[MMS^3_{i,l_i} \mid MSN_{i,l_i}] \mid PSN'_{m-1} = C'_i,\ PSN'_{m-1} = SN^1_{j_1} \mid \cdots \mid SN^1_{j_{h-1}} \mid SN^1_{j_{h+1}} \mid \cdots \mid SN^1_{j_m}$ then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid PSN_m \mid \cdots \mid C_n \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid PSN_m \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid PSN_{m-1} \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_{m-1} \mid !MSA \mid !SN) = C'$

(iii) Let $1 \leq i \leq n$ such that $C_i \equiv GMS_{i,l_i}[MMS^4_{i,l_i} \mid MSN_{i,l_i}]$ we have 2 cases:

- **(Conditional-then)** if $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, y_{i,l_i})) =_E$ TMSI_REALL we have that $C_i \xrightarrow{\tau} GMS_{i,l_i}[MMS^5_{i,l_i} \mid MSN_{i,l_i}] = C'_i$, then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) = C'$

– **(Conditional-else)** if $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, y_{i,l_i})) \neq_E$ TMSI_REALL we have that $C_i \xrightarrow{\tau} GMS_{i,l_i}[MMS^8_{i,l_i} \mid MSN_{i,l_i}] = C'_i$, then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) = C'$

(ii) **(New session)** let $1 \leq i \leq n$, $k = 6$ then $C_i \equiv GMS_{i,l_i}[SMS^6_{i,l_i} \mid MSN_{i,l_i}] \equiv \nu\, \widetilde{ms}_{i,l_i}.\widetilde{nid}_{i,l_i}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^6_{i,l_i} \mid MSN_{i,l_i} \mid !RMS_i) \equiv \nu\, \widetilde{ms}_{i,l_i}.\widetilde{ms}_{i,l_i}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^6_{i,l_i} \mid MSN_{i,l_i} \mid \nu\, ck_{i,l_i+1} \nu\, mr_{i,l_i+1}.MMS^0_{i,l_i+1} \mid !RMS_i) \xrightarrow{\tau} \nu\, \widetilde{ms}_{i,l_i+1}.\widetilde{nid}_{i,l_i}.$
$(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^7_{i,l_i} \mid MMS^1_{i,l_i+1} \mid MSN_{i,l_i} \mid !RMS_i) \equiv \nu\, \widetilde{ms}_{i,l_i+1}.\widetilde{nid}_{i,l_i}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i} \mid MMS^1_{i,l_i+1} \mid MSN_{i,l_i} \mid 0 \mid !RMS_i) \equiv GMS_{i,l_i+1}[MMS^1_{i,l_i+1} \mid MSN_{i,l_i+1}] = C'_i$ then $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) = C'$

□

Lemma 7.4 deals with the possible labelled transitions of the single (resp. multi)-session system.

Lemma 7.4. *Let* $C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ *where either* $SA = SSA$ *or* $SA = MSA$ *and either* $\mathcal{X} = \mathcal{S}$ *or* $\mathcal{X} = \mathcal{M}$ *accordingly. We have that if* $C \xrightarrow{\alpha} C'$ *with* $fv(\alpha) \subseteq dom(C)$ *then* $C' \equiv \nu\ dck.(C'_1 \mid \cdots \mid C'_{n'} \mid PSN'_{m'} \mid !SA \mid !SN) \in \mathcal{X}$, *and*

- *either* $\exists i\ C_i \xrightarrow{\alpha} C'_i \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n \wedge PSN'_{m'} = PSN_m$
- *or* $C'_j = C_j\ \forall 1 \leq j \leq n, \wedge n' = n \wedge PSN'_{m'} = PSN_m \mid SN^1_{j_{m+1}},\ m' = m + 1$

Proof. Let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ where $SA = SSA$ and $\mathcal{X} = \mathcal{S}$, by definition we have that $C_i \equiv GSS_i[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}],\ 1 \leq i \leq n,\ PSN_m \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_m}$. If $C \equiv C'' \xrightarrow{\alpha} C''' \equiv C'$ then we have that $!SSA$ and PSN_m cannot do an α-transition, while $!SN$ can do an α-transition (case labelled **SN pre-synch α-transition**), and C_i can do an α-transition if $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}],\ k \in \{1,3,5\}$ (cases labelled **Active session MS α-transition** and **SN post-synch α-transition**). Hence we have 2 cases:

(i) let $1 \leq i \leq n,\ k \in \{1,3,5\}$ then $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}]$ we have 2 cases:

- **(Active session MS α-transition)** if $SMS^{k_i}_{i,l_i}$ does the α-transition we have that $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}] \xrightarrow{\alpha} GSS_{i,l_i}[SMS^{k_i+1}_{i,l_i} \mid SSN_{i,l_i}] = C'_i$ for all $k \in \{1,3,5\}$. Hence $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) = C'$

– **(SN post-synch α-transition)** if SSN_{i,l_i} does the α-transition, let $1 \leq j \leq l_i$ such that $SN_{i,j}^{h_j} \neq 0$ and $2 \leq h_j \leq 3$ we have that $SSN_{i,l_i} \equiv SN_{i,1}^{h_1} \mid \cdots \mid SN_{i,j}^{h_j} \mid \cdots \mid SN_{i,l_i}^{h_{l_i}} \xrightarrow{\alpha} SN_{i,1}^{k_1} \mid \cdots \mid SN_{i,j}^{k_j+1} \mid \cdots \mid SN_{i,l_i}^{h_{l_i}} = SSN'_{i,l_i}$, hence $C_i \equiv GSS_{i,l_i}[SMS_{i,l_i}^{k_i} \mid SSN_{i,l_i}] \equiv GSS_{i,l_i}^{k_i}[SMS_{i,l_i}^{k_i} \mid SSN_{i,l_i}] \xrightarrow{\alpha} GSS_{i,l_i}[SMS_{i,l_i}^{k_i} \mid SSN'_{i,l_i}] = C'_i$. Then we have that $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) = C'$

(ii) **(SN pre-synch α-transition)** $!SN \equiv SN_h^0 \mid !SN \xrightarrow{\alpha} SN_h^1 \mid !SN$ let $PSN_{m+1} = SN_{j_1}^1 \mid \cdots \mid SN_{j_m}^1 \mid SN_h^1$ then we have that $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !U \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid SN_h^0 \mid !SN) \equiv \nu\, dck.(SN_h^0 \mid C_i \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(SN_h^1 \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_{m+1} \mid !SSA \mid !SN) = C'$

Let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN) \in \mathcal{X}$ where $SA = MSA$ and $\mathcal{X} = \mathcal{M}$, by definition we have that $C_i \equiv GMS_i[MMS_{i,l_i}^{k_i} \mid MSN_{i,l_i}]$, $1 \leq i \leq n$, $PSN_m \equiv SN_{j_1}^1 \mid \cdots \mid SN_{j_m}^1$. If $C \equiv C'' \xrightarrow{\alpha} C''' \equiv C'$ then we have that $!MSA$ and PSN_m cannot do an α-transition, while $!SN$ can do an α-transition (case labelled **SN pre-synch α-transition**), and C_i can do an α-transition if $C_i \equiv GMS_{i,l_i}[MMS_{i,l_i}^{k_i} \mid MSN_{i,l_i}]$, $k \in \{1,3,5\}$ (cases labelled **Active session MS α-transition** and **SN post-synch α-transition**). Hence we have 2 cases:

(i) let $1 \leq i \leq n$, $k \in \{1,3,5\}$ then $C_i \equiv GMS_{i,l_i}[MMS_{i,l_i}^{k_i} \mid MSN_{i,l_i}]$ we have 2 cases:

– **(Active session MS α-transition)** if $MMS^{k_i}_{i,l_i}$ does the α-transition we have that $C_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\alpha} GMS_{i,l_i}[MMS^{k_i+1}_{i,l_i} \mid MSN_{i,l_i}] = C'_i$ for all $k \in \{1,3,5\}$. Hence $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) = C'$

– **(SN post-synch α-transition)** if MSN_{i,l_i} does the α-transition, let $1 \leq j \leq l_i$ such that $SN^{h_j}_{i,j} \neq 0$ and $2 \leq h_j \leq 3$ we have that $MSN_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_j}_{i,j} \mid \cdots \mid SN^{h_{l_i}}_{i,l_i} \xrightarrow{\alpha} SN^{k_1}_{i,1} \mid \cdots \mid SN^{k_j+1}_{i,j} \mid \cdots \mid SN^{k_{l_i}}_{i,l_i} = MSN'_{i,l_i}$, hence $C_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\alpha} GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN'_{i,l_i}] = C'_i$. Then we have that $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(C'_i \mid C_1 \mid \cdots \mid C_{i-1} \mid C_{i+1} \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C'_i \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) = C'$

(ii) **(SN pre-synch α-transition)** $!SN \equiv SN^0_h \mid !SN \xrightarrow{\alpha} SN^1_h \mid !SN$ let $PSN_{m+1} = SN^1_{j_1} \mid \cdots \mid SN^1_{j_m} \mid SN^1_h$ then we have that $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid SN^0_h \mid !SN) \equiv \nu\, dck.(SN^0_h \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(SN^1_h \mid C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_{m+1} \mid !MSA \mid !SN) = C'$

□

Intuitively, Lemma 7.1 states that if one of the components of the single-session system can do a transition then the corresponding component of the multi-session

system can do it as well, and vice versa, if a component of the multi-session system can do a transition then the mimicking component of the single-session system can do it as well.

Lemma 7.1. *Let* $C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SA \mid !SN)$, $D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !\overline{SA} \mid !SN)$ *such that* $SA = SSA$ *(resp.* $SA = MSA$*) and* $(\overline{SA} = MSA$ *(resp.* $\overline{SA} = SSA)$ *and* $(C, D) \in \mathcal{R}$
if $C \xrightarrow{\ell} C'$ *with* $fv(\ell) \subseteq dom(C)$ *and* $bn(\ell) \cap fn(D) = \emptyset$ *then* $D \xrightarrow{\ell} D'$ *and* $(C', D') \in \mathcal{R}$ *for any* $\ell \in \{\tau, \alpha\}$.

Proof. Let $(C,\ D) \in \mathcal{R}$ and let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid !PSN_m \mid !SSA \mid !SN)$ and $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid !PSN_m \mid !MSA \mid !SN)$.
If $C \xrightarrow{\tau} C'$ then by Lemma 7.3 we have that $C' \equiv \nu\, dck.(C'_1 \mid \cdots \mid C'_{n'} \mid !PSN'_{m'} \mid !SSA \mid !SN)$ and we have 3 cases:

- $\exists i\ C_i \xrightarrow{\tau} C'_i \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n \wedge PSN'_{m'} = PSN_m$, hence $C_i \equiv GSS_{i,l_i}[SMS^{k_{l_i}}_{i,l_i} \mid SSN_{i,l_i}]$ can do a silent transition this means that C_i is of the form $C_i \equiv GSS_{i,l_i}[SMS^{4}_{i,l_i} \mid SSN_{i,l_i}]$ i.e. we have two cases depending on the evaluation of the conditional statement:

  - **(Conditional-then)**
    if $C_i \equiv GSS_{i,l_i}[SMS^{4}_{i,l_i} \mid SSN_{i,l_i}] \xrightarrow{\tau} GSS_{i,l_i}[SMS^{5}_{i,l_i} \mid SSN_{i,l_i}] = C'_i$ then $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) =_E$ TMSI_REALL i.e.
    $N_{i,l_i} =_E \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))$. By definition of $\mathcal{R}$ we have that $D_i \equiv GMS_{i,l_i}[MMS^{4}_{i,l_i} \mid MSN_{i,l_i}]$, $MMS^{4}_{i,l_i} = MX_{i,l_i} \mid MChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\}$ and
    $MChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\} \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle.\ \overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i}))\rangle$ *else* $0 \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$.

$\overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))\rangle$
else $0 \xrightarrow{\tau} \overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$.
$\overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))))\rangle$ hence $MMS^4_{i,l_i} \xrightarrow{\tau} MMS^5_{i,l_i}$ and $D_i \xrightarrow{\tau} GMS_{i,l_i}[MMS^5_{i,l_i} \mid MSN_{i,l_i}] = D'_i$ then let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

– **(Conditional-else)**
if $C_i \equiv GSS_{i,l_i}[SMS^4_{i,l_i} \mid SSN_{i,l_i}] \xrightarrow{\tau} GSS_{i,l_i}[SMS^8_{i,l_i} \mid SSN_{i,l_i}] = C'_i$ then $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) \neq_E$ TMSI_REALL i.e.
$N_{i,l_i} \neq_E \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))$. By definition of $\mathcal{R}$ we have that $D_i \equiv GMS_{i,l_i}[MMS^4_{i,l_i} \mid MSN_{i,l_i}]$, $MMS^4_{i,l_i} = MX_{i,l_i} \mid MChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\}$ and $MChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\} \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i}))$ $=$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$.
$\overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i}))\rangle$ *else* $0 \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$. $\overline{d_{i,1}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))\rangle$ *else* $0 \xrightarrow{\tau} 0$ hence $MMS^4_{i,l_i} \xrightarrow{\tau} MMS^8_{i,l_i}$ and $D_i \xrightarrow{\tau} GMS_{i,l_i}[MMS^8_{i,l_i} \mid MSN_{i,l_i}] = D'_i$ then let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

- **(MS/SN synch)** $\exists i\ C_i \mid PSN_m \xrightarrow{\tau} C'_i \mid PSN'_{m'} \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n$ the only possible silent transition between C_i and PSN_m is the communication

on the channel dck hence, $C_i \equiv GSS_{i,l_i}[SMS^2_{i,l_i} \mid SSN_{i,l_i}]$ and $PSN'_{m'} \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_m},\ m \geq 1$ if $C_i \mid PSN_m \xrightarrow{\tau} C'_i \mid PSN'_{m'}$ we have that $C'_i \equiv GSS_{i,l_i}[SMS^3_{i,l_i} \mid SSN'_{i,l_i}],\ SSN'_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_{l_i-1}}_{l_i-1} \mid SN^{h_{l_i}}_{i,l_i}$, and $\nu\, nid_{i,l_i}.SN^{h_{l_i}}_{i,l_i} \equiv \nu\, nid_{j_h}.\nu\, sr_{j_h}.SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\}$ for some $1 \leq h \leq m$ such that $SN^1_{j_h}$ occurs in PSN_m and $PSN'_{m'} \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_{h-1}} \mid SN^1_{j_{h+1}} \mid \cdots \mid SN^1_{j_m}$. By definition of $\mathcal{R}$ we have that $D_i \equiv GMS_{i,l_i}[MMS^2_{i,l_i} \mid MSN_{i,l_i}],\ MSN_{i,l_i} \equiv SSN_{i,l_i}$. We have that $D_i \mid PSN_m \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^2_{i,l_i} \mid MSN_{i,l_i} \mid !RMS_i \mid PSN_m) \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^2_{i,l_i} \mid SN^1_{j_h} \mid MSN_{i,l_i} \mid !RMS_i \mid PSN'_{m'}) \xrightarrow{\tau} \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\nu\, nid_{j_h}.\nu\, sr_{j_h}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^3_{i,l_i} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}\} \mid MSN_{i,l_i} \mid !RMS_i \mid PSN'_{m'}) \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,l_i-1} \mid MMS^3_{i,l_i} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\} \mid MSN_{i,l_i} \mid !RMS_i \mid PSN'_{m'}) \equiv GMS_{i,l_i}[MMS^3_{i,l_i} \mid MSN'_{i,l_i}] \mid PSN'_{m'} \equiv D'_i \mid PSN'_{m'}$ where $MSN'_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_{l_i-1}}_{l_i-1} \mid SN^{h_{l_i}}_{i,l_i},\ \nu\, nid_{i,l_i}.\nu\, sr_{i,l_i}.SN^{h_{l_i}}_{i,l_i} = \nu\, nid_{j_h}.\nu\, sr_{j_h}.SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\}$. Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN'_{m'} \mid !MSA \mid !SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_i \mid PSN_m \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid !MSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid PSN'_{m'} \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN'_{m'} \mid !MSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

- **(New MS)** $n' = n{+}1 \wedge C'_j = C'\ \forall j \neq n{+}1,\ C'_{n+1} = GSS_{n+1,1}[SMS^1_{n+1,1} \mid 0]$ and $PSN'_{m'} = PSN_m$. In this case the $!SSA$ component is unrolled and a new single session mobile station C'_{n+1} synchronizes with the $Init$ process. if $SSA = \nu\, \widetilde{ss}_{n+1,1}.(Init_{n+1,1} \mid SMS^0_{n+1,1}) \xrightarrow{\tau} C'_{n+1} \equiv \nu\, \widetilde{ss}_{n+1,1}.SMS^1_{n+1,1} \equiv GSS_{n+1,1}[SMS^1_{n+1,1} \mid SSN_{n+1,1}],\ SSN_{n+1,1} \equiv 0$. Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D_n \mid D'_{n+1} \mid PSN_m \mid !MSA \mid !SN)$ where $D'_{n+1} = GMS_{n+1,1}[MMS^1_{n+1,1} \mid MSN_{n+1,1}],\ MSN_{n+1,1} \equiv 0$ then we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid \nu\, \widetilde{ms}_{n+1,1}.(Init_{n+1,1} \mid$

$MMS^0_{n+1,1})\ |!MSA\ |!SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ms}_{n+1,1}.(Init_{n+1,1} \mid MMS^0_{n+1,1}) \mid PSN_m\ |!MSA\ |!SN) \xrightarrow{\tau} \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ms}_{n+1,1}.(MMS^1_{n+1,1}) \mid PSN_m\ |!MSA\ |!SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ms}_{n+1,1}.(MMS^1_{n+1,1} \mid 0) \mid PSN_m\ |!MSA\ |!SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid D'_{n+1} \mid PSN_m\ |!MSA\ |!SN) = D'$ and $(C',\ D') \in \mathcal{R}$

Let $(C,\ D) \in \mathcal{R}$ and let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n\ |!PSN_m\ |!MSA\ |!SN)$ and $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n\ |!PSN_m\ |!SSA\ |!SN)$.
If $C \xrightarrow{\tau} C'$ by Lemma 7.3 we have that $C' \equiv \nu\, dck.(C'_1 \mid \cdots \mid C'_{n'}\ |!PSN'_{m'}\ |!MSA\ |!SN)$ and we have 3 cases:

- $\exists i\ C_i \xrightarrow{\tau} C'_i \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n \wedge PSN'_{m'} = PSN_m$, hence $C_i \equiv GMS_{i,l_i}[MMS^{kl_i}_{i,l_i} \mid MSN_{i,l_i}]$ can do a silent transition this means that C_i is either of the form $C_i \equiv GMS_{i,l_i}[MMS^4_{i,l_i} \mid MSN_{i,l_i}]$ (i.e. we have two cases depending on the evaluation of the conditional statement), or of the form $C_i \equiv GMS_{i,l_i}[MMS^6_{i,l_i} \mid MSN_{i,l_i}]$ (i.e a new session of the mobile station is created). Hence we have 3 cases:

  - **(Conditional-then)** if $C_i \equiv GMS_{i,l_i}[MMS^4_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\tau} GMS_{i,l_i}[MMS^5_{i,l_i} \mid MSN_{i,l_i}] = C'_i$ then $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) =_E$ TMSI_REALL i.e. $N_{i,l_i} =_E \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))$. By definition of $\mathcal{R}$ we have that $D_i \equiv GSS_{i,l_i}[SMS^4_{i,l_i} \mid SSN_{i,l_i}]$, $SMS^4_{i,l_i} = SX_{i,l_i} \mid SChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\}$ and $SChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\} \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle.\ \overline{d_{i,l_i}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i}))\rangle$ *else* $0 \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle.\overline{d_{i,l_i}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))))\rangle$ *else* $0 \xrightarrow{\tau} \overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle.\ \overline{d_{i,l_i}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,j}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))))\rangle$ hence $SMS^4_{i,l_i} \xrightarrow{\tau} SMS^5_{i,l_i}$ and $D_i \xrightarrow{\tau} SMS_{i,l_i}[SMS^5_{i,l_i} \mid SSN_{i,l_i}] = D'_i$ then let

$D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

– **(Conditional-else)** if $C_i \equiv GMS_{i,l_i}[MMS^4_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\tau} GMS_{i,l_i}[MMS^8_{i,l_i} \mid MSN_{i,l_i}] = C'_i$ then $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) \neq_E$ TMSI_REALL i.e. $N_{i,l_i} \neq_E \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i}))$. By definition of $\mathcal{R}$ we have that $D_i \equiv GSS_{i,l_i}[SMS^4_{i,l_i} \mid SSN_{i,l_i}],\ SMS^4_{i,l_i} = SX_{i,l_i} \mid SChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\}$ and $SChk_{i,l_i}\{^{N_{i,l_i}}/_{y_{i,l_i}}\} \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i})) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$. $\overline{d_{i,l_i}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, N_{i,l_i}))\rangle$ *else* $0 \equiv$ *if* $\mathtt{fst}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})) =$ TMSI_REALL *then* $\overline{up}\langle \mathtt{senc}(ck_{i,l_i}, mr_{i,l_i}, \text{COMPLETE})\rangle$. $\overline{d_{i,l_i}}\langle \mathtt{snd}(\mathtt{sdec}(ck_{i,l_i}, \mathtt{senc}(ck_{i,l_i}, sr_{i,l_i}, \mathtt{pair}(\text{TMSI\_REALL}, z_{i,l_i})))\rangle$ *else* $0 \xrightarrow{\tau} 0$ hence $SMS^4_{i,l_i} \xrightarrow{\tau} SMS^8_{i,l_i}$ and $D_i \xrightarrow{\tau} SMS_{i,l_i}[SMS^8_{i,l_i} \mid SSN_{i,l_i}] = D'_i$ then let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

– **(New session)** if $C_i \equiv GMS_{i,l_i}[MMS^6_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\tau} D'_i \equiv GMS_{i,l_i+1}[MMS^1_{i,l_i+1} \mid MSN_{i,l_i+1}],\ MSN_{i,l_i+1} \equiv MSN_{i,l_i} \mid 0$ then by definition of $\mathcal{R}$ we have that $D_i \equiv GSS_{i,l_i}[SMS^6_{i,l_i} \mid SSN_{i,l_i}]$ and $D_i \mid !SSA \equiv GSS_{i,l_i}[SMS^6_{i,l_i} \mid SSN_{i,l_i}] \mid \nu\, d_{i,l_i+1},\ id_{i,l_i+1},\ ck_{i,l_i+1}\nu\, mr_{i,l_i+1}.(Init_{i,l_i+1} \mid SMS^0_{i,l_i+1}) \mid !SSA \xrightarrow{\tau} GSS_{i,l_i}[SMS^6_{i,l_i} \mid SSN_{i,l_i}] \mid \nu\, d_{i,l_i+1},\ id_{i,l_i+1},\ ck_{i,l_i+1}.\nu\, mr_{i,l_i+1}.(SMS^1_{i,l_i+1}) \mid !SSA \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i} \mid SSN_{i,l_i}) \mid \nu\, d_{i,l_i+1},\ id_{i,l_i+1},\ ck_{i,l_i+1}.\nu\, mr_{i,l_i+1}.(SMS^1_{i,l_i+1}) \mid !SSA \equiv \nu\, \widetilde{ss}_{i,l_i+1}.\nu\, \widetilde{nid}_{i,l_i}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i} \mid SMS^1_{i,l_i+1} \mid SSN_{i,l_i} \mid$

0) $|!SSA \equiv GSS_{i,l_i+1}[SMS^1_{i,l_i+1} \mid SSN_{i,l_i+1}]$ $|!SSA = C'_i$ $|!SSA$ Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_{n'} \mid PSN_m \; |!SSA \; |!SN)$, we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \; |!SSA \; |!SN) \equiv \nu\, dck.(SSA \mid D_1 \mid \cdots \mid D_n \mid PSN_m \; |!SSA \; |!SN) \xrightarrow{\tau} \nu\, dck.(D'_{n'} \mid D_1 \mid \cdots \mid D_n \mid PSN_m \; |!SSA \; |!SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_{n'} \mid PSN_m \; |!SSA \; |!SN) = D'$ and $(C',\ D') \in \mathcal{R}$

- **(MS/SN synch)** $\exists i\ C_i \mid PSN_m \xrightarrow{\tau} C'_i \mid PSN'_{m'} \wedge C'_j = C_j\ \forall j \neq i \wedge n' = n$ the only possible silent transition between C_i and PSN_m is the communication on the channel dck hence, $C_i \equiv GMS_{i,l_i}[MMS^2_{i,l_i} \mid MSN_{i,l_i}]$ and $PSN_{m-} \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_m},\ m \geq 1$ if $C_i \mid PSN_m \xrightarrow{\tau} C'_i \mid PSN'_{m'}$ we have that $C'_i \equiv GMS_{i,l_i}[MMS^3_{i,l_i} \mid MSN'_{i,l_i}],\ MSN'_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_{l_i-1}}_{l_i-1} \mid SN^{h_{l_i}}_{i,l_i}$, and $\nu\, nid_{i,l_i}.SN^{h_{l_i}}_{i,l_i} \equiv \nu\, nid_{j_h}.\nu\, sr_{j_h}.SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\}$ for some $1 \leq h \leq m$ such that $SN^1_{j_h}$ occurs in PSN_m and $PSN'_{m'} \equiv SN^1_{j_1} \mid \cdots \mid SN^1_{j_{h-1}} \mid SN^1_{j_{h+1}} \mid \cdots \mid SN^1_{j_m}$. By definition of $\mathcal{R}$ we have that $D_i \equiv GSS_{i,l_i}[SMS^2_{i,l_i} \mid SSN_{i,l_i}],\ SSN_{i,l_i} \equiv MSN_{i,l_i}$. We have that $D_i \mid PSN_m \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^2_{i,l_i} \mid SSN_{i,l_i} \mid PSN_m) \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^2_{i,l_i} \mid SN^1_{j_h} \mid SSN_{i,l_i} \mid PSN'_{m'}) \xrightarrow{\tau} \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i-1}.\nu\, nid_{j_h}.\nu\, sr_{j_h}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^3_{i,l_i} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}}\} \mid SSN_{i,l_i} \mid PSN'_{m'}) \equiv \nu\, \widetilde{ss}_{i,l_i}.\nu\, \widetilde{nid}_{i,l_i}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,l_i-1} \mid SMS^3_{i,l_i} \mid SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\} \mid SSN_{i,l_i} \mid PSN'_{m'}) \equiv GSS_{i,l_i}[SMS^3_{i,l_i} \mid SSN'_{i,l_i}] \mid PSN'_{m'} \equiv D'_i \mid PSN'_{m'}$ where $SSN'_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_{l_i-1}}_{l_i-1} \mid SN^{h_{l_i}}_{i,l_i},\ \nu\, nid_{i,l_i}.\nu\, sr_{i,l_i}.SN^{h_{l_i}}_{i,l_i} = \nu\, nid_{j_h}.\nu\, sr_{j_h}.SN^2_{j_h}\{^{ck_{i,l_i}}/_{xck_{j_h}},\ ^{nid_{i,l_i}}/_{nid_{j_h}},\ ^{sr_{i,l_i}}/_{sr_{j_h}},\ ^{w_{i,l_i}}/_{w_{j_h}}\}$. Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN'_{m'} \; |!SSA \; |!SN)$ we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \; |!SSA \; |!SN) \equiv \nu\, dck.(D_i \mid PSN_m \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \; |!SSA \; |!SN) \xrightarrow{\tau} \nu\, dck.(D'_i \mid PSN'_{m'} \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \; |!SSA \; |!SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN'_{m'} \; |!SSA \; |!SN) = D'$ and $(C',\ D') \in \mathcal{R}$

- **(New MS)** $n' = n+1 \wedge C'_j = C' \; \forall j \neq n+1,\; C'_{n+1} = GMS_{n+1,1}[MMS^1_{n+1,1} \mid 0]$ and $PSN'_{m'} = PSN_m$. In this case the $!MSA$ component is unrolled and a new single session mobile station C'_{n+1} synchronizes with the $Init$ process. if $MSA = \nu\, \widetilde{ms}_{n+1,1}.(Init_{n+1,1} \mid MMS^0_{n+1,1}) \xrightarrow{\tau} C'_{n+1} \equiv \nu\, \widetilde{ms}_{n+1,1}.MMS^1_{n+1,1}$ $\equiv GMS_{n+1,1}[MMS^1_{n+1,1} \mid MSN_{n+1,1}],\; MSN_{n+1,1} \equiv 0$. Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D_n \mid D'_{n+1} \mid PSN_m \mid !SSA \mid !SN)$ where $D'_{n+1} = GSS_{n+1,1}[SMS^1_{n+1,1} \mid SSN_{n+1,1}],\; SSN_{n+1,1} \equiv 0$ then we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid \nu\, \widetilde{ss}_{n+1,1}.(Init_{n+1,1} \mid SMS^0_{n+1,1}) \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ss}_{n+1,1}.(Init_{n+1,1} \mid SMS^0_{n+1,1}) \mid PSN_m \mid !SSA \mid !SN) \xrightarrow{\tau} \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ss}_{n+1,1}.$ $(SMS^1_{n+1,1}) \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid \nu\, \widetilde{ss}_{n+1,1}.$ $(SMS^1_{n+1,1} \mid 0) \mid PSN_m \mid !SSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid D'_{n+1} \mid PSN_m \mid !MSA \mid !SN) = D'$ and $(C',\; D') \in \mathcal{R}$

Let $(C,\; D) \in \mathcal{R}$ and let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid !PSN_m \mid !SSA \mid !SN)$ and $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid !PSN_m \mid !MSA \mid !SN)$. If $C \xrightarrow{\alpha} C'$ then by Lemma 7.4 we have 2 cases:

- $\exists i \; C_i \xrightarrow{\alpha} C'_i \wedge C'_j = C_j \; \forall j \neq i \wedge n' = n \wedge PSN'_{m'} = PSN_m$ in this case one of the C_i components does the transition. We have 2 cases:
  - **(Active session MS α-transition)** if $C_i = GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}]$ $\xrightarrow{\alpha} C'_i \equiv GSS_{i,l_i}[SMS^{k_i+1}_{i,l_i} \mid SSN_{i,l_i}], k \neq 2, 4$ then we have 3 cases:
    i) if $k_i = 1,\; SMS^{k_i}_{i,l_i} = SMS^1_{i,l_i} \xrightarrow{\nu x_{i,l_i}.\overline{up}\langle x_{i,l_i}\rangle} SMS^2_{i,l_i} = SMS^{k_i+1}_{i,l_i}$, then by definition of $\mathcal{R}$ we have that $MMS^{k_i}_{i,l_i} = MMS^1_{i,l_i}$ $\xrightarrow{\nu x_{i,l_i}.\overline{up}\langle x_{i,l_i}\rangle} MMS^2_{i,l_i} = MMS^{k_i+1}_{i,l_i}$, hence $D_i = GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\nu x_{i,l_i}.\overline{up}\langle x_{i,l_i}\rangle} GMS_{i,l_i}[MMS^{k_i+1}_{i,l_i} \mid MSN_{i,l_i}] = D'_i$
    ii) if $k_i = 3,\; SMS^{k_i}_{i,l_i} = SMS^3_{i,l_i} \xrightarrow{dw(N_{i,l_i})} SMS^4_{i,l_i} = SMS^{k_i+1}_{i,l_i}$, then by definition of $\mathcal{R}$ we have that $MMS^{k_i}_{i,l_i} = MMS^3_{i,l_i} \xrightarrow{dw(N_{i,l_i})}$

$MMS^4_{i,l_i} = MMS^{k_i+1}_{i,l_i}$ hence $GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{dw(N_{i,l_i})} GMS_{i,l_i}[MMS^{k_i+1}_{i,l_i} \mid MSN_{i,l_i}] = D'_i$

iii) if $k_i = 5$, $SMS^{k_i}_{i,l_i} = SMS^5_{i,l_i} \xrightarrow{\nu k_{i,l_i}.\overline{up}\langle k_{i,l_i}\rangle} SX_{i,l_i} \mid SK_{i,l_i} \mid \overline{d_{i,l_i}}\langle \texttt{snd}(\texttt{sdec}(ck_{i,l_i},\ N_{i,l_i}))\rangle = SMS^{k_i+1}_{i,l_i}$. By definition of $\mathcal{R}$ we have that $MMS^{k_i}_{i,l_i} \equiv MMS^6_{i,l_i} \xrightarrow{\nu k_{i,l_i}.\overline{up}\langle k_{i,l_i}\rangle} MX_{i,l_i} \mid MK_{i,l_i} \mid \overline{d_{i,1}}\langle \texttt{snd}(\texttt{sdec}(ck_{i,l_i},\ N_{i,l_i}))\rangle = MMS^{k_i+1}_{i,l_i}$, hence $D_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}] \xrightarrow{\nu k_{i,l_i}.\overline{up}\langle k_{i,l_i}\rangle} GMS_{i,l_i}[MMS^{k_i+1}_{i,l_i} \mid MSN_{i,l_i}] = D'_i$

Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ where $D'_i = GMS_{i,l_i}[MMS^{k_i+1}_{i,l_i} \mid MSN_{i,l_i}]$ as shown in the cases above, then we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

– **(SN post-synch α-transition)** if $C_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}] \xrightarrow{\alpha} C'_i \equiv GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN'_{i,l_i}]$ for some $1 \leq j \leq l_i$ such that $SSN_{i,l_i} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_j}_{i,j} \mid \cdots \mid SN^{h_{l_i}}_{i,l_i} \xrightarrow{\alpha} SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_j+1}_{i,j} \mid \cdots \mid SN^{h_{l_i}}_{i,l_i} = SSN'_{i,l_i}$ and $h_j \in \{2,3\}$ then by definition of $\mathcal{R}$ we have that $D_i \equiv GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}]$ and $MSN_{i,l_i} \equiv SSN_{i,l_i} \overset{S}{S} N'_{i,l_i} = MSN_{i,l_i}$ then $D_i \xrightarrow{\alpha} GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN'_{i,l_i}] = D'_i$ where $MSN'_{i,l_i} = SSN'_{i,l_i}$. Let $D' = \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ then we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{\alpha} \nu\, dck.(D'_i \mid D_1 \mid \cdots \mid D_{i-1} \mid D_{i+1} \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D'_i \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) = D'$ and $(C',\ D') \in \mathcal{R}$

- **(SN pre-synch α-transition)** $C'_j = C_j\ \forall \wedge n' = n \wedge PSN'_{m'} = PSN_m \mid SN^1_{j_{m+1}},\ m' = m+1$. In this case a new SN is unrolled and does the first labelled transition before synchronizing with the MS. If $!SN \xrightarrow{\alpha} SN^1_{j_{m+1}} \mid !SN$

and $C' \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_{m+1} \mid !SSA \mid !SN), PSN_{m+1} \equiv PSN_m \mid SN^1_{j_{m+1}}$ then let $D' \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_{m+1} \mid !MSA \mid !SN)$, where $PSN_{m+1} \equiv PSN_m \mid SN^1_{j_{m+1}}$, we have that $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN \mid SN^0_{j_{m+1}}) \equiv \nu\, dck.(SN^0_{j_{m+1}} \mid D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \xrightarrow{dw(z_{j_{m+1}})} \nu\, dck.(SN^1_{j_{m+1}} \mid D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN) \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid SN^1_{j_{m+1}} \mid !MSA \mid !SN) = D'$, and $(C',\ D') \in \mathcal{R}$

Let $(C,\ D) \in \mathcal{R}$ and let $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !MSA \mid !SN)$ and $D \equiv \nu\, dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !SSA \mid !SN)$.

Analogous to the previous case

□

B.3 Proof of Lemma 7.2

In order to prove Lemma 7.2, we define the general structure of the frames produced by partial evolution of our processes through the following substitutions:

$$\sigma^{id}_{i,j} \stackrel{def}{=} \left\{ {}^{id_{i,1}}/_{x_{i,1}}, {}^{id_{i,2}}/_{x_{i,2}}, \ldots, {}^{id_{i,j}}/_{x_{i,j}} \right\}$$

$$\sigma^{M}_{i,j} \stackrel{def}{=} \left\{ {}^{id_{i,1}}/_{x_{i,1}}, {}^{M_{i,2}}/_{x_{i,2}} \ldots, {}^{M_{i,j}}/_{x_{i,j}} \right\}$$

$$\sigma^{K}_{i,j} \stackrel{def}{=} \left\{ {}^{\mathtt{senc}(ck_{i,1},\ mr_{i,1},\ \textsc{Complete})}/_{k_{i,1}}, \ldots, {}^{\mathtt{senc}(ck_{i,j},\ mr_{i,j},\ \textsc{Complete})}/_{k_{i,j}} \right\}$$

$$\sigma^{nid}_{i,j} \stackrel{def}{=} \left\{ {}^{\mathtt{senc}(ck_{i,1},\ sr_{i,1},\ \mathtt{pair}(\text{TMSI\_}\textsc{Reall},\ nid_{i,1}))}/_{y_{i,1}}, \ldots, {}^{\mathtt{senc}(ck_{i,j},\ sr_{i,j},\ \mathtt{pair}(\text{TMSI\_}\textsc{Reall},\ nid_{i,j}))}/_{y_{i,j}} \right\}$$

We define the general structure of the frame of one of the grouped single (resp. multi)-session system components

$$\begin{aligned}\sigma(C_i) &\stackrel{def}{=} \sigma^{id}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}}\\ \sigma(D_i) &\stackrel{def}{=} \sigma^{M}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}}\end{aligned}$$

These frames represent the knowledge a grouped single (resp. multi)-session system component releases to the environment after $i + l_i - 1$ mobile stations completely executed (resp. the i^{th} MS completely executed $l_i - 1$ sessions of) the TMSI reallocation protocol while the l_i^{th} executed k_{l_i} steps (resp. while the l_i session is at the k_{l_i} execution step).

In the following lemma we prove the correctness of the given frame structure.

Lemma 7.5. *Let* $(C, D) \in \mathcal{R}, C \equiv \nu\ dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid !SN)$, $D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ *then* $\varphi(C) \equiv \nu\ dck.(\varphi(C_1) \mid \cdots \mid \varphi(C_n))$, $\varphi(D) \equiv \nu\ dck.(\varphi(D_1) \mid \cdots \mid \varphi(D_n))$ *and* $\forall i,\ l_i, 1 \leq i \leq n, l_i \geq 1$,

$$\begin{aligned}\varphi(C_i) &\equiv \varphi(GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}])\\ &\equiv \nu\ \widetilde{ss}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.\sigma(C_i) = \nu\ \widetilde{ss}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.(\sigma^{id}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})\\ \varphi(D_i) &\equiv \varphi(GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}])\\ &\equiv \nu\ \widetilde{ms}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.\sigma(D_i) = \nu\ \widetilde{ms}_{i,l_i}.\nu\ \widetilde{nid}_{i,j_{nid}}.(\sigma^{M}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})\end{aligned}$$

$$\textit{where } j_{id} = \begin{cases} l_i - 1 & \textit{if } k_i \leq 1\\ l_i & \textit{otherwise}\end{cases} \quad j_K = \begin{cases} l_i - 1 & \textit{if } k_i \leq 5, k_i = 8\\ l_i & \textit{otherwise}\end{cases}$$

$$j_{nid} = \begin{cases} l_i - 1 & \textit{if } k_i \leq 2\\ l_i & \textit{otherwise}\end{cases}$$

Proof. By definition of PSN_m, PSN_m we have that $\varphi(PSN_m) \equiv \varphi(PSN_m) \equiv 0,\ \forall\ m \geq 0$ and by definition of $SSA,\ MSA$ and SN we have that $\varphi(!SSA) \equiv \varphi(!MSA) \equiv \varphi(!SN) \equiv 0$.

By definition of $\mathcal{R}$ we have that $C \equiv \nu\, dck.(C_1 \mid \cdots \mid C_n \mid PSN_m \mid !SSA \mid$

$!SN), D \equiv \nu\ dck.(D_1 \mid \cdots \mid D_n \mid PSN_m \mid !MSA \mid !SN)$ and $\varphi(C) \equiv \nu\, dck.(\varphi(C_1) \mid \cdots \mid \varphi(C_n)),\ \varphi(D) \equiv \nu\, dck.(\varphi(D_1) \mid \cdots \mid \varphi(D_n))$. We show by induction over l_i that

$\varphi(C_i) \equiv \nu\, \widetilde{ss_{i,l_i}}.\nu\, \widetilde{nid}_{i,j_{nid}}.(\sigma^{id}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})$,

$\varphi(D_i) \equiv \nu\, \widetilde{ms_{i,l_i}}.\nu\, \widetilde{nid}_{i,j_{nid}}.(\sigma^{M}_{i,j_{id}} \mid \sigma^{K}_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})$

where $C_i = GSS_{i,l_i}[SMS^{k_i}_{i,l_i} \mid SSN_{i,l_i}]$, $D_i = GMS_{i,l_i}[MMS^{k_i}_{i,l_i} \mid MSN_{i,l_i}]$ and

$$j_{id} = \begin{cases} l_i - 1 & \text{if } k_i \leq 1 \\ l_i & \text{otherwise} \end{cases} \quad j_K = \begin{cases} l_i - 1 & \text{if } k_i \leq 5, k_i = 8 \\ l_i & \text{otherwise} \end{cases}$$

$$j_{nid} = \begin{cases} l_i - 1 & \text{if } k_i \leq 2 \\ l_i & \text{otherwise} \end{cases}$$

- $l_i = 0$). We prove the statements hold for $l_i = 0$. Let $l_i = 0$, by definition of $\mathcal{R}, C_i \equiv GSS_{i,0}[SMS^{k_i}_{i,0} \mid SSN_{i,0}] \equiv 0,\ D_i \equiv GMS_{i,0}[MMS^{k_1}_{i,0} \mid MSN_{i,0}] \equiv 0$ $\varphi(C_i) \equiv 0 \equiv \varphi(D_i)$

- $l_i = m + 1$). We assume the statement holds for $l_i = m$, and show that it holds for $l_i = m + 1$ i.e.
  $\varphi(C_i) \equiv \varphi(GSS_{i,m+1}[SMS^{k_{m+1}}_{i,m+1} \mid SSN_{i,m+1}]) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{id}_{i,j'_{id}} \mid \sigma^{K}_{i,j'_K+1} \mid \sigma^{nid}_{i,j'_{nid}+1})$, $\varphi(D_i) \equiv \varphi(GMS_{i,m+1}[MMS^{k_{m+1}}_{i,m+1} \mid MSN_{i,m+1}]$
  $\equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{M}_{i,j'_{id}} \mid \sigma^{K}_{i,j'_K} \mid \sigma^{nid}_{i,j'_{nid}})$, where

  $$j'_{id} = \begin{cases} m & \text{if } k_{m+1} \leq 1 \\ m+1 & \text{otherwise} \end{cases} \quad j'_K = \begin{cases} m & \text{if } k_{m+1} \leq 5, \text{ or } k_{m+1} = 8 \\ m+1 & \text{otherwise} \end{cases}$$

  $$j'_{nid} = \begin{cases} m & \text{if } k_{m+1} \leq 2 \\ m+1 & \text{otherwise} \end{cases}$$

  Let $l_i = m + 1$, by definition of $\mathcal{R}$, we have that $C_i \equiv GSS_{i,m+1}[SMS^{k_{m+1}}_{i,m+1} \mid SSN_{i,m+1}]$, $D_i \equiv GMS_{i,m+1}[MMS^{k_{m+1}}_{i,m+1} \mid MSN_{i,m+1}]$ where $GSS_{i,m+1}[] = \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,m} \mid \_) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(SMS^7_{i,1} \mid \cdots \mid SMS^7_{i,m-1} \mid SMS^7_{i,m} \mid \_) \equiv \nu\, id_{i,m+1}, d_{i,m+1}, ck_{i,m+1}.\nu\, nid_{i,j'_{nid}}.(GSS_{i,m}$

$[SMS^7_{i,m}] \mid [\_])$ and
$GMS_{i,m+1}[] = \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{1,j'_{nid}}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,m} \mid \_ \mid !R_i) \equiv$
$\nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{1,j'_{nid}}.(MMS^7_{i,1} \mid \cdots \mid MMS^7_{i,m-1} \mid MMS^7_{i,m} \mid \_ \mid !R_i) \equiv$
$\nu\, ck_{i,m+1}.\nu\, nid_{i,j'_{nid}}.(GMS_{i,m}[MMS^7_{1,m} \mid \_])$ then $\varphi(C_i) \equiv$
$\varphi(GSS_{i,m+1}[SMS^{k_{m+1}}_{i,m+1} \mid SSN_{i,m+1}]) \equiv \varphi(GSS_{i,m}[SMS^7_{i,m} \mid SSN_{i,m}])$
$\mid \nu\, id_{i,m+1}, d_{i,m+1}, ck_{i,m+1}.\nu\, nid_{i,m+1}.\varphi(SMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1}) \equiv$ (by ind. hyp. and since $\varphi(MMS^7_{i,j}) \equiv MX_{i,j} \mid K_{i,j} \forall i, j$ and since the conditional enables the process $SMS^{k_j}_{i,j}$ to reach state 7 if and only if it receives
$\texttt{sdec}(ck_{i,j}, \texttt{pair}(\text{TMSI\_REALL}, nid_{i,j}$ from the $SN_{i,j}$ we have that $SSN_{i,m} \equiv$
$SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_m}_{i,m}$ with $h_1, \ldots, h_m \geq 3$ hence $\varphi(SSN_{i,m}) \equiv \sigma^{nid}_{i,m}$)
$\nu\, \widetilde{ss}_{i,m}.\nu\, \widetilde{nid}_{i,m}.(\sigma^{id}_{i,m} \mid \sigma^{k}_{i,m} \mid \sigma^{nid}_{i,m}) \mid \nu\, id_{i,m+1}, d_{i,m+1}, ck_{i,m+1}, nid_{i,j'_{nid}}.$
$\varphi(SMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1})) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{id}_{i,m} \mid \sigma^{k}_{i,m} \mid \sigma^{nid}_{i,m} \mid$
$\varphi(SMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1}))$ and
$\varphi(D_i) \equiv \varphi(GMS_{i,m+1}[MMS^{k_{m+1}}_{i,m+1} \mid MSN_{i,m+1}]) \equiv \varphi(GMS_{i,m}[MMS^7_{i,m} \mid$
$MSN_{i,m} \mid \varphi(\nu\, ck_{i,m+1}, nid_{i,j'_{nid}}.(MMS^{k_{m+1}}_{1,m+1} \mid SN^{h_{m+1}}_{1,m+1})]) \equiv$(since $\varphi(MMS^7_{i,j})$
$\equiv MX_{i,j} \mid K_{i,j} \forall i, j$ since the conditional enables the process $SMS^{k_j}_{i,j}$ to reach state 7 if and only if it receives $\texttt{sdec}(ck_{i,j}, \texttt{pair}(\text{TMSI\_REALL}, nid_{i,j}$ from the $SN_{i,j}$ we have that $MSN_{i,m} \equiv SN^{h_1}_{i,1} \mid \cdots \mid SN^{h_m}_{i,m}$ with $h_1, \ldots, h_m \geq 3$ hence $\varphi(MSN_{i,m}) \equiv \sigma^{nid}_{i,m}$ and by ind. hyp. $\varphi(GMS_{i,m}[MMS^7_{i,m} \mid MSN_{i,m}]) \equiv$
$\nu\, \widetilde{ms}_{i,m}.\nu\, \widetilde{nid}_{i,m}.(\sigma^{M}_{i,m} \mid \sigma^{K}_{i,m} \mid \sigma^{nid}_{i,m}))\ \nu\, \widetilde{ms}_{i,m}.\nu\, \widetilde{nid}_{i,m}.(\sigma^{M}_{i,m} \mid \sigma^{k}_{i,m} \mid \sigma^{nid}_{i,m} \mid$
$\nu\, ck_{i,m+1}, \nu\, nid_{i,j'_{nid}}.\varphi(MMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1})) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{M}_{i,m} \mid$
$\sigma^{k}_{i,m} \mid \sigma^{nid}_{i,m} \mid \varphi(MMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1}))$
we have 6 cases:

1. $k_{m+1} = 1,\ SN^{h_{m+1}}_{i,m+1} = 0$ then we have that $\varphi(SMS^1_{i,m+1} \mid 0) \equiv 0$, hence $\varphi(C_i) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.$
$(\sigma^{id}_{i,m} \mid \sigma^{K}_{i,m} \mid \sigma^{nid}_{i,m} \mid 0) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{id}_{i,j'_{id}} \mid \sigma^{K}_{i,j'_K} \mid \sigma^{nid}_{i,j'_{nid}})$ and $\varphi(MMS^1_{i,m+1} \mid 0) \equiv 0$, hence, $\varphi(D_i) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{M}_{i,m} \mid \sigma^{K}_{i,m} \mid$
$\sigma^{nid}_{i,m} \mid 0) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{M}_{i,j'_{id}} \mid \sigma^{K}_{i,j'_K} \mid \sigma^{nid}_{i,j'_{nid}})$

2. $k_{m+1} = 2,\ SN_{i,m+1}^{h_{m+1}} = 0$ then we have that
   $\varphi(SMS_{i,m+1}^{2} \mid 0) \equiv SX_{i,m+1}$, hence, $\varphi(C_i) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{id_{i,m+1}}/_{x_{i,m+1}}\}) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid})$ and
   $\varphi(MMS_{i,m+1}^{2} \mid 0) \equiv MX_{i,m+1}$, hence, $\varphi(D_i) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,m}.(\sigma_{i,m}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{M_{i,m+1}}/_{x_{i,m+1}}\}) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid}),\ j'_{nid} = m$

3. $k_{m+1} \in \{3,\ 4,\ 5,\ 8\},\ h_{m+1} = 2$ then we have that
   $\varphi(SMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{h_{m+1}}) \equiv \{^{id_{i,m+1}}/_{x_{i,m+1}}\}$, hence $\varphi(C_i) \equiv \nu\, \widetilde{ss}_{i,m+1}.\ \nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{id_{i,m+1}}/_{x_{i,m+1}}\}) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid})$ and
   $\varphi(MMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{h_{m+1}}) \equiv \{^{M_{i,m+1}}/_{x_{i,m+1}}\}$, hence $\varphi(D_i) \equiv \nu\, \widetilde{ms}_{i,m+1}.\ \nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid}) \mid \{^{M_{i,m+1}}/_{x_{i,m+1}}\}) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.\ (\sigma_{i,m+1}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid}),\ j'_{nid} = m+1$

4. $k_{m+1} \in \{3,\ 4,\ 5,\ 8\},\ h_{m+1} \in \{3, 4\}$ then we have that
   $\varphi(SMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{h_{m+1}}) \equiv$
   $\{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid \{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}$, hence
   $\varphi(C_i) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid \{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}) \equiv$
   $\nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m+1}^{nid})$ and $\varphi(MMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{h_{m+1}}) \equiv$
   $\{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid \{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}$, hence
   $\varphi(D_i) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid \{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}) \equiv \nu\, dck.\nu\, \widetilde{ms}_{i,m+1}.$
   $\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{M} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m+1}^{nid}),\ j'_{nid} = m+1$

5. $k_{m+1} \in \{6,\ 7\},\ h_{m+1} = 2$ then we have that
   $\varphi(SMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{2}) \equiv \{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid SK_{i,m+1})$, hence $\varphi(C_i) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m}^{id} \mid \sigma_{i,m}^{K} \mid \sigma_{i,m}^{nid} \mid \{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid SK_{i,m+1}) \equiv \nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma_{i,m+1}^{id} \mid \sigma_{i,m+1}^{K} \mid \sigma_{i,m}^{nid})$ and
   $\varphi(MMS_{i,m+1}^{k_{m+1}} \mid SN_{i,m+1}^{2}) \equiv \{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid MK_{i,m+1}$, hence $\varphi(D_i) \equiv$

$\nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^M_{i,m} \mid \sigma^K_{i,m} \mid \sigma^{nid}_{i,m} \mid \{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid MK_{i,m+1}) \equiv$
$\nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^M_{i,m+1} \mid \sigma^K_{i,m+1} \mid \sigma^{nid}_{i,m}),\ j'_{nid} = m+1$

6. $k_{m+1} \in \{6,\ 7\},\ h_{m+1} \in \{3, 4\}$ then we have that
$\varphi(SMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1}) \equiv \{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid SK_{i,m+1} \mid \nu\, nid_{i,m+1}.$
$\{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}$, hence $\varphi(C_i) \equiv$
$\nu\, \widetilde{ss}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{id}_{i,m} \mid \sigma^K_{i,m} \mid \sigma^{nid}_{i,m} \mid \{^{id_{i,m+1}}/_{x_{i,m+1}}\} \mid SK_{i,m+1} \mid$
$\nu\, nid_{i,m+1}.\{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}) \equiv \nu\, \widetilde{ss}_{i,m+1}.$
$\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^{id}_{i,m+1} \mid \sigma^K_{i,m+1} \mid \sigma^{nid}_{i,m+1})$ and
$\varphi(MMS^{k_{m+1}}_{i,m+1} \mid SN^{h_{m+1}}_{i,m+1}) \equiv \{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid MK_{i,m+1} \mid \nu\, nid_{i,m+1}.$
$\{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}$, hence
$\varphi(D_i) \equiv \nu\, \widetilde{ms}_{i,m+1}.\nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^M_{i,m} \mid \sigma^K_{i,m} \mid \sigma^{nid}_{i,m} \mid \{^{M_{i,m+1}}/_{x_{i,m+1}}\} \mid$
$MK_{i,m+1} \mid \nu\, nid_{i,m+1}.\{^{\mathtt{senc}(ck_{i,m+1}, sr_{i,m+1}, \mathtt{pair}(\text{TMSI\_REALL}, nid_{i,m+1}))}/_{y_{i,m+1}}\}) \equiv$
$\nu\, \widetilde{ms}_{i,m+1}.\ \nu\, \widetilde{nid}_{i,j'_{nid}}.(\sigma^M_{i,m+1} \mid \sigma^K_{i,m+1} \mid \sigma^{nid}_{i,m+1}),\ j'_{nid} = m+1$

□

We can now prove Lemma 7.2

Lemma 7.2. *Let* $(C, D) \in \mathcal{R}$ *then* $C \approx_s D$

Proof. by Lemma 7.5 we have that $\forall\ (C, D) \in \mathcal{R},\ \varphi(C) \equiv \nu\, dck.(\varphi(C_1) \mid \cdots \mid \varphi(C_n)),\ \varphi(D) \equiv \nu\, dck.(\varphi(D_1) \mid \cdots \mid \varphi(D_n))$. By lemma 7.5 we have that $\forall i, j,\ 1 \leq i \leq n,\ 1 \leq j \leq l_i,\ \varphi(D_i) = \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,j_{nid}}.(\sigma^M_{i,j_{id}} \mid \sigma^K_{i,j_K} \mid \sigma^{nid}_{i,j_{nid}})$ and by definition $\sigma^M_{i,j_{id}} = \{^{id_{i,1}}/_{x_{i,1}},\ ^{M_{i,2}}/_{x_{i,2}} \ldots,\ ^{M_{i,j_{id}}}/_{x_{i,j_{id}}}\}$. We show that $\forall i, j,\ 1 \leq i \leq n,\ 1 \leq j \leq l_i\ M_{i,j} \rightarrow nid_{i,j-1}$. By definition of the process D we have that $M_{i_j} =_E \mathtt{snd}(\mathtt{sdec}(ck_{i,j-1}, N_{i,j-1}))$. Hence $M_{i_j} \rightarrow^* nid_{i,j-1}$ if $N_{i,j-1} =_E \mathtt{senc}(ck_{i,j-1}, sr_{i,j-1}, (\text{TMSI\_REALL}, nid_{i,j-1}))$ and
$M_{i_j} \rightarrow^* \mathtt{snd}(\mathtt{sdec}(ck_{i,j-1}, N_{i,j-1}\downarrow))$ otherwise. By contradiction let assume that

$M_{i,j} \neq_E nid_{i,j-1}$ then $N_{i,j-1} \neq_E \mathtt{senc}(ck_{i,j-1}, sr_{i,j-1}, (\text{TMSI\_REALL}, nid_{i,j-1}))$ we have 3 cases:

- $N_{i,j-1} \neq_E \mathtt{senc}(T_{i,j-1}, U_{i,j-1}, L_{i,j-1})$ then
  $\mathtt{fst}(\mathtt{sdec}(ck_{i,j-1}, N_{i,j-1})) \neq_E$ TMSI_REALL and the if check fails so $M_{i,j}$ is not outputted at all;

- $N_{i,j-1} =_E \mathtt{senc}(T_{i,j-1}, U_{i,j-1}, L_{i,j-1})$ and $T_{i,j-1} \neq_E ck_{i,j-1}$ then
  $\mathtt{fst}($
  $sdec(ck_{i,j-1}, N_{i,j-1})) \neq_E$ TMSI_REALL and the if check fails so $M_{i,j}$ is not outputted at all;

- $N_{i,j-1} \neq_E \mathtt{senc}(T_{i,j-1}, U_{i,j-1}, L_{i,j-1})$, $T_{i,j-1} =_E ck_{i,j-1}$ and $L_{i,j-1} \neq_E nid_{i,j-1}$ since $N_{i,j-1}$ is input by the process and by Lemma 7.5
  $\forall i,k \{^{\mathtt{senc}(T_{h,k},U_{h,k},L_{h,k})}/_{y_{h,k}}\}$ we have that $T_{i,k} = ck_{i,j-1}$ if and only if $h = i, k = j-1$ we have that the message
  $N_{i,j-1} =_E \mathtt{senc}(ck_{i,j-1}, sr_{i,j-1}, (\text{TMSI\_REALL}, N_{i,j-1}))$ was constructed by the adversary. This is absurd since $ck_{i,j-1}$ is restricted. Hence, $\forall\, i,j \in \varphi(D)$ we have that $M_{i,j} = nid_{i,j-1}$.

Let $\sigma_{i,j}^{M_{nid}} = \{^{nid_{i,1}}/_{x_{i,2}}\} \mid \cdots \mid \{^{nid_{i,j-1}}/_{x_{i,j}}\}$ we have that $\forall\ i,\ 1 \leq i \leq n,\ \varphi(D_i) = \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,j_{nid}}.(\sigma_{i,j_{id}}^{M} \mid \sigma_{i,j_K}^{K} \mid \sigma_{i,j_{nid}}^{nid}) \equiv \nu\, \widetilde{ms}_{i,l_i}.\nu\, \widetilde{nid}_{i,j_{nid}}.(\{^{id_{1,1}}/_{x_{1,1}}\} \mid \sigma_{i,j_{id}}^{M_{nid}} \mid \sigma_{i,j_K}^{K} \mid \sigma_{i,j_{nid}}^{nid})$. Now that we have defined our frame independently from the process input from its memory (or state) we can automatically verify statical equivalence by using the ProVerif tool. We define the following bi-process which outputs the same terms of our process and hence produces the frames $\varphi(C)$ and $\varphi(D)$:

```
free c.
free d.

fun senc/3.
```

```
reduc sdec(xk, senc(xk, xr, xm)) = xm.

let S = !new id; new nid; new ck; new sr; new mr;
        out(c, choice[nid, id]) |
        out(d, choice[senc(ck, sr, (tmsi_realloc, nid)),
                      senc(ck, sr, (tmsi_realloc, nid))]) |
        out(c, senc(ck, mr, tmsi_complete)).

process S
```

ProVerif can prove the observational equivalence and consequently the statical equivalence of the two frames. □

C

StatVerif Extension Related Proofs

C.1 Proof of Theorem 8.5

Theorem 8.5. *Let A_0 be a closed stateful biprocess. If, for all plain stateful evaluation contexts C and reductions $C[A_0] \to^* A$, the stateful biprocess A is uniform, then A_0 satisfies observational equivalence.*

Proof. Let A_0 be a closed stateful biprocess such that $C[A_0] \to^* B$ always yields a uniform stateful biprocess B, and consider the relation

$$\mathcal{R} = \{(\mathtt{fst}(B), \mathtt{snd}(B)) | C[A_0] \to^* B\}$$

In particular, we have $\mathtt{fst}(A_0)\ \mathcal{R}\ \mathtt{snd}(A_0)$, so we can show that A_0 satisfies observational equivalence by establishing that the relation $\mathcal{R} = \mathcal{R} \cup \mathcal{R}^{-1}$ meets the three conditions of Definition 8.3. By symmetry, we focus on $\mathcal{R}$.
Assume $\mathtt{fst}(B)\ \mathcal{R}\ \mathtt{snd}(B)$.

1. Assume $\mathtt{fst}(B)\downarrow_M$, and let
   $T_M = (\{c\}_\_, \_, \{(\mathtt{in}(M,x);\mathtt{out}(c,c),\emptyset)\}_\_)$ for some fresh name c. We want to test for the ability to send any message on M. We have that for any plain stateful biprocess $P_i = (\mathcal{E}_{p_i}, \mathcal{S}_{p_i}, \mathcal{P}_{p_i})$, $P_i \downarrow_M$ if and only if $T_M[P_i] \rightarrow (\mathcal{E}_{p_i} \cup \{c\}, \mathcal{S}_{p_i}, \mathcal{P}'_{p_i} \cup \{(\mathtt{out}(c,c),\emptyset)\})$ for some $\mathcal{P}'_{p_i}$. In our case, we have that $T_M[\mathtt{fst}(B)] \rightarrow (\mathcal{E}_b \cup \{c\}, \mathtt{fst}(\mathcal{S}_b),$
   $\mathtt{fst}(\mathcal{P}'_b) \cup \{(\mathtt{out}(c,c),\emptyset)\}) = P_1$ for some $\mathtt{fst}(\mathcal{P}'_b)$. Since $C[A_0] \rightarrow^* B$ we have that $T_M[C[A_0]] \rightarrow^* T_M[B]$. Let $C'[\_] = (\mathcal{E}_c \cup \{c\}_\_, \mathcal{S}_{c\_}, \mathcal{P}_c \cup$
   $\{(\mathtt{in}(M,x);\mathtt{out}(c,c),\emptyset)\}_\_) = T_M[C[\_]]$, we have that $C'[A_0] \rightarrow^* T_M[B]$ which by hypothesis is uniform, hence $T_M[B] \rightarrow B'$ with $\mathtt{fst}(B') = P_1$. Since c does not occur in Q (c is fresh) we obtain that $B' = (\mathcal{E}_b \cup \{c\}, \mathcal{S}_b, \mathcal{P}'_b \cup$ $\{(\mathtt{out}(c,c),\emptyset)\})$ for some $\mathcal{P}'_b$, thus
   $T_M[\mathtt{snd}(B)] \rightarrow \mathtt{snd}(B') = (\mathcal{E}_b \cup \{c\}, \mathtt{fst}(\mathcal{S}_b), \mathtt{snd}(\mathcal{P}'_b) \cup \{(\mathtt{out}(c,c),\emptyset)\})$ and $\mathtt{snd}(B)\downarrow_M$

2. if $\mathtt{fst}(B) \rightarrow P_1$ then by uniformity we have that $B \rightarrow B'$ and $fst(B') = P_1$ thus $C[A_0] \rightarrow^* B'$ and by definition of $\mathcal{R}$ we have that $\mathtt{fst}(B')\ \mathcal{R}\ \mathtt{snd}(B')$ and finally by definition of the semantics of stateful biprocess $B \rightarrow B'$ implies $\mathtt{snd}(B) \rightarrow \mathtt{snd}(B')$

3. Let C' be a plain stateful evaluation context. By definition of the semantics of stateful biprocess $C[A_0] \rightarrow^* B$ always implies $C'[C[A_0]] \rightarrow^* C'[B]$ hence $C'[\mathtt{fst}(B)] = \mathtt{fst}(C'[B])\ \mathcal{R}\ \mathtt{snd}(C'[B]) = C'[\mathtt{snd}(B)]$

□

C.2 Proof of Corollary 8.6

Corollary 8.6. *Let A_0 be a closed stateful biprocess. Suppose that, for all plain stateful evaluation contexts C, all stateful evaluation contexts C' , and all*

reductions $C[A_0] \to^* A$,

1. *if* $A = C'[(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathbf{out}(N, M); P, \emptyset), (\mathbf{in}(N', x); Q, \emptyset)\})]$, *then* $\Sigma \vdash \mathbf{fst}(N) = \mathbf{fst}(N')$ *if and only if* $\Sigma \vdash \mathbf{snd}(N) = \mathbf{snd}(N')$,

2. *if* $A = C'[(\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathbf{let}\ x = D\ \mathbf{in}\ P\ \mathbf{else}\ Q, \lambda)\})]$, *then there exists* M_1 *such that* $\mathbf{fst}(D) \Downarrow M_1$ *if and only if there exists* M_2 *such that* $\mathbf{snd}(D) \Downarrow M_2$.

Then A_0 *satisfies observational equivalence.*

Proof. We show that A is uniform, then we conclude by Theorem 8.5. We show that, if $\mathtt{fst}(A) \to A_1$ then there exists a stateful biprocess A' such that $A \to A'$ and $\mathtt{fst}(A') = A_1$. The case for $\mathtt{snd}(A) \to A_2$ is symmetric. By case analysis of the derivation of $\mathtt{fst}(A) \to A_1$, we first show that there exist C, B, and B_1 such that $A = C[B]$, $A_1 = \mathtt{fst}(C)[B_1]$, and $fst(B) \to B_1$ using one of the reduction rules every step in this derivation trivially commutes with $\mathtt{fst}$. For each of these rules, relying on a hypothesis of Corollary 8.6, we find B' such that $\mathtt{fst}(B') = B_1$ and $B \to B'$ using the corresponding biprocess rule:

- (Red Nil): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(0, \lambda)\})$ with $B_1 = (\mathcal{E}, \mathtt{fst}(\mathcal{S}), \mathtt{fst}(\mathcal{P}))$. We take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P})$ so that $\mathtt{fst}(B') = B_1$ and $B \to B'$.

- (Red Repl): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!R, \emptyset)\})$ and $B_1 = (\mathcal{E}, \mathtt{fst}(\mathcal{S}), \mathtt{fst}(\mathcal{P}) \cup \{(\mathtt{fst}(R)\ |!\mathtt{fst}(R), \emptyset)\})$. We take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\ |!R, \emptyset)\})$, so that $\mathtt{fst}(B') = B_1$ and $B \to B'$.

- (Red Par): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R \mid R', \emptyset)\})$ with $B_1 = (\mathcal{E}, \mathtt{fst}(\mathcal{S}), \mathtt{fst}(\mathcal{P}) \cup \{(\mathtt{fst}(R), \emptyset), (\mathtt{fst}(R'), \emptyset)\})$. We take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \emptyset), (R', \emptyset)\})$ so that $\mathtt{fst}(B') = B_1$ and $B \to B'$.

- (Red New): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{new}\ a; R, \lambda)\})$ with $a \notin bn(\texttt{fst}(R))$ (otherwise we rename a) $B_1 = (\mathcal{E} \cup \{a'\}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R)\{a'/a\}, \lambda)\})$ where a' is a fresh name. We take $B' = (\mathcal{E} \cup \{a'\}, \mathcal{S}, \mathcal{P} \cup \{(R\{a'/a\}, \lambda)\})$ so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Fun1): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ R\ \texttt{else}\ R', \lambda)\})$ with $\texttt{fst}(D) \Downarrow M_1$ and $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R)\{M_1/x\}, \lambda)\})$. By hypothesis 2, $\texttt{snd}(D) \Downarrow M_2$ for some M_2. Let $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{\texttt{diff}[M_1, M_2]/x\}$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Fun2): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{let}\ x = D\ \texttt{in}\ R\ \texttt{else}\ R', \lambda)\})$ with no M_1 such that $\texttt{fst}(D) \Downarrow M_1$ and $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R'), \lambda)\})$. By hypothesis 2, there is no M_2 such that $\texttt{snd}(D) \Downarrow M_2$. Let $B' = ((\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \lambda)\})$ we have $B \to B'$.

- (Red Comm): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{out}(N, M); R, \emptyset), (\texttt{in}(N, x); R', \emptyset)\})$ with $\Sigma \vdash \texttt{fst}(N) = \texttt{fst}(N')$ and $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R), \emptyset), (\texttt{fst}(R)\{fst(M)/x\}, \emptyset)\})$. Let $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \emptyset), (R'\{M/x\}, \emptyset)\})$, we have $\texttt{fst}(B') = B_1$ and, by hypothesis 1, $\Sigma \vdash \texttt{snd}(N) = \texttt{snd}(N')$, hence $B \to B'$.

- (Red State): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{([s \mapsto M], \emptyset)\})$ with $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}) \cup \{s \mapsto \texttt{fst}(M)\}, \texttt{fst}(\mathcal{P})\})$. We take $B' = (\mathcal{E}, \mathcal{S} \cup \{s \mapsto M\}, \mathcal{P}\})$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Lock): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; R, \lambda)\})$ with $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R), \lambda \cup \{i_1, \ldots, i_m\})\})$ where $\forall\ (Q, \lambda') \in \texttt{fst}(\mathcal{P}).\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset$. Since by definition $\{\lambda' \mid (Q, \lambda) \in \mathcal{P}\} = \{\lambda' \mid (Q, \lambda) \in \texttt{fst}(\mathcal{P})\}$ (in fact we do not allow to lock different cells in the two stateful biprocesses

variant, i.e. $\texttt{lock}\ \texttt{diff}[s_i, s_j]$ is not allowed) we take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \lambda \cup \{i_1, \ldots, i_m\})\})$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Unlock): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{unlock}\ s_{i_1}, \ldots, s_{i_m}; R, \lambda)\})$ with $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R), \lambda \setminus \{i_1, \ldots, i_m\})\})$ where $\forall\ (Q, \lambda') \in \texttt{fst}(\mathcal{P}).\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset$. We take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \lambda \setminus \{i_1, \ldots, i_m\})\})$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Read): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; R, \lambda)\})$ with $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S}), \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R)\{^{\texttt{fst}(\mathcal{S}(s_{i_1}))}/_{x_1}, \ldots, ^{\texttt{fst}(\mathcal{S}(s_{i_m}))}/_{x_m}\}, \lambda)\ \})$ if $s_{i_1}, \ldots, s_{i_m} \in \mathcal{S}$ and $\forall\ (Q, \lambda') \in \texttt{fst}(\mathcal{P}).\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset$. Since by definition $\{\lambda' \mid (Q, \lambda) \in \mathcal{P}\} = \{\lambda' \mid (Q, \lambda) \in \texttt{fst}(\mathcal{P})\}$ (in fact we do not allow to lock different cells in the two stateful biprocesses variant, i.e. $\texttt{lock}\ \texttt{diff}[s_i, s_j]$ is not allowed) we can take $B' = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{^{\texttt{snd}(\mathcal{S}(s_{i_1}))}/_{x_1}, \ldots, ^{\texttt{snd}(\mathcal{S}(s_{i_m}))}/_{x_m}\}, \lambda)\})$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

- (Red Assign): We have $B = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; R, \lambda)\})$ with $B_1 = (\mathcal{E}, \texttt{fst}(\mathcal{S})[s_{i_1} \mapsto \texttt{fst}(M_1), \ldots, s_{i_m} \mapsto \texttt{fst}(M_m)]\ \texttt{fst}(\mathcal{P}) \cup \{(\texttt{fst}(R), \lambda)\})$ if $s_{i_1}, \ldots, s_{i_m} \in \mathcal{S}$ and $\forall\ (Q, \lambda') \in \texttt{fst}(\mathcal{P}).\ \{i_1, \ldots, i_m\} \cap \lambda' = \emptyset$. Since by definition $\{\lambda' \mid (Q, \lambda) \in \mathcal{P}\} = \{\lambda' \mid (Q, \lambda) \in \texttt{fst}(\mathcal{P})\}$ (in fact we do not allow to lock different cells in the two stateful biprocess variant, i.e. $\texttt{lock}\ \texttt{diff}[s_i, s_j]$ is not allowed) we can take $B' = (\mathcal{E}, \mathcal{S}[s_{i_1} \mapsto M_1, \ldots, s_{i_m} \mapsto M_m], \mathcal{P} \cup \{(R, \lambda)\})$, so that $\texttt{fst}(B') = B_1$ and $B \to B'$.

To conclude, we take the biprocess $A = C[B]$ and the reduction $B \to B'$. □

C.3 Preliminary Lemmas

We recall the following lemmas taken from [BAF05], they are preliminary to some of the later proofs.

Lemma C.1. *Let N be either a name or a variable. If $\Sigma \vdash M = N$ and $nf_{T,\Sigma}(\{M\})$, then $M = N$. For any set of terms $\mathcal{M}$, if $nf_{T,\Sigma}(\mathcal{M})$, then $nf_{T,\Sigma}(\mathcal{M} \cup \{N\})$.*

Lemma C.2. *Let σ be a closed substitution. Let D be a plain term evaluation. If $\sigma D \Downarrow_{\Sigma'} M$, then there exist M', σ_1, and σ'_1 such that $D \Downarrow' (M', \sigma_1), M = \sigma'_1 M'$, and $\sigma = \sigma'_1 \sigma_1$ except on fresh variables introduced in the computation of $D \Downarrow' (M', \sigma_1)$. Let $D_1, \ldots, D_n$ be plain term evaluations. If for all $i \in \{1, \ldots, n\}$, $\sigma D_i \Downarrow'_{\Sigma'} M_i$, then there exist $M'_1, \ldots, M'_n, \sigma_1, and \sigma'_1$ such that $(D_1, \ldots, D_n) \Downarrow' ((M'_1, \ldots, M'_n), \sigma_1)$, $M_i = \sigma'_1 M'_i$ for all $i \in \{1, \ldots, n\}$, and $\sigma = \sigma'_1 \sigma_1$ except on fresh variables introduced in the computation of $(D_1, \ldots, D_n) \Downarrow' ((M'_1, \ldots, M'_n), \sigma_1)$.*

Lemma C.3. *If $h(N_1, \ldots, N_n) \rightarrow N \in def_\Sigma(h), \Sigma \vdash M_i = \sigma N_i$ for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash M = \sigma N$, and $nf_{S,\Sigma}(\{M_1, \ldots, M_n, M\})$, then there exist $h(N'_1, \ldots, N'_n) \rightarrow N' \in def_{\Sigma'}(h)$ and σ' such that $M_i = \sigma' N'_i$ for all $i \in \{1, \ldots, n\}$ and $M = \sigma' N'$.*

Lemma C.4. *Let D be a plain term evaluation. If $D \Downarrow_\Sigma M$, $\Sigma \vdash M = M'$, $\Sigma \vdash D = D'$, and $nf_{S,\Sigma}(\{M', D'\})$, then $D' \Downarrow_\Sigma M'$.*

Lemma C.5. *If $h(N_1, \ldots, N_n) \rightarrow N \in def_{\Sigma'}(h)$ then there exists $h(N'_1, \ldots, N'_n) \rightarrow N' \in def_\Sigma(h)$ and σ such that $\Sigma \vdash N_i = \sigma N'_i$ for all $i \in \{1, \ldots, n\}$ and $\Sigma \vdash N = \sigma N'$.*

Lemma C.6. *Let D be a plain term evaluation. If $\Sigma \vdash D' = D$ and $D' \Downarrow_{\Sigma'} M'$ then $D \Downarrow_\Sigma M$ for some M' such that $\Sigma \vdash M = M'$.*

Lemma C.7. *Let D be a plain term evaluation. If $\Sigma \vdash D' = D$ and $D' \Downarrow_\Sigma M'$ then $D \Downarrow_\Sigma M$ for some M such that $\Sigma \vdash M = M'$.*

C.4 Proof of Lemma 8.8

LEMMA 8.8. *Let A_0 be a closed, unevaluated stateful biprocess. If $A_0 \to^*_\Sigma A'_0, \Sigma \vdash B'_0 = A'_0$, and $nf_{T,\Sigma}(\{B'_0\})$, then $A_0 \to_{\Sigma',\Sigma} B'_0$ by a reduction whose intermediate stateful biprocesses B all satisfy $nf_{T,\Sigma}(\{B\})$.*
Conversely, if $A_0 \to^*_{\Sigma',\Sigma} B'_0$ then there exists A'_0 such that $\Sigma \vdash B'_0 = A'_0$ and $A_0 \to^*_\Sigma A'_0$.

Proof. We write $VC(P)$ when P is a closed stateful biprocess whose terms M are either variables or terms of the form $\mathtt{diff}[M_1, M_2]$ where M_1 and M_2 are closed terms that do not contain `diff`. (Function symbols prefixed by `eval` are not constrained.) We have the following properties:

P1. if $VC(P)$ and $P \to_\Sigma P'$ then $VC(P')$.
The proof is by induction on the derivation of $P \to_\Sigma P'$. The only change of terms is done by the substitution $\{^M/_x\}$ in the rules (RED I/O), (RED FUN 1) and (RED READ). This substitution replaces a variable with a closed term $M = \mathtt{diff}[M_1, M_2]$, hence the result. (For (RED I/O) and (RED READ), M is of the form $\mathtt{diff}[M_1, M_2]$ because of $VC(P)$.)

P2. If $VC(P\{^{\mathtt{diff}[M_1,M_2]}/_x\}), \Sigma \vdash P\{^{\mathtt{diff}[M_1,M_2]}/_x\} = P''$, and $\mathtt{nf}_{T,\Sigma}(\mathcal{Q} \cup \{P''\})$, then by property S2 there exist P', M'_1, and M'_2 such that $\Sigma \vdash P = P', \Sigma \vdash M_1 = M'_1, \Sigma \vdash M_2 = M'_2, P'' = P'\{^{\mathtt{diff}[M_1,M_2]}/_x\}$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P', M'_1, M'_2\})$.

Since A_0 is closed and unevaluated, $VC(A_0)$. Therefore, by P1, if $A_0 \to^*_\Sigma A$, then $VC(A)$. Moreover, the only process A such that $\Sigma \vdash A_0 = A$ and $nf_{T,\Sigma}(\{A\})$ is A_0 by Lemma C.1.

Let us show that, if $VC(A), A \to_\Sigma A', \Sigma \vdash B' = A'$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, then there exists B such that $\Sigma \vdash B = A$, $nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \to_{\Sigma',\Sigma} B'$. The proof is by induction on the derivation of $A \to_\Sigma A'$.

Case (RED COMM): Since $VC(A)$, we have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{out}(\mathtt{diff}[M_1, M_2], N); R, \lambda_1), (\mathtt{in}(\mathtt{diff}[M_1', M_2'], x); R', \lambda_2)\}) \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R, \lambda_1), (R'\{^N/_x\}, \lambda_2)\}) = A'$ with $\Sigma \vdash M_1 = M_1'$ and $\Sigma \vdash M_2 = M_2'$. Since $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, we have $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1, \lambda_1) \mid (R_1'\{^{N_1}/_x\}, \lambda_2)\})$ for some $\mathcal{S}', \mathcal{P}', R_1, R_1', N_1$ such that $\Sigma \vdash \mathcal{P} = \mathcal{P}', \Sigma \vdash \mathcal{S}'(s_1) = \mathcal{S}(s_1), \ldots, \Sigma \vdash \mathcal{S}'(s_n) = \mathcal{S}(s_n), \Sigma \vdash R_1 = R, \Sigma \vdash R_1' = R', \Sigma \vdash N_1 = N$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, R_1', N_1\})$ by P2. By Property S2, there exist M_1'' and M_2'' such that $\Sigma \vdash M_1'' = M_1' = M_1, \Sigma \vdash M_2'' = M_2' = M_2$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, R_1', N_1, M_1'', M_2''\})$. We let $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(\mathtt{out}(\mathtt{diff}[M_1'', M_2''], N_1); R_1, \lambda_1), (\mathtt{in}(\mathtt{diff}[M_1'', M_2''], x); R_1')\})$. Then $\Sigma \vdash B = A$. Moreover $nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$ since $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, R_1', N_1, M_1'', M_2''\})$, and $B \rightarrow_{\Sigma', \Sigma} B'$.

Case (RED FUN1): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{let}\ x = D\ \mathtt{in}\ R\ \mathtt{else}\ R', \lambda)\} \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{^{diff[M,M']}/_x\}, \lambda)\}) = A'$ with $\mathtt{fst}(D) \Downarrow_\Sigma M$ and $\mathtt{snd}(D) \Downarrow_\Sigma M'$. Since $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, we have $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1\{^{diff[M_1,M_1']}/_x\}, \lambda)\})$ for some $\mathcal{S}', \mathcal{P}', R_1, M_1, M_1'$ such that $\Sigma \vdash \mathcal{P} = \mathcal{P}', \Sigma \vdash \mathcal{S}'(s_1) = \mathcal{S}(s_1), \ldots, \Sigma \vdash \mathcal{S}'(s_n) = \mathcal{S}(s_n), \Sigma \vdash R_1 = R, \Sigma \vdash M_1 = M, \Sigma \vdash M_1' = M'$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, M_1, M_1'\})$ by P2. By Property S2, there exist D_1 and R_1' such that $\Sigma \vdash D_1 = D, \Sigma \vdash R_1' = R$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), D_1, R_1, R_1', M_1, M_1'\})$. By Lemma C.4, $\mathtt{fst}(D_1) \Downarrow_{\Sigma'} M_1$ and $\mathtt{snd}(D_1) \Downarrow_{\Sigma'} M_1'$. Let $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(\mathtt{let}\ x = D_1\ \mathtt{in}\ R_1\ \mathtt{else}\ R_1', \lambda)\}$. Then $\Sigma \vdash B = A, nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \rightarrow_{\Sigma', \Sigma} B'$.

Case (RED FUN2): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{let}\ x = D\ \mathtt{in}\ R\ \mathtt{else}\ R', \lambda)\} \rightarrow_\Sigma A'$, there exists no M such that $\mathtt{fst}(D) \Downarrow_\Sigma M$, and there exists no M' such that $\mathtt{snd}(D) \Downarrow_\Sigma M'$. We have $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$. By Property S2, there exist D_1 and R_1 such that $\Sigma \vdash D_1 = D, \Sigma \vdash R_1 = R$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{R_1, D_1, B'\})$. Then, there exists no M such that $\mathtt{fst}(D_1) \Downarrow_\Sigma M$, and there exists no M' such that $\mathtt{snd}(D_1) \Downarrow_\Sigma M'$. (Otherwise, by Lemma C.7, there

would exist M such that $\texttt{fst}(D) \Downarrow_\Sigma M$, and M' such that $\texttt{snd}(D) \Downarrow_\Sigma M'$). Let $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(\texttt{let } x = D_1 \texttt{ in } R_1 \texttt{ else } Q', \lambda)\})$. Then $\Sigma \vdash B = A, nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \rightarrow_{\Sigma',\Sigma} B'$.

Case (Red Repl): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!R, \emptyset)\}) \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R \mid !R, \emptyset)\}) = A'$. Since $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, we have $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1 \mid !R_1, \lambda)\})$ for some $\mathcal{S}', \mathcal{P}', R_1$ such that $\Sigma \vdash \mathcal{P} = \mathcal{P}', \Sigma \vdash \mathcal{S}'(s_1) = \mathcal{S}(s_1), \ldots, \Sigma \vdash \mathcal{S}'(s_n) = \mathcal{S}(s_n), \Sigma \vdash R_1 = R$. Let $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(!R_1, \lambda)\})$. Then we have $\Sigma \vdash B = A, nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \rightarrow_{\Sigma',\Sigma} B'$.

Case (Red Read): We have
$A = (E, \mathcal{S}, \mathcal{P} \cup \{(\texttt{read } s_{j_1}, \ldots, s_{j_m} \texttt{ as } x_1, \ldots, x_m; R, \lambda)\}) \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{^{\mathcal{S}(s_{j_1})}/_{x_1}, \ldots, ^{\mathcal{S}(s_{j_m})}/_{x_m}\}, \lambda)\}) = A'$. Since $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, we have $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1\{^{M_1}/_{x_1}, \ldots, ^{M_m}/_{x_m}\}, \lambda)\})$ for some $\mathcal{S}', \mathcal{P}', R_1, M_1, \ldots M_m$ such that $\Sigma \vdash \mathcal{P} = \mathcal{P}', \Sigma \vdash \mathcal{S}'(s_1) = \mathcal{S}(s_1), \ldots, \Sigma \vdash \mathcal{S}'(s_n) = \mathcal{S}(s_n), \Sigma \vdash R_1 = R, \Sigma \vdash M_1 = \mathcal{S}(s_{j_1}), \ldots, \Sigma \vdash M_m = \mathcal{S}(s_{j_m})$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, M_1, \ldots, M_m\})$ by P2. Let $B = (E, \mathcal{S}'[j_k \mapsto M_k \mid 1 \leq k \leq m], \mathcal{P}' \cup \{(\texttt{read } s_{j_1} \ldots, s_{j_m} \texttt{ as } x_1, \ldots, x_m; R_1, \lambda)\})$. Then $\Sigma \vdash B = A, nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \rightarrow_{\Sigma',\Sigma} B'$.

Case (Red Assign): We have $A = (E, \mathcal{S}, \mathcal{P} \cup \{(s_{j_1}, \ldots, s_{j_m} := M_1, \ldots, M_m; R, \lambda)\}) \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}[j_k \mapsto M_k \mid 1 \leq k \leq m], \mathcal{P} \cup \{(R, \lambda)\}) = A'$. Since $\Sigma \vdash B' = A'$ and $nf_{T,\Sigma}(\mathcal{Q} \cup \{B'\})$, we have $B' = (\mathcal{E}, \mathcal{S}'[j_k \mapsto M'_k \mid 1 \leq k \leq m], \mathcal{P}' \cup \{(R_1, \lambda)\})$ for some $R_1, M'_1, \ldots M'_m$ such that $\Sigma \vdash \mathcal{P} = \mathcal{P}', \Sigma \vdash \mathcal{S}'(s_1) = \mathcal{S}(s_1), \ldots, \Sigma \vdash \mathcal{S}'(s_n) = \mathcal{S}(s_n), \Sigma \vdash R_1 = R, \Sigma \vdash M_1 = M'_1, \ldots, \Sigma \vdash M_m = M'_m$, and $nf_{T,\Sigma}(\mathcal{Q} \cup \{P' \mid (P', \lambda) \in \mathcal{P}'\} \cup \{\mathcal{S}'(s_1), \ldots, \mathcal{S}'(s_n), R_1, M'_1, \ldots, M'_m\})$. Let $B = (E, \mathcal{S}', \mathcal{P}' \cup \{(s_{j_1} \ldots, s_{j_m} := M'_1, \ldots, M'_m; R_1, \lambda)\})$. Then $\Sigma \vdash B = A, nf_{T,\Sigma}(\mathcal{Q} \cup \{B\})$, and $B \rightarrow_{\Sigma',\Sigma} B'$.

Cases (Red Par),(Red New), (Red State), (Red Lock), and (Red Unlock): Easy by induction hypothesis.

Therefore, if $A_0 \rightarrow^* A_0, \Sigma \vdash B'_0 = A'_0$, and $nf_{T,\Sigma}(\{B'_0\})$, then there exists B_0 such that $nf_{T,\Sigma}(\{B_0\}), \Sigma \vdash B_0 = A_0$, and $B_0 \rightarrow_{\Sigma',\Sigma} B'_0$ by a reduction whose intermediate biprocesses B all satisfy $nf_{T,\Sigma}(\{B\})$, simply by applying several times the results shown above. Since the only process A such that $\Sigma \vdash A_0 = A$ and $nf_{T,\Sigma}(\{A\})$ is A_0, we have $B_0 = A_0$, so we conclude that if $A_0 \rightarrow^* A'_0, \Sigma \vdash B'_0 = A'_0$, and $nf_{T,\Sigma}(\{B'_0\})$, then $A_0 \rightarrow_{\Sigma',\Sigma} B'_0$ by a reduction whose intermediate biprocesses B all satisfy $nf_{T,\Sigma}(\{B\})$.

For the converse, we show that, if $\Sigma \vdash B = A$, then there exists B' such that, if $VC(A)$, $A \rightarrow_{\Sigma',\Sigma} A'$ and $\Sigma \vdash B = A$, then there exists B' such that $\Sigma \vdash B' = A'$, and $B \rightarrow_{\Sigma} B'$. The proof is by induction on the derivation of $A \rightarrow_{\Sigma',\Sigma} A'$.

Case (Red Comm): Since $VC(A)$, we have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(out(\mathtt{diff}[M_1, M_2], N); R, \lambda), (in(\mathtt{diff}[M_1, M_2], x; R', \lambda')\}) \rightarrow_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{R, \lambda), (R'\{^N/_x\}, \lambda')\}) = A'$. Since $\Sigma \vdash A = B$, we have $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(out(\mathtt{diff}[M'_1, M'_2], N'); R_1, \lambda), (in(\mathtt{diff}[M''_1, M''_2], x; R'_1, \lambda')\})$ with $\Sigma \vdash M_1 = M'_1 = M''_1$, $\Sigma \vdash M_2 = M'_2 = M''_2$, $\Sigma \vdash N = N'$, $\Sigma \vdash R = R_1$, $\Sigma \vdash R' = R'_1$. Then $B \rightarrow_{\Sigma} B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1, \lambda), (R'_1\{^{N'}/_x\}, \lambda')\})$ with $\Sigma \vdash B' = A'$

Case (Red Fun1): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{let}\ x = D\ \mathtt{in}\ R\ \mathtt{else}\ R', \lambda)\}) \rightarrow_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{^{\mathtt{diff}[M_1,M_2]}/_x\}, \lambda)\}) = A'$ with $\mathtt{fst}(D) \Downarrow_{\Sigma} M_1$ and $\mathtt{snd}(D) \Downarrow_{\Sigma} M_2$. Since $\Sigma \vdash A = B$, we have $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(\mathtt{let}\ x = D'\ \mathtt{in}\ R_1\ \mathtt{else}\ R'_1, \lambda)\})$ with $\Sigma \vdash D = D'$, $\Sigma \vdash R = R_1$, $\Sigma \vdash R' = R'_1$. By Lemma C.6, $\mathtt{fst}(D') \Downarrow_{\Sigma} M'_1$ with $\Sigma \vdash M_1 = M'_1$ and $\mathtt{snd}(D') \Downarrow M'_2$ with $\Sigma \vdash M_2 = M'_2$. Hence $B \rightarrow_{\Sigma} B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1\{^{diff[M'_1,M'_2]}/_x\}, \lambda)\})$ with $\Sigma \vdash B = A$.

Case (Red Fun2): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(\mathtt{let}\ x = D\ \mathtt{in}\ R\ \mathtt{else}\ R', \lambda)\}) \rightarrow_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R', \lambda)\}) = A'$ and there exists no M_1 such that $\mathtt{fst}(D) \Downarrow_{\Sigma} M_1$, and there exists no M_2 such that $\mathtt{snd}(D) \Downarrow_{\Sigma} M_2$. Since $\Sigma \vdash A = B$, we have $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(\mathtt{let}\ x = D'\ \mathtt{in}\ R_1\ \mathtt{else}\ R'_1, \lambda)\})$ with $\Sigma \vdash D = D'$, $\Sigma \vdash R = R_1$ and $\Sigma \vdash R' = R'_1$. Then there exists no M'_1 such that $\mathtt{fst}(D') \Downarrow_{\Sigma} M'_1$,

and there exists no M_2' such that $\texttt{snd}(D') \Downarrow_\Sigma M_2'$ (otherwise, by Lemma C.7, there would exist M_1 such that $\texttt{fst}(D) \Downarrow_\Sigma M_1$, and there exists no M_2 such that $\texttt{snd}(D) \Downarrow_\Sigma M_2$. Hence $B \rightarrow_\Sigma B'$ and $\Sigma \vdash B' = A'$.

Case (RED REPL): We have $A = (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(!R, \emptyset)\}) \rightarrow_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R \mid !R, \emptyset)\}) = A'$. Since $\Sigma \vdash A = B$, we have $B = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(!R_1, \emptyset)\})$ with $\Sigma \vdash R_1 = R$. Let $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1 \mid !R_1, \emptyset)\})$. So $\Sigma \vdash A' = B'$ and $B \rightarrow_\Sigma B'$.

Case (RED READ): We have
$A = (E, \mathcal{S}, \mathcal{P} \cup \{(\texttt{read } s_{j_1}, \ldots, s_{j_m} \texttt{ as } x_1, \ldots, x_m; R, \lambda)\}) \rightarrow_{\Sigma',\Sigma} (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(R\{^{\mathcal{S}(s_{j_1})}/_{x_1}, \ldots, ^{\mathcal{S}(s_{j_m})}/_{x_m}\}, \lambda)\}) = A'$. Since $\Sigma \vdash B' = A'$, we have $B = (E, \mathcal{S}', \mathcal{P}' \cup \{(\texttt{read } s_{j_1}, \ldots, s_{j_m} \texttt{ as } x_1, \ldots, x_m; R_1, \lambda)\})$ with $\Sigma \vdash R_1 = R$ and $\Sigma \vdash \mathcal{S}'(s_{j_1}) = \mathcal{S}(s_{j_1}), \ldots, \Sigma \vdash \mathcal{S}(s_{j_m}) = \mathcal{S}(s_{j_m})$. Let $B' = (\mathcal{E}, \mathcal{S}', \mathcal{P}' \cup \{(R_1\{^{\mathcal{S}'(s_{j_1})}/_{x_1}, \ldots, ^{\mathcal{S}'(s_{j_m})}/_{x_m}\}, \lambda)\})$. So $\Sigma \vdash A' = B'$ and $B \rightarrow_\Sigma B'$.

Case (RED ASSIGN): We have $A = (E, \mathcal{S}, \mathcal{P} \cup \{(s_{j_1}, \ldots, s_{j_m} := M_1, \ldots, M_m; R, \lambda)\}) \rightarrow_\Sigma (\mathcal{E}, \mathcal{S}[j_k \mapsto M_k \mid 1 \leq k \leq m], \mathcal{P} \cup \{(R, \lambda)\}) = A'$. Since $\Sigma \vdash B = A$, we have $B = (E, \mathcal{S}', \mathcal{P}' \cup \{(s_{j_1}, \ldots, s_{j_m} := M_1', \ldots, M_m'; R_1, \lambda)\})$ with $\Sigma \vdash R_1 = R$, $\Sigma \vdash M_1' = M_1, \ldots, \Sigma \vdash M_m' = M_m$ and $\Sigma \vdash \mathcal{S}'(s_{j_1}) = \mathcal{S}(s_{j_1}), \ldots, \Sigma \vdash \mathcal{S}(s_{j_m}) = \mathcal{S}(s_{j_m})$. Let $B' = (\mathcal{E}, \mathcal{S}'[j_k \mapsto M_k' \mid 1 \leq k \leq m], \mathcal{P}' \cup \{(R_1, \lambda)\})$. So $\Sigma \vdash A' = B'$ and $B \rightarrow_\Sigma B'$.

Cases (RED PAR), (RED NEW), (RED STATE), (RED LOCK) and (RED UNLOCK): Easy by induction hypothesis.

We conclude that, if $A_0 \rightarrow_{\Sigma',\Sigma}^* B_0'$ then there exists A_0' such that $\Sigma \vdash B_0' = A_0'$ and $A_0 \rightarrow_\Sigma^* A_0'$, simply by applying several times the results shown above, with $B = A$ in the first application. □

C.5 Proof of Lemma 8.9

Following the proof method used in [BAF05] we first show that it is enough to consider unevaluated processes as initial configurations (Lemma C.10), and then prove Lemma 8.9 itself.

Let $A \mathcal{R} A'$ if and only if $A = (\mathcal{E}, \mathcal{S}, \mathcal{P}), A' = (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ and $\mathcal{P}'$ is obtained from $\mathcal{P}$ by adding some lets on terms with constructors that occur in inputs or outputs (for instance transforming $\mathtt{out}(M, N); P$ into $\mathtt{let}\ x = M\ \mathtt{in}\ \mathtt{let}\ y = N\ \mathtt{in}\ \mathtt{out}(x, y); P$ where x and y are fresh variables), prefixing some constructors in lets with eval, and replacing some terms M with $\mathtt{diff}[\mathtt{fst}(M), \mathtt{snd}(M)]$.
Similarly, we write $P \mathcal{R} P'$ if and only if $\mathcal{P}'$ is obtained from $\mathcal{P}$ by adding some lets on terms with constructors that occur in inputs or outputs (for instance transforming $\mathtt{out}(M, N); P$ into $\mathtt{let}\ x = M\ \mathtt{in}\ \mathtt{let}\ y = N\ \mathtt{in}\ \mathtt{out}(x, y); P$ where x and y are fresh variables), prefixing some constructors in lets with eval, and replacing some terms M with $\mathtt{diff}[\mathtt{fst}(M), \mathtt{snd}(M)]$.

Lemma C.8. *If $A \mathcal{R} B$ and $A \to_\Sigma A'$ then there exists B' such that $A' \mathcal{R} B'$ and $B \to_\Sigma^* B'$.*

Proof. Obvious, by induction on the derivation of $A \to A'$. □

Lemma C.9. *If $\Sigma \vdash A = B, B \mathcal{R} R$, and $R \to_\Sigma R'$ then there exists A' and B' such that $\Sigma \vdash A' = B', B' \mathcal{R} R'$, and $A \to_\Sigma A'$ or $A = A'$.*

Proof. Obvious, by induction on the derivation of $R \to R'$. □

Lemma C.10. *Let A_0 be a closed stateful biprocess. The hypotheses of Corollary 8.6 are true if and only if they are true with $\mathtt{uneval}(C[A_0])$ instead of $C[A_0]$.*

Proof. We have $C[A_0] \mathcal{R}\ \mathtt{uneval}(C[A_0])$. We first show that if the hypotheses of Corollary 8.6 are true for $\mathtt{uneval}(C[A_0])$, then they are true for $C[A_0]$.

- If $C[A_0] \rightarrow_\Sigma^* C_1'[(\emptyset, \emptyset, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N_1, x); R_1, \lambda)\})]$, then by Lemma C.8, we have $\texttt{uneval}(C[A_0]) \rightarrow_\Sigma^* A'$ with $C_1'[(\emptyset, \emptyset, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N_1, x); R_1, \lambda)\})]\mathcal{R}A'$. Then we have $A' \rightarrow_\Sigma^* C'[(\emptyset, \emptyset, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ with $C_1' \ \mathcal{R} \ C'$, $\texttt{fst}(N) = \texttt{fst}(N_1)$, $\texttt{snd}(N) = \texttt{snd}(N_1)$, $\texttt{fst}(N') = \texttt{fst}(N_1'), \texttt{snd}(N') = \texttt{snd}(N_1'), \texttt{fst}(M) = \texttt{fst}(M_1)$, $\texttt{snd}(M) = \texttt{snd}(M_1), Q_1 \ \mathcal{R} \ Q$, and $R_1 \ \mathcal{R} \ R$, by reducing the term evaluations of constructors that may occur above inputs and outputs in A. So $\texttt{uneval}(C[A_0]) \rightarrow_\Sigma^* C'[(\emptyset, \emptyset, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$, with $\texttt{fst}(N) = \texttt{fst}(N_1)$, $\texttt{snd}(N) = \texttt{snd}(N_1), \texttt{fst}(N') = \texttt{fst}(N_1')$, and $\texttt{snd}(N') = \texttt{snd}(N_1')$. Hence, if the first hypothesis of Corollary 8.6 is true with $\texttt{uneval}(C[A_0])$, then it is true with $C[A_0]$.

- If $C[A_0] \rightarrow_\Sigma^* C_1'[(\emptyset, \emptyset, \{(\texttt{let } y_1 = D_1 \texttt{ in } Q_1 \texttt{ else } R_1, \lambda)\})]$, then by the same reasoning as above, $\texttt{uneval}(C[A_0]) \rightarrow_\Sigma^* C'[A]$ where $(\emptyset, \emptyset, \{(\texttt{let } y_1 = D_1 \texttt{ then } Q_1 \texttt{ else } R_1, \lambda)\}) \ \mathcal{R} \ A$. Hence, we have $A = (\emptyset, \emptyset, \{(\texttt{let } y_1 = D_1' \texttt{ then } Q_1' \texttt{else} R_1', \lambda)\})$ where D_1' is obtained by prefixing some constructors of D_1 with `eval` and reorganizing `diffs`. We have $\texttt{fst}(D_1) \Downarrow_\Sigma M_1$ if and only if $\texttt{fst}(D_1') \Downarrow_\Sigma M_1$, if and only if $\texttt{snd}(D_1') \Downarrow_\Sigma M_2$ (by the second hypothesis of Corollary 8.6 for $\texttt{uneval}(C[A_0])$), if and only if $snd(D_1) \Downarrow_\Sigma M_2$. This yields the second hypothesis of Corollary 8.6 for $C[A_0]$.

We now show the converse: if the hypotheses of Corollary 8.6 are true for $C[A_0]$, then they are true for $\texttt{uneval}(C[A_0])$.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow_\Sigma^* C_1'[(\emptyset, \emptyset, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N_1, x); R_1, \lambda)\})]$. By Lemma C.9, $C[A_0] \rightarrow_\Sigma^* A$ with $\Sigma \vdash A = A'$ and $A'\mathcal{R}C_1'[(\emptyset, \emptyset, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N_1, x); R_1, \lambda)\})]$. Then $A = C'[(\emptyset, \emptyset, \{(\texttt{out}(N, M); Q_1 \mid \texttt{in}(N', x); R, \lambda)\})]$ with $\Sigma \vdash \texttt{fst}(N) = \texttt{fst}(N_1), \Sigma \vdash \texttt{snd}(N) = \texttt{snd}(N_1), \Sigma \vdash \texttt{fst}(N') = \texttt{fst}(N_1'), \Sigma \vdash \texttt{snd}(N') = \texttt{snd}(N_1'), \Sigma \vdash \texttt{fst}(M) = \texttt{fst}(M_1)$, and $\Sigma \vdash \texttt{snd}(M) = \texttt{snd}(M_1)$. So, if the first hypothesis of Corollary 8.6 is true with $C[A_0]$, then it is true with $\texttt{uneval}(C[A_0])$.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow_\Sigma^* C_1'[(\emptyset, \emptyset, \{(\texttt{let}\ y_1 = D_1\ \texttt{in}\ Q_1\ \texttt{else}\ R_1, \lambda)\})]$. By Lemma C.9, $C[A_0] \rightarrow_\Sigma^* A$ with $\Sigma \vdash A = A'$, and $A'\ \mathcal{R}\ C_1'[(\emptyset, \emptyset, \{(\texttt{let}\ y_1 = D_1\ \texttt{in}\ Q_1\ \texttt{else}\ R_1, \lambda)\})]$. We have two cases:
  - Case 1: $\texttt{let}\ y_1$ is introduced by $\mathcal{R}$. Then $R_1 = 0$ and D_1 does not contain destructors. Hence there exists M_1 such that $\texttt{fst}(D_1) \Downarrow_\Sigma M_1$ and there exists M_2 such that $\texttt{snd}(D_1) \Downarrow_\Sigma M_2$.
  - Case 2: $\texttt{let}\ y_1$ comes from A. Hence $A = C'[(\emptyset, \emptyset, \{(\texttt{let}\ y_1 = D_1'\ \texttt{then}\ Q_1'\ \texttt{else}\ R_1', \lambda)\})]$ where D_1' is obtained by removing some $\texttt{eval}$ prefixes of D_1, reorganizing $\texttt{diff}s$, and replacing terms with equal terms modulo Σ. We have $\texttt{fst}(D_1) \Downarrow_\Sigma M_1$ for some M_1 if and only if $\texttt{fst}(D_1') \Downarrow_\Sigma M_1$ for some M_1, if and only if $\texttt{snd}(D_1') \Downarrow_\Sigma M_2$ for some M_2 (by the second hypothesis of Corollary 8.6 for $C[A_0]$), if and only if $\texttt{snd}(D_1) \Downarrow_\Sigma M_2$ for some M_2.

This yields the second hypothesis of Corollary 8.6 for $\texttt{uneval}(C[A_0])$. □

Lemma 8.9. *Let A_0 be a closed stateful biprocess. Suppose that, for all plain evaluation stateful contexts C, all evaluation stateful contexts C', and all reductions $\texttt{uneval}(C[A_0]) \rightarrow_{\Sigma',\Sigma}^* A$ whose intermediate biprocesses A' all satisfy $nf_{T,\Sigma}(\{A'\})$,*

1. *if $A = C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $\texttt{fst}(N) = \texttt{fst}(N')$, then $\Sigma \vdash \texttt{snd}(N) = snd(N')$,*
2. *if $A = C'[(\emptyset, \mathcal{S}, \{(\texttt{let}\ x = D\ \texttt{in}\ Q\ \texttt{else}\ R, \lambda)\})]$ and $\texttt{fst}(D) \Downarrow_{\Sigma'} M_1$ for some M_1, then $\texttt{snd}(D) \Downarrow_\Sigma M_2$ for some M_2,*

as well as the symmetric properties where we swap $\texttt{fst}$ and $\texttt{snd}$. Then A_0 satisfies the hypotheses of Corollary 8.6.
Conversely, if A_0 satisfies the hypotheses of Corollary 8.6, then for all plain evaluation stateful contexts C, evaluation stateful contexts C' , and reductions

$\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma'} A$, we have properties 1 and 2 above, as well as the symmetric properties where we swap $\texttt{fst}$ and $\texttt{snd}$.

Proof. By Lemma C.10, we can work with $\texttt{uneval}(C[A_0])$ instead of $C[A_0]$. We show the two hypotheses of Corollary 8.6.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow^*_\Sigma C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $\Sigma \vdash \texttt{fst}(N) = \texttt{fst}(N')$. By Property S2, there exists A' such that $\Sigma \vdash A' = C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $nf_{T,\Sigma}(\{A'\})$. By Lemma 8.8, $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} A'$.Moreover, $A' = C''[(\emptyset, \mathcal{S}, \{(\texttt{out}(\texttt{diff}[N_1, N_2], M'); Q_1 \mid \texttt{in}(\texttt{diff}[N_1', N_2'], x); R_1, \lambda)\})]$, where $\Sigma \vdash N_1 = \texttt{fst}(N), \Sigma \vdash N_2 = \texttt{snd}(N), \Sigma \vdash N_1' = fst(N')$, and $\Sigma \vdash N_2' = \texttt{snd}(N')$. Since $nf_{T,\Sigma}(\{A'\}), N_1 = N_1'$. Hence, by hypothesis 1, $\Sigma \vdash N_2 = N_2'$. So $\Sigma \vdash snd(N) = snd(N')$.

  We obtain the case $\texttt{uneval}(C[A_0]) \rightarrow^*_\Sigma C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $\Sigma \vdash \texttt{snd}(N) = \texttt{snd}(N')$ by symmetry.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow^*_\Sigma C'[(\emptyset, \mathcal{S}, \{\texttt{let } y = D \texttt{ in } Q \texttt{ else } R, \lambda)\})]$ and there exists M_1 such that $\texttt{fst}(D) \Downarrow_\Sigma M_1$. By Property S2, there exist A', M_1', and D' such that $\Sigma \vdash A' = C'[(\emptyset, \mathcal{S}, \{\texttt{let } y = D \texttt{ in } Q \texttt{ else } R, \lambda)\})], \Sigma \vdash M_1 = M_1', \Sigma \vdash D = D'$, and $nf_{T,\Sigma}(\{A', M_1', D'\})$. Then $A' = C''[(\emptyset, \mathcal{S}, \{\texttt{let } y = D' \texttt{ in } Q' \texttt{ else } R', \lambda)\})]$.
  By Lemma 8.8, $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} A'$. By Lemma C.4, $\texttt{fst}(D') \Downarrow_\Sigma M_1'$. By hypothesis 2, $\texttt{snd}(D') \Downarrow_\Sigma M_2'$. By Lemma C.7, since $\Sigma \vdash \texttt{snd}(D) = \texttt{snd}(D')$ and $\texttt{snd}(D') \Downarrow_\Sigma M_2'$, we have $\texttt{snd}(D) \Downarrow_\Sigma M_2$.

  We obtain the case $\texttt{uneval}(C[A_0]) \rightarrow^*_\Sigma C'[(\emptyset, \mathcal{S}, \{\texttt{let } y = D \texttt{ in } Q \texttt{ else } R, \lambda)\})]$ and there exists M_2 such that $\texttt{snd}(D) \Downarrow_\Sigma M_2$ by symmetry.

Next, we prove the converse property.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $\texttt{fst}(N) = fst(N')$. By Lemma 8.8, we have $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma} C'_1[(\emptyset, \mathcal{S}, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N'_1, x); R_1, \lambda)\})]$ with $\Sigma \vdash C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})] = C'_1[(\emptyset, \mathcal{S}, \{(\texttt{out}(N_1, M_1); Q_1 \mid \texttt{in}(N'_1, x); R_1, \lambda)\})]$ so $\Sigma \vdash N = N_1$ and $\Sigma \vdash N' = N'_1$. Using the first hypothesis of Corollary 8.6, since $\Sigma \vdash \texttt{fst}(N_1) = \texttt{fst}(N'_1)$, we have $\Sigma \vdash \texttt{snd}(N_1) = \texttt{snd}(N'_1)$, hence $\Sigma \vdash \texttt{snd}(N) = \texttt{snd}(N')$.

  We obtain the case $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} C'[(\emptyset, \mathcal{S}, \{(\texttt{out}(N, M); Q \mid \texttt{in}(N', x); R, \lambda)\})]$ and $\Sigma \vdash \texttt{snd}(N) = \texttt{snd}(N')$ by symmetry.

- Assume that $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} C'[(\emptyset, \mathcal{S}, \{\texttt{let } y = D \texttt{ in } Q \texttt{ else } R, \lambda)\})]$ and there exists M_1 such that $\texttt{fst}(D) \Downarrow_{\Sigma'} M_1$. As above, $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma} C'_1[(\emptyset, \mathcal{S}, \{\texttt{let } y = D_1 \texttt{ in } Q_1 \texttt{ else } R_1, \lambda)\})]$ with $\Sigma \vdash D_1 = D$. By Lemma C.6, $\texttt{fst}(D_1) \Downarrow_{\Sigma} M'_1$ for some M'_1. Using the second hypothesis of Corollary 8.6, $\texttt{snd}(D_1) \Downarrow_{\Sigma} M'_2$, hence by Lemma C.7, $\texttt{snd}(D) \Downarrow_{\Sigma} M_2$.

  We obtain the case $\texttt{uneval}(C[A_0]) \rightarrow^*_{\Sigma',\Sigma} C'[(\emptyset, \mathcal{S}, \{\texttt{let } y = D \texttt{ in } Q \texttt{ else } R, \lambda)\})]$ and there exists M_2 such that $\texttt{snd}(D) \Downarrow_{\Sigma'} M_2$ by symmetry.

□

C.6 Results Preliminary to the Proof of Theorem 8.11

As in [BAF05] we use the following definitions to aid the proof of theorem 8.11 Let $\mathcal{F}$ be a set containing patterns, facts, sequences of patterns or facts, clauses, environments that map variables and names to pairs of patterns, ..., we say that $nf_{T,\Sigma}(\mathcal{F})$ if and only if all patterns that appear in $\mathcal{F}$ are irreducible by T and for all p_1, p_2 sub-patterns of elements of $\mathcal{F}$, if $\Sigma \vdash p_1 = p_2$ then $p_1 = p_2$.

We say that $nf'_{T,\Sigma}(\mathcal{F})$ if and only if $nf_{T,\Sigma}(\mathcal{F}')$ where $\mathcal{F}'$ is obtained from $\mathcal{F}$ by removing nounif facts. Let $\mathcal{D}$ be a derivation, we say that $nf'_{T,\Sigma}(\mathcal{D})$ when $nf_{T,\Sigma}(\mathcal{F})$ where $\mathcal{F}$ is the set of intermediately derived facts of $\mathcal{D}$.

We say that $F_1 \wedge \cdots \wedge F_n \sim F'_1 \wedge \cdots \wedge F'_n$ when, for all $i \in \{1, \ldots, n\}$, either $F_i = F'_i$ or F_i and F'_i are *nounif* facts and $\Sigma \vdash F_i = F'_i$. We say that $\Sigma \vdash F_1 \wedge \cdots \wedge F_n \sim F'_1 \wedge \cdots \wedge F'_n$ when for all $i \in \{1, \ldots, n\}, \Sigma \vdash F_i = F'_i$. This definition is naturally extended to clauses. The special treatment of nounif facts in the definition of $\sim$ and in Lemma C.17 is necessary so that the following results hold. In particular, Lemma C.15 would be wrong for Clauses (Rt) and (Rt'), which contain nounif facts.

We also use the following lemmas which have been established in [BAF05]:

Lemma C.11. *If $h(N_1, \ldots, N_n) \to N$ is in $def_{\Sigma'}(h), \Sigma \vdash N'' = \sigma N, \Sigma \vdash N''_i = \sigma N_i$ for all $i \in \{1, \ldots, n\}$, and $nf_{T,\Sigma}(\{N''_1, \ldots, N''_n, N''\})$, then there exist a closed substitution σ' and $h(N'_1, \ldots, N'_n) \to N'$ in $def_{\Sigma'}(h)$ such that $N'' = \sigma' N'$ and $N''_i = \sigma' N'_i$ for all $i \in \{1, \ldots, n\}$.*

Lemma C.12. *Let D be a plain term evaluation. If $D \Downarrow' (p, \sigma)$ and σ' is a closed substitution, then there exists p' such that $\sigma'\sigma D \Downarrow_\Sigma p'$ and $\Sigma \vdash p' = \sigma' p$.*

Let $D_1, \ldots, D_n$ be plain term evaluations. If $(D_1, \ldots, D_n) \Downarrow' ((p_1, \ldots, p_n), \sigma)$ and σ is a closed substitution then there exist $p'_1, \ldots, p'_n$ such that for all $i \in \{1, \ldots, n\}, \sigma'\sigma D_i \Downarrow_\Sigma p'_i$ and $\Sigma \vdash p'_i = \sigma' p_i$.

Lemma C.13. *Let D be a plain term evaluation such that the subterms M of D are variables or names. If $\rho(D) \Downarrow' (p', \sigma'), \sigma$ is a closed substitution, $\Sigma \vdash p = \sigma p', \Sigma\sigma'_0\rho' = \sigma\sigma'\rho$, and $nf_{T,\Sigma}(\{p, \sigma'_0\rho'\})$, then there exist $\sigma'', p'', \sigma''_0$ such that $\rho'(D) \Downarrow' (p'', \sigma''), \sigma'_0 = \sigma''_0\sigma''$ except on fresh variables introduced in the computation of $\rho'(D) \Downarrow' (p'', \sigma'')$, and $p = \sigma''_0 p''$.*

Let $D_i (i \in \{1, \ldots, n\})$ be plain term evaluations such that the subterms M of D_i are variables or names. If $(\rho(D_1), \ldots, \rho(D_n)) \Downarrow' ((p'_1, \ldots, p'_n), \sigma')$, σ is a closed substitution, $\Sigma \vdash p_i = \sigma p'_i$ for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash \sigma'_0 \rho' = \sigma\sigma'\rho$, and $nf_{T,\Sigma}(\{p_1, \ldots, p_n, \sigma'_0\rho'\})$, then there exist $\sigma'', p''_1, \ldots, p''_n, \sigma''_0$ such that $(\rho'(D_1), \ldots, \rho'(D_n)) \Downarrow' ((p''_1, \ldots, p''_n), \sigma'')$, $\sigma'_0 = \sigma''_0\sigma''$ except on fresh variables introduced in the computation of $(\rho'(D_1), \ldots, \rho'(D_n)) \Downarrow' ((p''_1, \ldots, p''_n), \sigma'')$, and $p_i = \sigma''_0 p''_i$ for all $i \in \{1, \ldots, n\}$.

Further, we extend some of the results presented in [BAF05] to our stateful biprocesses and their translation in horn clauses.

Lemma C.14. *Let A_0 be a closed, unevaluated stateful process such that $A_0 = (\mathcal{E}_0, \mathcal{S}_0, \{(P_0, \lambda_0)\})$. If $[\![P]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called during the generation of $[\![P_0]\!]\rho_0\emptyset\emptyset true\ \varphi_{1_0}\varphi_{2_0}\emptyset, \sigma$ is a closed substitution, $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \sigma\omega_2, \Sigma \vdash H_2 \sim \sigma H, \Sigma \vdash \varphi_1^2 = \varphi_1, \Sigma \vdash \varphi_2^2 = \varphi_2$, and $nf'_{T,\Sigma}(\{\rho_2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$, then there exist $\sigma_1, \rho^1, H_1, \omega_1^1, \omega_2^1, \varphi_1^1, \varphi_2^1, \lambda'$ such that $\rho^2 = \sigma_1\rho^1, \omega_1^2 = \sigma_1\omega_1^1, \omega_2^2 = \sigma_1\omega_2^1, H_2 \sim \sigma_1 H_1, \varphi_1^2 = \sigma_1\varphi_1^1, \varphi_2^2 = \sigma_1\varphi_2^1$, and $[\![P]\!]\rho_1\omega_1^1\omega_2^1 H_1\varphi_1^1\varphi_2^1\lambda'$ is called during the generation of $[\![P_0]\!]\rho_0\emptyset\emptyset H\varphi_{1_0}\varphi_{2_0}\emptyset$.*

Proof. The process P is a subprocess of P_0. We proceed by induction on P: we show the result for P_0 itself, and we show that if the result is true for some occurrence of P, then it is also true for the occurrences of the direct subprocesses of P.

Case P_0: We have $\rho^2 = \rho_0, \omega_1^2 = \omega_2^2 = \emptyset, \varphi_1^2 = \varphi_{1_0}, \varphi_2^2 = \varphi_{2_0}$, and $H_2 = true$. Then we obtain the result by letting σ_1 be any substitution, $\rho^1 = \rho_0, \omega_1^1 = \omega_2^1 = \emptyset$, and $H_1 = true$.

Case 0: Void, since it has no subprocesses.

Case $P \mid Q$: Obvious by induction hypothesis.

Case $!P'$: Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then since the translation rule for $!P$ was applied $\rho = \rho^3, \omega_1 = i :: \omega_1^3, \omega_2 = i :: \omega_2^3, H = H_3, \varphi_1 = \varphi_1^3, \varphi_2 =$

φ_2^3, $\lambda = \lambda_3$, and $[\![!P']\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3\lambda_3$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2$, φ_2^2 such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \sigma\omega_2, H_2 \sim \sigma H, \Sigma \vdash \varphi_1^2 = \sigma\varphi_1, \Sigma \vdash \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4, \omega_1^2 = p :: \omega_1^4, \omega_2^2 = p :: \omega_2^4, H_2 = H_4, \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where $\Sigma \vdash \rho^4 = \sigma\rho^3$, $\Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3$ $\Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3$, and $\Sigma \vdash p = \sigma i$. By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma_1\varphi_2^5$ and $[\![!P']\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called. Let $\lambda' = \lambda$ since i is a fresh variable, we can define $\sigma_1 i = p$. Then $[\![P']\!]\rho^5 i :: \omega_1^5 i :: \omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called, $\rho^2 = \sigma_1\rho^5, \omega_1^2 = \sigma_1(i :: \omega_1^5), \omega_2^2 = \sigma_1(i :: \omega_2^5), \varphi_1^2 = \sigma_1\varphi_1^5, \varphi_2^2 = \sigma_1\varphi_2^5$ and $H_2 \sim \sigma_1 H_5$.

Case $\texttt{new}\ a; P'$: Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then since the translation rule for $\texttt{new}\ a; P'$ was applied $\rho = \rho^3[a \mapsto (a[\omega_1], a[\omega_2])]$ and $[\![\texttt{new}\ a; P']\!]\rho^3\omega_1\omega_2$ $H\varphi_1\varphi_2\lambda$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4[a \mapsto (a[\omega_1^2], a[\omega_2^2])]$ where $\Sigma \vdash \rho^4 = \sigma\rho^3$. By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^1, \omega_2^1, H_1, \varphi_1^1, \varphi_2^1, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^2 = \sigma_1\omega_1^1, \omega_2^2 = \sigma_1\omega_2^1, H_2 \sim \sigma_1 H_1, \varphi_1^2 = \sigma_1\varphi_1^1, \varphi_2^2 = \sigma_1\varphi_2^1$ and $[\![\texttt{new}\ a; P']\!]\rho^5\omega_1^1\omega_2^1 H1\varphi_1^1\varphi_2^1\lambda'$ has been called. Let $\lambda' = \lambda$ then $[\![P']\!](\rho^5[a \mapsto (a[\omega_1^1], a[\omega_1^2])])\omega_1^1\omega_1^2 H_1\varphi_1^1\varphi_2^1$ has been called, $\rho^2 = \sigma_1(\rho_5[a \mapsto (a[\omega_1^1], a[\omega_1^2])]), \omega_1^2 = \sigma_1\omega_1^1, \omega_2^2 = \sigma_1\omega_2^1$, and $H_2 \sim \sigma_1 H_1$.

Case $\texttt{out}(M, N); P'$: Obvious by induction hypothesis.

Case $\texttt{in}(M, x); P'$: Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then since the translation rule for $\texttt{in}(M, x); P'$ was applied we have $\rho = \rho^3[x \mapsto (x', x'')] \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda\}$, $\omega_1 = x :: \omega_1^3, \omega_2 = x :: \omega_2^3, H = H_3 \wedge msg(\rho_1^3(\varphi_1), \rho_1^3(\texttt{fst}(M)), x', \rho_2^3(\varphi_2), \rho_1^3(\texttt{snd}(M)), x''), \varphi_1 = \varphi_1^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda], \varphi_2 = \varphi_2^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda]$, and $[\![\texttt{in}(M, x); P']\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3\lambda$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$

such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4[x \mapsto (p', p''), y_k \mapsto (p'_{y_k}, p''_{y_k}) \mid 1 \leq k \leq n, k \notin \lambda_1], \omega_1^2 = p' :: \omega_1^4, \omega_2^2 = p'' :: \omega_2^4, H_2 = H_4 \wedge msg(\rho_1^4(\varphi_1), \rho_1^4(\mathtt{fst}(M)), p', \rho_2^4(\varphi_2), \rho_2^4(\mathtt{snd}(M)), p''), \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where $\Sigma \vdash \rho^4 = \sigma\rho^3, \Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3, \Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3, \Sigma \vdash p' = \sigma x$, and $\Sigma \vdash p'' = \sigma x'', \Sigma \vdash p'_{y_k} = \sigma y'_k, \Sigma \vdash p''_{y_k} = \sigma y''_k, 1 \leq k \leq n, k \notin \lambda_1$. (Since P_0 is unevaluated, M is either a variable y or $\mathtt{diff}[a, a]$ for some name a. Let $u = y$ in the first case and $u = a$ in the second case. We have $u \in dom(\rho^3) = dom(\rho^4)$. We have $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2\})$ so a fortiori $nf'_{T,\Sigma}(\{\rho_4, H_2\})$, and the first and third arguments of msg are equal to $\rho_1^4(\mathtt{fst}(M)) = \rho_1^4(u)$ and $\rho_2^4(\mathtt{snd}(M)) = \rho_2^4(u)$ modulo Σ respectively, so they are exactly $\rho_1^4(\mathtt{fst}(M))$ and $\rho_2^4(\mathtt{snd}(M))$). By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma\varphi_2^5$, and $[\![\mathtt{in}(M, x); P']\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called. Since x' and x'' are fresh variables, we can define $\sigma_1 x' = p'$ and $\sigma_1 x'' = p''$ and since $y'_k, y''_k 1 \leq k \leq n, k \notin \lambda_1$ are fresh variables, we can define $\sigma_1 y'_k = p'_{y_k}, \sigma_1 y''_k = p''_{y_k} 1 \leq k \leq n, k \notin \lambda_1$. Then $[\![P']\!](\rho^5[x \mapsto (x', x'')] \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda\})(x' :: \omega_1^5)(x'' :: \omega_2^5)(H_5 \wedge msg(\rho_1^5(\varphi_1), \rho_1^5(\mathtt{fst}(M)), x',$ $\rho_2^5(\varphi_2), \rho_1^5(\mathtt{snd}(M)), x''), \varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\lambda'$ has been called, and $\rho^2 = \sigma_1(\rho^5[x \mapsto (x', x'')] \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda'\}), \omega_1^2 = \sigma_1(x' :: \omega_1^5), \omega_2^2 = \sigma_1(x'' :: \omega_2^5), \varphi_1^2 = \sigma_1\varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda'], \varphi_2^2 = \sigma_1\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']$, and $H_2 \sim \sigma_1(H_5 \wedge msg(\rho_1^5(\varphi_1), \rho_1^5(\mathtt{fst}(M)), x', \rho_2^5(\varphi_2), \rho_1^5(\mathtt{snd}(M)), x''))$.

Case $\mathtt{let}\ x = D\ \mathtt{in}\ P'\ \mathtt{else}\ Q$: we have two cases

- Subprocess P': Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then we have $\rho = (\sigma_1\rho^3)[x \mapsto (p'_1, p''_1)], \omega_1 = p'_1 :: \sigma_1\omega_1^3, \omega_2 = p''_1 :: \sigma_1\omega_2^3, \varphi_1 = \sigma_1\varphi_1^3, \varphi_2 = \sigma_1\varphi_2^3$ and $H = \sigma_1 H_3$ where $[\![\mathtt{let}\ x = D\ \mathtt{in}\ P'\ \mathtt{else}\ Q]\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3$

has been called and $(\rho_1(D), \rho_2(D)) \Downarrow' ((p'_1, p''_1), \sigma_1)$. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4[x \mapsto (p'_4, p''_4)], \omega_1^2 = p'_4 :: \omega_1^4, \omega_2^2 = p''_4 :: \omega_2^4, H_2 = H_4, \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ with $\Sigma \vdash \rho^4 = \sigma\sigma_1\rho^3, \Sigma \vdash \omega_1^4 = \sigma\sigma_1\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\sigma_1\omega_2^3, \Sigma \vdash H_4 \sim \sigma\sigma_1 H_3, \Sigma \vdash p'_4 = \sigma p'_1, \Sigma \vdash p''_4 = \sigma p''_1, \Sigma \vdash \varphi_1^4 = \sigma\sigma_1\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\sigma_1\varphi_2^3$, and $nf'_{T,\Sigma}(\{\rho 4, \omega_1^4, \omega_2^4, H_4, p'_4, p''_4\})$. By induction hypothesis, there exist $\sigma_0, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_0\rho 5, \omega_1^4 = \sigma_0\omega_1^5, \omega_2^4 = \sigma_0\omega_2^5, H_4 \sim \sigma_0 H_5, \varphi_1^4 = \sigma_0\varphi_1^5, \varphi_2^4 = \sigma_0\varphi_2^5$, and $[\![\texttt{let } x = D \texttt{ in } P' \texttt{ else } Q]\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called. By Lemma C.13, there exist σ_2, p'_2, p''_2, and σ_3 such that $(\rho_1^5(D), \rho_2^5(D)) \Downarrow' ((p'_2, p''_2), \sigma_2), \sigma_0 = \sigma_3\sigma_2$ except on fresh variables introduced in the computation of $(\rho_1^5(D), \rho_2^5(D)) \Downarrow' ((p'_2, p''_2), \sigma_2), p'_4 = \sigma_3 p'_2$, and $p''_4 = \sigma_3 p''_2$. Moreover, by definition of $[\![\texttt{let } x = D \texttt{ in } P' \texttt{ else } Q]\!]$, we have that $[\![P']\!]((\sigma_2\rho^5)[x \mapsto (p'_2, p''_2)])(p'_2 :: \sigma_2\omega_1^5)(p''_2 :: \sigma_2\omega_2^5)(\sigma_2 H_5)\sigma_2\varphi_1^5\sigma_2\varphi_2^5\lambda'$ has been called, so we obtain the result by letting $\rho^1 = (\sigma_2\rho^5)[x \mapsto (p'_2, p''_2)], \omega_1^1 = (p'_2 :: \sigma_2\omega_1^5), \omega_2^1 = (p''_2 :: \sigma_2\omega_2^5), H_1 = \sigma_2 H_5, \varphi_1^1 = \sigma_2\varphi_1^5, \varphi_2^1 = \sigma_2\varphi_2^5$: we have $\rho^2 = \rho^4[x \mapsto (p'_4, p''_4)] = (\sigma_0\rho^5)[x \mapsto (\sigma_3 p'_2, \sigma_3 p''_2)] = \sigma_3((\sigma_2\rho^5)[x \mapsto (p'_2, p''_2)]) = \sigma_3\rho_1$, and similarly for $\omega_1^2, \omega_2^2, \varphi_1^2, \varphi_2^2$, and H_2.

– Subprocess Q: Assume $[\![Q]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then $H = H_3 \wedge \rho_1(fails(\texttt{fst}(D))) \wedge \rho_2(fails(\texttt{snd}(D)))$ and $[\![\texttt{let } x = D \texttt{ in } P' \texttt{ else } Q]\!]$ $\rho\omega_1\omega_2 H_3\varphi_1\varphi_2\lambda$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \sigma\omega_2, \Sigma \vdash H_2 \sim \sigma H, \Sigma \vdash \varphi_1^2 = \sigma\varphi_1, \Sigma \vdash \varphi_2^2 = \sigma\varphi_2$, and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $H_2 = H_4 \wedge H_{4_{nounif}}$ where $H_{4_{nounif}}$ consists of $nounif$ facts, $\Sigma \vdash H_{4_{nounif}} \sim \sigma\rho_1(fails(\texttt{fst}(D))) \wedge \sigma\rho_2(fails(\texttt{snd}(D)))$, and $\Sigma \vdash H_4 \sim \sigma H_3$. By induction hypothesis, there exist $\sigma_1, \rho_1, \omega_1^1, \omega_2^1, H_5, \varphi_1^1, \varphi_2^1, \lambda'$ such that $\rho^2 = \sigma_1\rho^1, \omega_1^2 = \sigma_1\omega_1^1, \omega_2^2 = \sigma_1\omega_2^1, H_4 \sim \sigma_1 H_5, \varphi_1^2 = \sigma_1\varphi_1^1, \varphi_2^2 = \sigma_1\varphi_2^1$,

and $[\![\texttt{let}\ x = D\ \texttt{in}\ P'\ \texttt{else}\ Q]\!]\rho_1\omega_1^1\omega_2^1 H_5\varphi_1^1\varphi_2^1\lambda'$ has been called. Then $[\![Q]\!]\rho_1\omega_1^1\omega_2^1(H_5 \wedge \rho_1(fails(\texttt{fst}(D))) \wedge \rho_2(fails(\texttt{snd}(D)))$ has been called, which yields the desired result, knowing that $H_2 = H_4 \wedge H_{4_{nounif}} \sim \sigma_1 H_5 \wedge \sigma_1\rho_1^1(fails(\texttt{fst}(D))) \wedge \sigma_1\rho_2^1(fails(\texttt{snd}(D)))$, since $\Sigma \vdash \sigma_1\rho^1 = \rho^2 = \sigma\rho$.

Case $\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P'$: Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then since the translation rule for $\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P'$ was applied we have $\rho = \rho^3 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda_1\}$, $\omega_1 = \omega_1^3, \omega_2 = \omega_2^3, H = H_3, \varphi_1 = \varphi_1^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1], \varphi_2 = \varphi_2^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1], \lambda = \lambda_1 \cup \{i_1, \ldots, i_m\}$, and $[\![\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P']\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3\lambda_1$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4[y_k \mapsto (p'_{y_k}, p''_{y_k}) \mid 1 \leq k \leq n, k \notin \lambda_1], \omega_1^2 = \omega_1^4, \omega_2^2 = \omega_2^4, H_2 = H_4, \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where $\Sigma \vdash \rho^4 = \sigma\rho^3, \Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3, \Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3, \Sigma \vdash p'_{y_k} = \sigma y'_k, \Sigma \vdash p''_{y_k} = \sigma y''_k, 1 \leq k \leq n, k \notin \lambda_1$. By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma\varphi_2^5$, and $[\![\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P']\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called. Since $y'_k, y''_k 1 \leq k \leq n, k \notin \lambda_1$ are fresh variables, we can define $\sigma_1 y'_k = p'_{y_k}, \sigma_1 y''_k = p''_{y_k} 1 \leq k \leq n, k \notin \lambda_1$. Then $[\![P']\!](\rho^5 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda'\})(x' :: \omega_1^5)(x'' :: \omega_2^5) H_5\varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\lambda' \cup \{i_1, \ldots, i_m\}$ has been called, and $\rho^2 = \sigma_1(\rho^5 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda\}), \omega_1^2 = \sigma_1\omega_1^5, \omega_2^2 = \sigma_1\omega_2^5, \varphi_1^2 = \sigma_1\varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda'], \varphi_2^2 = \sigma_1\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']$, and $H_2 \sim \sigma_1 H_5$.

Case $\texttt{unlock}\ s_{i_1}, \ldots, s_{i_m}; P'$: Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called. Then since the translation rule for $\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P'$ was applied we have $\rho = \rho^3 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda_1\}$, $\omega_1 = \omega_1^3, \omega_2 = \omega_2^3, H = H_3, \varphi_1 = \varphi_1^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1], \varphi_2 = \varphi_2^3[k \mapsto y_k \mid 1 \leq k \leq$

$n, k \notin \lambda_1], \lambda = \lambda_1 \cap \{i_1, \ldots, i_m\}$, and $[\![\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P']\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3\lambda_1$ has been called. Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that $\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$. Then $\rho^2 = \rho^4[y_k \mapsto (p'_{y_k}, p''_{y_k}) \mid 1 \leq k \leq n, k \notin \lambda_1], \omega_1^2 = \omega_1^4, \omega_2^2 = \omega_2^4, H_2 = H_4, \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where $\Sigma \vdash \rho^4 = \sigma\rho^3, \Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3, \Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3, \Sigma \vdash p'_{y_k} = \sigma y'_k, \Sigma \vdash p''_{y_k} = \sigma y''_k, 1 \leq k \leq n, k \notin \lambda_1$. By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma\varphi_2^5$, and $[\![\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; P']\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called. Since $y'_k, y''_k 1 \leq k \leq n, k \notin \lambda_1$ are fresh variables, we can define $\sigma_1 y'_k = p'_{y_k}, \sigma_1 y''_k = p''_{y_k} 1 \leq k \leq n, k \notin \lambda_1$. Then $[\![P']\!](\rho^5 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda'\})(x' :: \omega_1^5)(x'' :: \omega_2^5)H_5\varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']\lambda' \cap \{i_1, \ldots, i_m\}$ has been called, and $\rho^2 = \sigma_1(\rho^5 \cup \{y_k \mapsto (y'_k, y''_k) \mid y_k, y'_k, y''_k$ are fresh, $1 \leq k \leq n, k \notin \lambda'\}), \omega_1^2 = \sigma_1\omega_1^5, \omega_2^2 = \sigma_1\omega_2^5, \varphi_1^2 = \sigma_1\varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda'], \varphi_2^2 = \sigma_1\varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']$, and $H_2 \sim \sigma_1 H_5$.

Case $\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; P'$:

Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called.

Let y_k, y'_k, y''_k where $1 \leq k \leq n, k \notin \lambda_1, \lambda_1 = \lambda$ and $vc, vc', vc'', vm, vm', vm''$ be fresh variables, since the translation rule for $\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; P'$ was applied we have:

$\rho = \rho^3[x_j \mapsto (\rho_1^3(\varphi_1[i_j]), \rho_2^3(\varphi_2[i_j])) \mid 1 \leq j \leq m] \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\}$,

$\omega_1 = \rho_1^3(\varphi_1[i_1]) :: \cdots :: \rho_1^3(\varphi_1[i_m]) :: \omega_1^3, \omega_2 = \rho_2^3(\varphi_2[i_1]) :: \cdots :: \rho_2^3(\varphi_2[i_m]) :: \omega_2^3$,

$H = H_3 \wedge msg(\rho_1^3(\varphi_1), vc', vm', \rho_2^3(\varphi_2), vc'', vm'')$,

$\varphi_1 = \varphi_1^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1], \varphi_2 = \varphi_2^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1]$, and

$[\![\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; P']\!]\rho^3\omega_1^3\omega_2^3 H_3\varphi_1^3\varphi_2^3\lambda_1$ has been called.

Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that
$\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$ and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$.
Then $\rho^2 = \rho^4[vc \mapsto (p'_{vc}, p''_{vc}), vm \mapsto (p'_{vm}, p''_{vm}), x_j \mapsto (\rho_1^4(\varphi_1^4[i_j]), \rho_2^4(\varphi_2^4[i_j])),$
$y_k \mapsto (p'_{y_k}, p''_{y_k}) \mid 1 \leq j \leq m, 1 \leq k \leq n, k \notin \lambda_1]$,
$\omega_1^2 = \rho_1^4(\varphi_1^4[i_1]) :: \cdots :: \rho_1^4(\varphi_1^4[i_m]) :: \omega_1^4, \omega_2^2 = \rho_2^4(\varphi_2^4[i_1]) :: \cdots :: \rho_2^4(\varphi_2^4[i_m]) :: \omega_2^4$,
$H_2 = H_4 \wedge msg(\rho_1^4(\varphi_1^4), p'_{vc}, p'_{vm}, \rho_2^4(\varphi_2^4), p''_{vc}, p''_{vm}), \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where
$\Sigma \vdash \rho^4 = \sigma\rho^3, \Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3, \Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3, \Sigma \vdash p'_{vc} = \sigma vc', \Sigma \vdash p''_{vc} = \sigma vc'', \Sigma \vdash p'_{vm} = \sigma vm', \Sigma \vdash p''_{vm} = \sigma vm'', \Sigma \vdash p'_{y_k} = \sigma y'_k, \Sigma \vdash p''_{y_k} = \sigma y''_k, 1 \leq k \leq n, k \notin \lambda_1$.
By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma\varphi_2^5$, and $[\![\texttt{read } s_{i_1}, \ldots, s_{i_m} \texttt{ as } x_1, \ldots, x_m; P']\!]\rho^5\omega_1^5\omega_2^5 H_5\varphi_1^5\varphi_2^5\lambda'$ has been called.
Let $\lambda' = \lambda_1$, since $y'_k, y''_k 1 \leq k \leq n, k \notin \lambda_1$ are fresh variables and vc', vc'', vm', vm'' are fresh variables, we can define $\sigma_1 y'_k = p'_{y_k}, \sigma_1 y''_k = p''_{y_k} 1 \leq k \leq n, k \notin \lambda_1, \sigma_1 vc' = p'_{vc}, \sigma_1 vc'' = p''_{vc}, \sigma_1 vm' = p'_{vm}, \sigma_1 vm'' = p''_{vm}$.
Let $\varphi_1^6 = \varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda'], \varphi_2^6 = \varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']$, then
$[\![P']\!](\rho^5[x_j \mapsto (\rho_1^5(\varphi_1^6[i_j]), \rho_2^5(\varphi_2^6[i_j])) \mid 1 \leq j \leq m] \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\})\ (\rho_1^5(\varphi_1^6[i_1]) :: \cdots :: \rho_1^5(\varphi_1^6[i_m]) :: \omega_1^5)(\rho_2^5(\varphi_2^6[i_1]) :: \cdots :: \rho_2^5(\varphi_2^6[i_m]) :: \omega_2^5)\ (H_5 \wedge msg(\rho_1^5(\varphi_1^6), vc', vm', \rho_2^5(\varphi_2^6), vc'', vm''))\varphi_1^6\varphi_2^6\lambda'$ has been called, and
$\rho^2 = \sigma_1(\rho^5[x_j \mapsto (\rho_1^5(\varphi_1^6[i_j]), \rho_2^5(\varphi_2^6[i_j])) \mid 1 \leq j \leq m] \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\}), \omega_1^2 = \sigma_1(\rho_1^5(\varphi_1^6[i_1]) :: \cdots :: \rho_1^5(\varphi_1^6[i_m]) :: \omega_1^5), \omega_2^2 = \sigma_1(\rho_2^5(\varphi_2^6[i_1]) :: \cdots :: \rho_2^5(\varphi_2^6[i_m]) :: \omega_2^5), \varphi_1^2 = \sigma_1\varphi_1^5, \varphi_2^2 = \sigma_1\varphi_2^5$, and $H_2 \sim \sigma_1(H_5 \wedge msg(\rho_1^5(\varphi_1^6), vc', vm', \rho_2^5(\varphi_2^6), vc'', vm''))$.

Case $s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; P'$:

Assume $[\![P']\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ is called.

Let y_k, y'_k, y''_k where $1 \leq k \leq n, k \notin \lambda_1, \lambda_1 = \lambda$ and $vc, vc', vc'', vm, vm', vm''$ be fresh variables, since the translation rule for $s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; P'$ was applied we have:

$\rho = \rho^3 \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\}$,

$\omega_1 = \omega_1^3, \omega_2 = \omega_2^3$, $H = H_3$

$\varphi_1 = \varphi_1^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1], \varphi_2 = \varphi_2^3[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda_1]$,

and $[\![s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; P']\!]\rho^3\omega_1^3\omega_2^3 H_3 \varphi_1^3 \varphi_2^3 \lambda_1$ has been called.

Let $\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2$ such that

$\Sigma \vdash \rho^2 = \sigma\rho, \Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \omega_2, \Sigma \vdash H_2 \sim \sigma H, \varphi_1^2 = \sigma\varphi_1, \varphi_2^2 = \sigma\varphi_2$

and $nf'_{T,\Sigma}(\{\rho^2, \omega_1^2, \omega_2^2, H_2, \varphi_1^2, \varphi_2^2\})$.

Then $\rho^2 = \rho^4[vc \mapsto (p'_{vc}, p''_{vc}), vm \mapsto (p'_{vm}, p''_{vm}), y_k \mapsto (p'_{y_k}, p''_{y_k}) \mid 1 \leq k \leq n, k \notin \lambda_1]$, $\omega_1^2 = \omega_1^4, \omega_2^2 = \omega_2^4$, $H_2 = H_4, \varphi_1^2 = \varphi_1^4, \varphi_2^2 = \varphi_2^4$ where

$\Sigma \vdash \rho^4 = \sigma\rho^3, \Sigma \vdash \omega_1^4 = \sigma\omega_1^3, \Sigma \vdash \omega_2^4 = \sigma\omega_2^3, \Sigma \vdash H_4 \sim \sigma H_3, \Sigma \vdash \varphi_1^4 = \sigma\varphi_1^3, \Sigma \vdash \varphi_2^4 = \sigma\varphi_2^3, \Sigma \vdash p'_{vc} = \sigma vc', \Sigma \vdash p''_{vc} = \sigma vc'', \Sigma \vdash p'_{vm} = \sigma vm', \Sigma \vdash p''_{vm} = \sigma vm'', \Sigma \vdash p'_{y_k} = \sigma y'_k, \Sigma \vdash p''_{y_k} = \sigma y''_k, 1 \leq k \leq n, k \notin \lambda_1$.

By induction hypothesis, there exist $\sigma_1, \rho^5, \omega_1^5, \omega_2^5, H_5, \varphi_1^5, \varphi_2^5, \lambda'$ such that $\rho^4 = \sigma_1\rho^5, \omega_1^4 = \sigma_1\omega_1^5, \omega_2^4 = \sigma_1\omega_2^5, H_4 \sim \sigma_1 H_5, \varphi_1^4 = \sigma_1\varphi_1^5, \varphi_2^4 = \sigma\varphi_2^5$, and $[\![s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; P']\!]\rho^5\omega_1^5\omega_2^5 H_5 \varphi_1^5 \varphi_2^5 \lambda'$ has been called.

Let $\lambda' = \lambda_1$, since $y'_k, y''_k 1 \leq k \leq n, k \notin \lambda_1$ are fresh variables and vc', vc'', vm', vm'' are fresh variables, we can define $\sigma_1 y'_k = p'_{y_k}, \sigma_1 y''_k = p''_{y_k} 1 \leq k \leq n, k \notin \lambda_1, \sigma_1 vc' = p'_{vc}, \sigma_1 vc'' = p''_{vc}, \sigma_1 vm' = p'_{vm}, \sigma_1 vm'' = p''_{vm}$.

Let $\varphi_1^6 = \varphi_1^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda'], \varphi_2^6 = \varphi_2^5[k \mapsto y_k \mid 1 \leq k \leq n, k \notin \lambda']$, then

$[\![P']\!](\rho^5 \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\})\, \omega_1^5\omega_2^5 H_5 \varphi_1^6 \varphi_2^6 \lambda'$ has been called, and

$\rho^2 = \sigma_1(\rho^5 \cup \{y_k \mapsto (y'_k, y''_k)\} \cup \{vc \mapsto (vc', vc'')\} \cup \{vm \mapsto (vm', vm'')\})$, $\omega_1^2 = \sigma_1\omega_1^5, \omega_2^2 = \sigma_1\omega_2^5$, $\varphi_1^2 = \sigma_1\varphi_1^5, \varphi_2^2 = \sigma_1\varphi_2^5$, and $H_2 \sim \sigma_1 H_5$.

□

Lemma C.15. *Let $A_0 = (\mathcal{E}0, \mathcal{S}0, \{(P_0, \emptyset)\})$ be a closed, unevaluated stateful biprocess. For all clauses $H \to C \in \mathcal{C}_{A_0}$, for all closed substitutions σ, for all $H_2 \to C_2$ such that $\Sigma \vdash H_2 \to C_2 \sim \sigma(H \to C)$ and $nf'_{T,\Sigma}(\{H_2, C_2\})$, there exist a closed substitution σ_1 and a clause $H_1 \to C_1 \in \mathcal{C}_{A_0}$ such that $H_2 \sim \sigma_1 H_1$ and $C_2 = \sigma_1 C_1$.*

Proof. The clauses of $[\![P_0]\!]\rho 0 \emptyset \emptyset true \varphi 0_1 \varphi 0_2 \emptyset$ are generated from the following cases:

- $H \to C = H \to input(\rho_1(\varphi_1[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_1(\mathtt{fst}(M)), \rho_2(\varphi_2[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_2(\mathtt{snd}(M)))$ where $[\![\mathtt{in}(M, x); P]\!]\rho\omega_1, \omega_2 H \varphi_1\varphi_2$ has been called during the generation of $[\![P_0]\!]\rho 0\emptyset\emptyset true\varphi 0_1\varphi 0_2\emptyset$. Since $\Sigma \vdash H_2 \to C_2 \sim \sigma(H \to C)$ and $nf'_{T,\Sigma}(\{H_2, C_2\})$, we have $\Sigma \vdash H_2 \sim \sigma H, C_2 = input(\rho_1(\varphi'_1), p'_2, \rho_2(\varphi'_2), p''_2), \Sigma \vdash p'_2 = \sigma\rho_1(\mathtt{fst}(M))$, and $\Sigma \vdash p''_2 = \sigma\rho_2(\mathtt{snd}(M))$. Since P_0 is unevaluated, M is a variable y or $\mathtt{diff}[a, a]$ for some name a. Let $u = y$ in the first case and $u = a$ in the second case. We have $u \in dom(\rho)$. We define ρ^2 by $\rho^2(u) = \mathtt{diff}[p'_2, p''_2]$ and extend ρ^2 to $dom(\rho)$ in such a way that $\Sigma \vdash \rho^2 = \sigma\rho$ and $nf'_{T,\Sigma}(\{H_2, \rho^2\})$ by Property S2. We also define $\omega_1^2, \omega_2^2, \varphi_1^2, \varphi_2^2$ so that $\Sigma \vdash \omega_1^2 = \sigma\omega_1, \Sigma \vdash \omega_2^2 = \sigma\omega_2, \Sigma \vdash \varphi_1^2 = \sigma\varphi_1, \Sigma \vdash \varphi_2^2 = \sigma\varphi_2$, and $nf'_{T,\Sigma}(\{H_2, \rho^2, \omega_1^2, \omega_2^2, \varphi_1^2, \varphi_2^2\})$ by Property S2. By Lemma C.14, there exist $\sigma_1, \rho^1, \omega_1^1, \omega_2^1, H_1, \varphi_1^1, \varphi_2^1, \lambda'$ such that $\rho^2 = \sigma_1\rho^1, \omega_1^2 = \sigma_1\omega_1^1, \omega_2^2 = \sigma_1\omega_2^1, H_2 \sim \sigma_1 H_1, \varphi_1^2 = \sigma_1\varphi_1^1, \varphi_2^2 = \sigma_1\varphi_2^1$, and $[\![\mathtt{in}(M, x); P]\!]\rho^1\omega_1^1\omega_2^1 H_1 \varphi_1^1\varphi_2^1\lambda'$ has been called. Then $H_1 \to input(\rho_1^1(\varphi_1^1[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_1^1(\mathtt{fst}(M)), \rho_2^1(\varphi_2^1[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_2^1(\mathtt{snd}(M)))$ is in $[\![P_0]\!]\rho\emptyset\emptyset true\varphi 0_1\varphi 0_2\emptyset, H_2 \sim \sigma_1 H_1, C_2 = input(\rho_1^2(\varphi_1^2[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), p'_2, \rho_2^2(\varphi_2^2[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), p''_2) = input(\rho_1^2(\varphi_1^2[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_1^2(\mathtt{fst}(M)), \rho_2^2(\varphi_2^2[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_2^2(\mathtt{snd}(M))) = \sigma_1 input(\rho_1^1(\varphi_1^1[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda']), \rho_1^1(\mathtt{fst}(M)),$
  $\rho_2^1(\varphi_2^1[k \mapsto y_k \mid 1 \le k \le n, k \notin \lambda'])\rho_2^1(\mathtt{snd}(M)))$.
- $H \to C = H \to msg(\rho_1(\varphi_1), \rho_1(\mathtt{fst}(M), \rho_1(\mathtt{fst}(N), \rho_2(\varphi_2), \rho_2(\mathtt{snd}(M)),$
  $\rho_2(\mathtt{snd}(N)))$ where $[\![\mathtt{out}(M, N); P]\!]\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$ has been called. This case

is similar to the previous one. (The terms M and N are variables or $\mathtt{diff}[a, a]$ for some name a.)

- $H \to C = \sigma' H' \wedge \sigma' \rho_2(fails(\mathtt{snd}(D))) \to bad$ where
  $[\![\mathtt{let}\ x = D\ \mathtt{then}\ P\ \mathtt{else}\ Q]\!]\rho\omega_1\omega_2 H' \varphi_1\varphi_2\lambda$ has been called and $\rho_1(D) \Downarrow' (p', \sigma')$. Since $\Sigma \vdash H_2 \to C_2 \sim \sigma(H \to C)$ and $nf'_{T,\Sigma}(\{H_2, C_2\})$, we have $H_2 = H_3 \wedge H_{3_{nounif}}$ where $\Sigma \vdash H_3 \sim \sigma\sigma' H'$ and $H_{3_{nounif}}$ consists of nounif facts such that $\Sigma \vdash H_{3_{nounif}} \sim \sigma\sigma'\rho_2(fails(\mathtt{snd}(D)))$. By Property S2, there exist $\rho^3, \omega_1^3, \omega_2^3, \varphi_1^3, \varphi_2^3$ such that $\Sigma \vdash \rho^3 = \sigma\sigma'\rho, \Sigma \vdash \omega_1^3 = \sigma\sigma'\omega_1, \Sigma \vdash \omega_2^3 = \sigma\sigma'\omega_2, \Sigma \vdash \varphi_1^3 = \sigma\sigma'\varphi_1, \Sigma \vdash \varphi_2^3 = \sigma\sigma'\varphi_2$, and $nf'_{T,\Sigma}(\{\rho^3, \omega_1^3, \omega_2^3, H_3, \varphi_1^3, \varphi_2^3\})$. By Lemma C.14, there exist $\sigma_1, \rho^1, \omega_1^1, \omega_2^1, H_1, \varphi_1^1, \varphi_2^1, \lambda'$ such that $\rho^3 = \sigma_1\rho_1, \omega_1^3 = \sigma_1\omega_1^1, \omega_2^3 = \sigma_1\omega_2^1, H_3 \sim \sigma_1 H_1, \varphi_1^3 = \sigma_1\varphi_1^1, \varphi_2^3 = \sigma_1\varphi_2^1$, and
  $[\![\mathtt{let}\ x = D\ \mathtt{in}\ P\ \mathtt{else}\ Q]\!]\rho^1\omega_1^1\omega_2^1 H_1\varphi_1^1\varphi_2^1\lambda'$ has been called. By Property S2, we can choose p such that $\Sigma \vdash p = \sigma p'$ and $nf_{T,\Sigma}(\{p, \sigma_1\rho^1\})$. By Lemma C.13, there exist σ'_1, p'_1, and σ''_1 such that $\rho_1^1(D) \Downarrow' (p'_1, \sigma'_1)$ and $\sigma_1 = \sigma''_1\sigma'_1$ except on fresh variables introduced in the computation of $\rho_1^1(D) \Downarrow' (p'_1, \sigma'_1)$. Then $\sigma'_1 H_1 \wedge \sigma'_1\rho_2^1(fails(\mathtt{snd}(D))) \to bad$ is in $[\![P_0]\!]\rho 0\emptyset\emptyset true\varphi 0_1\varphi 0_2\emptyset$. Moreover $\sigma''_1(\sigma'_1 H_1 \wedge \sigma'_1\rho_2^1(fails(\mathtt{snd}(D)))) = \sigma_1 H_1 \wedge \sigma_1\rho_2^1(fails(\mathtt{snd}(D))) \sim H_3 \wedge H_{3_{nounif}} \sim H_2$, since $\Sigma \vdash \sigma_1\rho^1 = \rho^3 = \sigma\sigma'\rho$, and $\sigma''_1 bad = bad = C$, so we have the desired result.

- $H \to C = \sigma' H' \wedge \sigma'\rho_1(fails(\mathtt{fst}(D))) \to bad$ where
  $[\![\mathtt{let}\ x = D\ \mathtt{in}\ P\ \mathtt{else}\ Q]\!]\rho\omega_1\omega_2 H'\varphi_1\varphi_2\lambda$ has been called and $\rho_2(D) \Downarrow' (p', \sigma')$. This case is symmetric from the previous one.

For the other clauses:

- For Clause (RINIT), $C_2 = C, H_2 = \emptyset$, so we have the result by taking $H_1 \to C_1 = H \to C$.

- For Clauses (RN), (RL), (RS), (RI), (RCOM), (RCOM'), (RM1), (RM2), and (RM3), $H_2 = \sigma' H$ and $C_2 = \sigma' C$ where for all $x \in fv(H \to C), \Sigma \vdash$

$\sigma' x = \sigma x$, and $nf_{T,\Sigma}(\{\sigma' x \mid x \in fv(H \to C)\})$. (Indeed, the function symbols in H, C do not appear in equations of Σ.) So we obtain the result by taking $H_1 \to C_1 = H \to C$ and $\sigma_1 = \sigma'$.

- For Clause (RF), $H = att(xs, M_1, xs', N_1) \wedge \cdots \wedge att(xs, M_n, xs', N_n), C = att(xs, M, xs', N), h(M_1, \ldots, M_n) \to M \in def_{\Sigma'}(h), h(N_1, \ldots, N_n) \to N \in def_{\Sigma'}(h), H_2 = att(xs_1, M_1'', xs_1', N_1'') \wedge \cdots \wedge att(xs_1, M_n'', xs_1', N_n'')$,
  $C_2 = att(xs_1, M'', xs_1', N'')$ with $\Sigma \vdash M'' = \sigma M, \Sigma \vdash N'' = \sigma N, \Sigma \vdash M_i'' = \sigma M_i, \Sigma \vdash N_i'' = \sigma N_i$ for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash xs_1 = \sigma' xs, \Sigma \vdash xs_1' = \sigma' xs'$ and $nf_{T,\Sigma}(\{M'', N'', M_1'', \ldots, M_n'', N_1'', \ldots, N_n'', xs_1, xs_1'\})$. By Lemma C.11, there exist σ_1 and $h(M_1', \ldots, M_n') \to M' \in def_{\Sigma'}(h)$ such that $M'' = \sigma_1 M'$ and for all $i \in \{1, \ldots, n\}, M_i'' = \sigma_1 M_i'$. By Lemma C.11 again, there exist σ_1 and $h(N_1', \ldots, N_n') \to N \in def_{\Sigma'}(h)$ such that $N'' = \sigma_1 N'$ and for all $i \in \{1, \ldots, n\}, N_i'' = \sigma_1 N_i$. (We can use the same substitution σ_1 since the first and second arguments of the predicate *att* do not share variables.) Hence $\sigma_1 att(xs_1, M_i', xs_1', N_i') = att(xs_1, M_i'', xs_1', N_i'')$ for all $i \in \{1, \ldots, n\}$ and $\sigma_1 att(xs_1, M', xs_1', N') = att(xs_1, M'', xs_1', N'')$. We take $H_1 \to C_1 = att(xs_1, M_1', xs_1', N_1') \wedge \cdots \wedge att(xs_1, M_n', xs_1', N_n') \to att(xs_1, M', xs_1', N')$, which yields the desired result.

- For Clause (RT), we have $C_2 = C = bad, H = H_{nounif} \wedge att(xs, M_1, xs', x_1) \wedge \cdots \wedge att(xs, M_n, xs', x_n), H_2 = H_{2_{nounif}} \wedge att(xs_1, M_1'', xs_1', x_1'') \wedge \cdots \wedge att(xs_1, M_n'', xs_1', x_n'')$ where H_{nounif} and $H_{2_{nounif}}$ consist of *nounif* facts, $\Sigma \vdash H_{2_{nounif}} = \sigma H_{nounif}, g(M_1, \ldots, M_n) \to M \in def_{\Sigma'}(g), \Sigma \vdash M_i'' = \sigma M_i$ and $\Sigma \vdash N_i'' = \sigma x_i, \Sigma \vdash xs_1 = \sigma xs, \Sigma \vdash xs_1' = \sigma xs'$ for all $i \in \{1, \ldots, n\}$, and $nf_{T,\Sigma}(\{M_1, \ldots, M_n, N_1, \ldots, N_n\})$. By Lemma C.11, there exist σ_1 and $g(M_1', \ldots, M_n') \to M' \in def_{\Sigma'}(g)$ such that $M'' = \sigma_1 M'$ and for all $i \in \{1, \ldots, n\}, M_i'' = \sigma_1 M_i'$. We extend σ_1 by defining for all $i \in \{1, \ldots, n\}, \sigma_1 x_i = N_i''$ and $\sigma_1 xs = xs_1, \sigma_1 xs' = xs_1'$.
  Hence $\sigma_1 att(xs, M_i', xs', x_i) = att(xs_1, M_i'', xs_1', N_i'')$ for all $i \in \{1, \ldots, n\}$ and $\Sigma \vdash H_{2_{nounif}} = \sigma H_{nounif} = \sigma_1 H_{nounif}$ since for all $i \in \{1, \ldots, n\}, \Sigma \vdash$

$\sigma_1 x_i = N_i'' = \sigma x_i$ and $fv(H_{nounif}) = \{x_1, \ldots, x_n\}$. We take $H_1 \to C_1 = H_{nounif} \wedge att(xs, M_1', xs', x_1) \wedge \cdots \wedge att(xs, M_n, xs', x_n) \to bad$ which yields the result.

- The case of Clause (Rt') is symmetric.

□

The following lemmas can be proved by induction and thanks to what established by Lemma C.15. The full proof can be found in [BAF05].

Lemma C.16. *Assume A_0 is a closed, unevaluated stateful biprocess. If F is derivable from $\mathcal{C}_{A_0}, \Sigma \vdash F'' \sim F$, and $nf'_{T,\Sigma}(\mathcal{F} \cup \{F''\})$, then F'' is derivable from $\mathcal{C}_{A_0}$ by a derivation $\mathcal{D}$ such that $nf'_{T,\Sigma}(\mathcal{F} \cup \{\mathcal{D}\})$.*

Lemma C.17. *If bad is derivable from $\mathcal{C}_{A_0}$ then bad is derivable from $\mathcal{C}_{A_0}$ by a derivation such that $nf_{S,\Sigma}(\mathcal{F})$ where $\mathcal{F}$ is the set of intermediately derived facts in this derivation, excluding nounif facts.*

Lemma C.17 is a particular case of Lemma C.16 taking $F = F'' = bad$.

Lemma C.18. *If $\sigma fails(D)$ is false then there exists a pattern p such that $\sigma D \Downarrow_\Sigma p$.*

This lemma can be established thanks to Lemma C.7 and Lemma C.12. The full proof can be found in [BAF05].

C.7 Proof of the Properties of the Type System

Lemma C.19. *Let M be a term, S be a state (a function from $\{s_1, \ldots, s_n\}$ to terms), and E an environment (a function from names and variables to pairs of patterns). If*

i) $\forall\, a \in fn(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), y) \in \mathcal{F}_{C'A_0}$*; and*

ii) $\forall\, x \in fv(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), y') \in \mathcal{F}_{C'A_0}$

then $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{S}}_2), y'') \in \mathcal{F}_{C'A_0}$.
Symmetrically, if

i) $\forall\, a \in fn(\mathtt{snd}(M)), att(E_1(\bar{\mathcal{S}}_1), y, E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$*; and*

ii) $\forall\, x \in fv(\mathtt{snd}(M)), att(E_1(\bar{\mathcal{S}}_1), y', E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$

then $att(E_1(\bar{\mathcal{S}}_1), y'', E_2(\bar{\mathcal{S}}_2), E_2(\mathtt{snd}(M))y'') \in \mathcal{F}_{C'A_0}$.

Proof. Let M be a term, we prove that if

i) $\forall\, a \in fn(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), y) \in \mathcal{F}_{C'A_0}$; and

ii) $\forall\, x \in fv(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), y') \in \mathcal{F}_{C'A_0}$

then $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{S}}_2), y'') \in \mathcal{F}_{C'A_0}$, by induction on the depth d of M. The second part of Lemma C.19 holds by symmetry.

d=1) In this case M is either a name or a variable and by definition either $M \in fn(M)$ or $M \in fv(M)$ since $\mathtt{fst}(M) = M$ by hypothesis
$att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{S}}_2), y) \in \mathcal{F}_{C'A_0}$

$d > 1$) We have 2 cases:

- either $M = f(M_1, \ldots, M_n)$ for some constructor f of arity n. In this case, let $i \in \{1, \ldots, n\}$ by definition we have that $fn(M_i) \subseteq fn(M)$ and $fn(\mathtt{fst}(M_i)) \subseteq fn(\mathtt{fst}(M))$, $fv(M_i) \subseteq fv(M)$ and $fv(\mathtt{fst}(M_i)) \subseteq fv(\mathtt{fst}(M))$ and by hypothesis $\forall\, a \in fn(\mathtt{fst}(M_i)), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), y_i) \in \mathcal{F}_{C'A_0}$; and $\forall\, x \in fv(\mathtt{fst}(M_i)), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2),$

$y'_i) \in \mathcal{F}_{C'A_0}$ then by inductive hypothesis we have that $\forall\, i \in \{1,\ldots,n\}$ $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M_i)), E_2(\bar{\mathcal{S}}_2), y''_i) \in \mathcal{F}_{C'A_0}$, let $y'' = f(y''_1,\ldots,y''_n)$ since by definition of Σ' we have that $f(x_1,\ldots,x_n) \to f(x_1,\ldots,x_n) \in def_{\Sigma'}(f)$ for all constructors f, by clause RF we have that $att(E_1(\bar{\mathcal{S}}_1)$, $E_1(f(\mathtt{fst}(M_i),\ldots,\mathtt{fst}(M_n))), E_2(\bar{\mathcal{S}}_2), y'') \in \mathcal{F}_{C'A_0}$ and hence $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(f(M_i,\ldots,M_n))), E_2(\bar{\mathcal{S}}_2), y'') \in \mathcal{F}_{C'A_0}$

– or $M = \mathtt{diff}[M_1, M_2]$. In this case by definition $fn(M_1) \subseteq fn(M)$ and $fv(M_1) \subseteq fv(M)$ and by hypothesis $\forall\, a \in fn(M_1), att(E_1(\bar{\mathcal{S}}_1), E_1(a)$, $E_2(\bar{\mathcal{S}}_2), y_1) \in \mathcal{F}_{C'A_0}$; and $\forall\, x \in fv(M_1), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), y'_1) \in \mathcal{F}_{C'A_0}$ hence by inductive hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(M_1), E_2(\bar{\mathcal{S}}_2), y''_1) \in \mathcal{F}_{C'A_0}$

□

Lemma C.20. *Let M be a term, S be a state (a function from $\{s_1,\ldots,s_n\}$ to terms), and E an environment (a function from names and variables to pairs of patterns). If*

i) $\forall\, a \in fn(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), y) \in \mathcal{F}_{C'A_0}$*; and*

ii) $\forall\, x \in fv(\mathtt{fst}(M)), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), y') \in \mathcal{F}_{C'A_0}$

iii) $\forall\, a \in fn(\mathtt{snd}(M)), att(E_1(\bar{\mathcal{S}}_1), y'', E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$*; and*

iv) $\forall\, x \in fv(\mathtt{snd}(M)), att(E_1(\bar{\mathcal{S}}_1), y''', E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$

then $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{S}}_2), E_2(\mathtt{snd}(M))) \in \mathcal{F}_{C'A_0}$.

Proof. By Lemma C.19 and by resolution. □

C.7.1 Substitution Lemma

Lemma C.21. (*Substitution lemma). Let E be an environment (a function from names and variables to patterns), x a variable such that $x \notin dom(E)$, and M a term. Let $E' = E \cup \{x \mapsto (E_1(\mathtt{fst}(M)), E_2(\mathtt{snd}(M)))\}$.*

1. *For all $N, E(N\{^M/_x\}) = E'(N)$;*
2. *For all $\mathcal{S}$ (from states to terms), Q, λ such that $bn(Q) \cap fn(M) = \emptyset$; and $x \notin bv(Q)$, if $(E', \mathcal{S}, \lambda) \vdash Q$ then $(E, \mathcal{S}, \lambda) \vdash Q\{^M/_x\}$.*

Proof.

- We prove the first statement by induction on the depth d of N.

$d = 1$) . In this case, N is either a variable or a name. If $N \neq x$. Then we have that
$E(N\{^M/_x\}) \stackrel{def}{=} E(N) \stackrel{def}{=} E'(N)$.
If $N = x$, then we have
$E(N\{^M/_x\}) \stackrel{def}{=} E(x\{^M/_x\}) \stackrel{def}{=} E(M) \stackrel{def}{=} E'(x) \stackrel{def}{=} E'(N)$.

$d > 1$) . We have 2 cases:

- if $N = f(N_1, \ldots, N_k)$ for some constructor f of arity n and some terms $N_1, \ldots, N_k$, we have that
$E(N\{^M/_x\}) \stackrel{def}{=} E(f(N_1\{^M/_x\}, \ldots, N_k\{^M/_x\})) \stackrel{def}{=}$
$f(E(N_1\{^M/_x\}), \ldots, E(N_k\{^M/_x\})) =$ (by inductive hypothesis)
$f(E'(N_1), \ldots, E'(N_k)) \stackrel{def}{=} E'(f(N_1, \ldots, N_k)) \stackrel{def}{=} E'(N)$
- if $N = \mathtt{diff}[N_1, N_2]$ we have that
$E(N\{^M/_x\}) \stackrel{def}{=} E(\mathtt{diff}[N_1\{^M/_x\}, N_2\{^M/_x\}]) \stackrel{def}{=}$
$\mathtt{diff}[E(N_1\{^M/_x\}), E(N_2\{^M/_x\})] =$ (by inductive hypothesis)
$\mathtt{diff}[E'(N_1), E'(N_2)] \stackrel{def}{=} E'(\mathtt{diff}[N_1, N_2]) \stackrel{def}{=} E'(N)$

- We prove the second statement by induction on the depth d of Q.

$d = 0)$. In this case $Q = 0$, thus $Q\{^M/_x\} = Q = 0$, and according to our typing system

$$\frac{}{(E, \mathcal{S}, \lambda) \vdash 0}\ T_{nil}$$

$d > 0)$. We proceed by case analysis on the structure of Q.

Case $Q = Q_1 \mid Q_2$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash Q_1 \mid Q_2, bn(Q_1 \mid Q_2) \cap fn(M) = \emptyset$ and $x \notin bv(Q_1 \mid Q_2)$ then by rule $\mathcal{T}_{par}$ we have that $\lambda = \emptyset, (E', \mathcal{S}, \lambda) \vdash Q_1, (E', \mathcal{S}, \lambda) \vdash Q_2, bn(Q_1) \cap fn(M) = \emptyset \wedge bn(Q_2) \cap fn(M) = \emptyset, x \notin bv(Q_1) \wedge x \notin bv(Q_2)$ thus by inductive hypothesis we have that $\lambda = \emptyset(E, \mathcal{S}, \lambda) \vdash Q_1\{^M/_x\}, (E, \mathcal{S}, \lambda) \vdash Q_2\{^M/_x\}$ and by rule $\mathcal{T}_{par}$ we have that $(E, \mathcal{S}, \lambda) \vdash Q_1\{^M/_x\} \mid Q_2\{^M/_x\}$ $(= Q\{^M/_x\})$

Case $Q = !^i Q'$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash !^i Q', bn(!^i Q') \cap fn(M) = \emptyset$ and $x \notin bv(!^i Q')$ then by rule $\mathcal{T}_{repl}$ and by definition of $bn()$ and $bv()$ we have that $\lambda = \emptyset, (E'[i \mapsto (l, l)], S, \lambda) \vdash Q', bn(Q') \cap fn(M) = \emptyset$ and $x \notin bv(Q')$ then by inductive hypothesis $(E[i \mapsto (l, l)], S, \lambda)Q'\{^M/_x\}$ and by rule $\mathcal{T}_{repl}$ we have that $(E, S, \lambda) \vdash !^i Q'\{^M/_x\}$

Case $Q = \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q', bn(\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q') \cap fn(M) = \emptyset$ and $x \notin bv(\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q')$ then by rule $\mathcal{T}_{new}$ and by definition of $bn()$ and $bv()$ we have that $(E[a \mapsto (a_0[E_1(M_1), \ldots, E_1(M_n)], a_0[E_2(M_1), \ldots, E_2(M_n)])], \mathcal{S}, \lambda) \vdash Q', bn(Q') \cap fn(M) = \emptyset, x \notin bv(Q')$ then by inductive hypothesis $(E[a \mapsto (a_0[E_1(M_1), \ldots, E_1(M_n)], a_0[E_2(M_1), \ldots, E_2(M_n)])], \mathcal{S}, \lambda) \vdash Q'\{^M/_x\}$ and by rule $\mathcal{T}_{new}$ we have that
$(E, \mathcal{S}, \lambda) \vdash \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'\{^M/_x\}$ $(= Q\{^M/_x\})$

Case $Q = \texttt{in}(N, y); Q'$.

Note first that by hypothesis $x \notin bv(Q)$ and by definition $y \in bv(Q)$, thus $y \neq x$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash \mathtt{in}(N, y); Q', bn(\mathtt{in}(N, y); Q') \cap fn(M) = \emptyset$ and $x \notin bv(\mathtt{in}(N, y); Q')$ then by rule $\mathcal{T}_{in}$ and by definition of $bn()$ and $bv()$ we have that $bn(Q') \cap fn(M) = \emptyset$ and $x \notin bv(Q')$ and

$$\bigwedge \quad (E'[y \mapsto (p_1, p_2)], \mathcal{R}, \lambda) \vdash Q'$$
$$\forall p_1, p_2 | \begin{cases} \exists \mathcal{R} | (\mathcal{S} \leq \mathcal{R} \wedge \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j)| \ j \in \lambda]) \wedge \\ msg(E'_1(\bar{\mathcal{R}}_1), E'_1(\mathtt{fst}(N)), p_1, E'_2(\bar{\mathcal{R}}_2), E'_2(\mathtt{snd}(N)), p_2) \in \mathcal{F}, \\ input(E'_1(\bar{\mathcal{R}}_1), E'_1(\mathtt{fst}(N)), E'_2(\bar{\mathcal{R}}_2), E'_2(\mathtt{snd}(N))) \in \mathcal{F} \end{cases}$$

then by inductive hypothesis

$$\bigwedge \quad (E[y \mapsto (p_1, p_2)], \mathcal{R}, \lambda) \vdash Q'\{^M/_x\}$$
$$\forall p_1, p_2 | \begin{cases} \exists \mathcal{R} | (\mathcal{S} \leq \mathcal{R} \wedge \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j)| \ j \in \lambda]) \wedge \\ msg(E'_1(\bar{\mathcal{R}}_1), E'_1(\mathtt{fst}(N)), p_1, E'_2(\bar{\mathcal{R}}_2), E'_2(\mathtt{snd}(N)), p_2) \in \mathcal{F} \\ input(E'_1(\bar{\mathcal{R}}_1), E'_1(\mathtt{fst}(N)), E'_2(\bar{\mathcal{R}}_2), E'_2(\mathtt{snd}(N))) \in \mathcal{F} \end{cases}$$

by the first part of this lemma

$$\bigwedge \quad (E[y \mapsto (p_1, p_2)], \mathcal{R}, \lambda) \vdash Q'\{^M/_x\}$$
$$\forall p_1, p_2 | \begin{cases} \exists \mathcal{R} | (\mathcal{S} \leq \mathcal{R} \wedge \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j)| \ j \in \lambda]) \wedge \\ msg(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(N\{^M/_x\})), p_1, E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(N\{^M/_x\})), p_2) \in \mathcal{F} \\ input(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(N\{^M/_x\})), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(N\{^M/_x\}))) \in \mathcal{F} \end{cases}$$

then by rule $\mathcal{T}_{in}$ we have that $(E, \mathcal{S}, \lambda) \vdash \mathtt{in}(N\{^M/_x\}, y); Q'\{^M/_x\}$ $(= Q\{^M/_x\})$

Case $Q = \mathtt{out}(N_1, N_2); Q'$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash \mathtt{out}(N_1, N_2); Q', bn(\mathtt{out}(N_1, N_2); Q') \cap fn(M) = \emptyset$, and $x \notin bv(\mathtt{out}(N_1, N_2); Q')$ then by rule $\mathcal{T}_{out}$ and by definition of $bn()$ and $bv()$ we have that
$msg(E'_1(\bar{\mathcal{S}}_1), E'_1(\mathtt{fst}(N_1)), E'_1(\mathtt{fst}(N_2)), E'_2(\bar{\mathcal{S}}_2), E'_2(\mathtt{snd}(N_1)),$
$E'_2(\mathtt{snd}(N_2))) \in \mathcal{F}_{C'A_0}, (E', \mathcal{S}, \lambda) \vdash Q', bn(Q') \cap fn(M) = \emptyset$, and $x \notin bv(Q')$ then by inductive hypothesis
$msg(E'_1(\bar{\mathcal{S}}_1), E'_1(\mathtt{fst}(N_1)), E'_1(\mathtt{fst}(N_2)), E'_2(\bar{\mathcal{S}}_2), E'_2(\mathtt{snd}(N_1)),$
$E'_2(\mathtt{snd}(N_2))) \in \mathcal{F}_{C'A_0} \wedge (E, \mathcal{S}, \lambda) \vdash Q'\{^M/_x\}$ then by the first part

of this lemma (Lemma C.21)

$msg(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(N_1\{^M/_x\})), E_1(\mathtt{fst}(N_2\{^M/_x\})), E_2(\bar{\mathcal{S}}_2),$
$E_2(\mathtt{snd}(N_1\{^M/_x\})), E_2(\mathtt{snd}(N_2\{^M/_x\}))) \in \mathcal{F}_{C'A_0} \wedge (E', \mathcal{S}, \lambda) \vdash$
$Q'\{^M/_x\}$ and by rule $\mathcal{T}_{out}$ we have that
$(E, \mathcal{S}, \lambda) \vdash \mathtt{out}(N_1\{^M/_x\}, N_2\{^M/_x\}); Q'\{^M/_x\} (= Q\{^M/_x\})$

Case $Q = \mathtt{let}\ y = D\ \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2$.

Note first that since by hypothesis $x \notin bv(Q)$ and by definition $y \in bv(Q)$, then $y \neq x$.

By hypothesis $(E', \mathcal{S}, \lambda) \vdash \mathtt{let}\ y = D\ \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2$,
$bn(\mathtt{let}\ y = D\ \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2) \cap fn(M) = \emptyset$ and
$x \notin bv(\mathtt{let}\ y = D\ \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2)$ then by rule $\mathcal{T}_{let}$ and by definition of $bn()$ and $bv()$ we have that

$(\forall\ p_1, p_2\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and
$E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2(E'[y \mapsto (p1, p2)], \mathcal{S}, \lambda) \vdash Q_1)$

$(\text{if } \nexists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$(E', \mathcal{S}, \lambda) \vdash Q_2)$

$(\text{if } \exists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$bad \in \mathcal{F}_{C'A_0})$

$(\text{if } \nexists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \exists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$bad \in \mathcal{F}_{C'A_0})$

and $bn(Q_1) \cap fn(M) = \emptyset, bn(Q_2) \cap fn(M) = \emptyset$, and
$x \notin bv(Q_1), x \notin bv(Q_2)$ then by inductive hypothesis

$(\forall\ p_1, p_2\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and
$E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2(E[y \mapsto (p1, p2)], \mathcal{S}, \lambda) \vdash Q_1\{^M/_x\})$

$(\text{if } \nexists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$(E, \mathcal{S}, \lambda) \vdash Q_2\{^M/_x\})$

$(\text{if } \exists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$bad \in \mathcal{F}_{C'A_0})$

$(\text{if } \nexists\ p_1,\ E'_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \exists\ p_2,\ E'_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then}$
$bad \in \mathcal{F}_{C'A_0})$

then by the first part of this lemma (Lemma C.21)

$(\forall\ p_1, p_2\ E_1(\texttt{fst}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_1$ and

$E_2(\texttt{snd}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_2(E[y \mapsto (p1, p2)], \mathcal{S}, \lambda) \vdash Q_1\{^M/_x\})$

(if $\nexists\, p_1,\ E_1(\texttt{fst}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_1$ and

$\nexists\, p_2,\ E_2(\texttt{snd}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_2$ then $(E, \mathcal{S}, \lambda) \vdash Q_2\{^M/_x\})$

(if $\exists\, p_1,\ E_1(\texttt{fst}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_1$ and

$\nexists\, p_2,\ E_2(\texttt{snd}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

(if $\nexists\, p_1,\ E_1(\texttt{fst}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_1$ and

$\exists\, p_2,\ E_2(\texttt{snd}(D\{^M/_x\})) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

Finally, by rule $\mathcal{T}_{let}$ we have that

$(E, \mathcal{S}, \lambda) \vdash$

$\texttt{let}\ y = D\{^M/_x\}\ \texttt{in}\ Q_1\{^M/_x\}\ \texttt{else}\ Q_2\{^M/_x\}\ (= Q\{^M/_x\})$

Case $Q = \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q', bn(\texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q') \cap fn(M) = \emptyset$ and

$x \notin bv(\texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q')$ then by rule $\mathcal{T}_{lock}$ and by definition of $bn()$ and $bv()$ we have that

$$\mathcal{R}|\begin{cases} \bigwedge & (E', \mathcal{R}, \lambda \cup \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j)|\ j \in \lambda] \end{cases}$$

and $bn(Q') \cap fn(M) = \emptyset, x \notin bv(Q')$ then by inductive hypothesis we have that

$$\mathcal{R}|\begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \cup \{j_1, \ldots, j_m\}) \vdash Q'\{^M/_x\} \\ \mathcal{R} \leq \mathcal{S} \wedge \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j)|\ j \in \lambda] \end{cases}$$

then by rule $\mathcal{T}_{lock}$ we have that

$(E, \mathcal{S}, \lambda) \vdash \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'\{^M/_x\}\ (= Q\{^M/_x\})$

Case $Q = \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'$,

$bn(\texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q') \cap fn(M) = \emptyset$ and
$x \notin bv(\texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q')$ then by rule $\mathcal{T}_{unlock}$ and by definition of $bn()$ and $bv()$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E', \mathcal{R}, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

and $bn(Q') \cap fn(M) = \emptyset, x \notin bv(Q')$ then by inductive hypothesis we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q'\{^M/_x\} \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

then by rule $\mathcal{T}_{unlock}$ we have that
$(E, \mathcal{S}, \lambda) \vdash \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'\{^M/_x\}\ (= Q\{^M/_x\})$

Case $Q = s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q'$.
By hypothesis $(E, \mathcal{S}, \lambda) \vdash s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q'$,
$bn(s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q') \cap fn(M) = \emptyset$
and $x \notin bv(s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q')$ then by rule $\mathcal{T}_{assign}$ and by definition of $bn()$ and $bv()$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & \begin{array}{c} \mathcal{R} \leq \mathcal{R}[j_k \mapsto E'(N_k) \mid 1 \leq k \leq m] \\ (E', \mathcal{R}[j_k \mapsto E'(N_k) \mid 1 \leq k \leq m], \lambda) \vdash Q' \end{array} \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

and $bn(Q') \cap fn(M) = \emptyset$ and $x \notin bv(Q')$ then by inductive hypothesis

$$\mathcal{R}| \begin{cases} \bigwedge & \begin{array}{c} \mathcal{R} \leq \mathcal{R}[j_k \mapsto E'(N_k) \mid 1 \leq k \leq m] \\ (E, \mathcal{R}[j_k \mapsto E'(N_k) \mid 1 \leq k \leq m], \lambda) \vdash Q'\{^M/_x\} \end{array} \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

by the first part of this lemma (Lemma C.21)

$$\mathcal{R}|\begin{cases} \bigwedge & \mathcal{R} \leq \mathcal{R}[j_k \mapsto E(N_k\{^M/_x\}) \mid 1 \leq k \leq m] \\ & (E, \mathcal{R}[j_k \mapsto E(N_k\{^M/_x\}) \mid 1 \leq k \leq m], \lambda) \vdash Q'\{^M/_x\} \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid j \in \lambda] & \end{cases}$$

Finally, by rule $\mathcal{T}_{assign}$ we have that

$(E, \mathcal{S}, \lambda) \vdash s_{j_1}, \ldots, s_{j_m} := N_1\{^M/_x\}, \ldots, N_m\{^M/_x\}; Q'\{^M/_x\}$ $(= Q\{^M/_x\})$

Case $Q = \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'$.

Note first that by hypothesis $x \notin bv(Q)$ and by definition $y_1, \ldots, y_m \in bv(Q)$, thus $x \notin \{y_1, \ldots, y_m\}$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'$, $bn(\texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q') \cap fn(M) = \emptyset$ and $x \notin bv(\texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ x_1, \ldots, x_m; Q')$ then by rule $\mathcal{T}_{read}$ and by definition of $bn()$ and $bv()$ we have that

$$\mathcal{R}|\begin{cases} \bigwedge & (E' \cup \{y_k \mapsto (E'_1(\mathcal{R}_1[j_k]), E'_2(\mathcal{R}_2[j_k]))\}, \mathcal{R}, \lambda) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid j \in \lambda] & \end{cases}$$

and $bn(Q') \cap fn(M) = \emptyset$ and $x \notin bv(Q')$ then by inductive hypothesis

$$\mathcal{R}|\begin{cases} \bigwedge & (E \cup \{y_k \mapsto (E_1(\mathcal{R}_1[j_k]), E_2(\mathcal{R}_2[j_k]))\}, \mathcal{R}, \lambda) \vdash Q'\{^M/_x\} \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid j \in \lambda] & \end{cases}$$

and, by rule $\mathcal{T}_{read}$ we have that

$(E, \mathcal{S}, \lambda) \vdash \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'\ (= Q\{^M/_x\})$

□

C.7.2 Type Propagation Lemma

Lemma C.22. *Let Q, be an instrumented unevaluated stateful biprocess, E an environment (from names and variables to pairs of patterns), $\mathcal{S}$ and $\mathcal{T}$ states (from cell names to terms), and λ a sequence of cell indices, such that $\mathcal{S} \leq \mathcal{T}$ and $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}(j) \mid j \in \lambda]$*

$$(E, \mathcal{S}, \lambda) \vdash Q \Rightarrow (E, \mathcal{T}, \lambda)Q$$

.

Proof. We prove this by induction on the depth d of the proof of $(E, \mathcal{S}, \lambda) \vdash Q$.

d = 0) In this case, $Q = 0$, and according to our type rule $\mathcal{T}_{nil}$, $(E, \mathcal{T}, \lambda) \vdash Q$.

d > 0) We proceed by case analysis on the structure of Q.

Case $Q = Q_1 \mid Q_2$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash Q_1 \mid Q_2$ by rule $\mathcal{T}_{par}$ $(E, \mathcal{S}, \lambda) \vdash Q_1 \wedge (E, \mathcal{S}, \lambda) \vdash Q_2 \wedge \lambda = \emptyset$ then by inductive hypothesis $(E, \mathcal{T}, \lambda) \vdash Q_1 \wedge (E, \mathcal{T}, \lambda) \vdash Q_2 \wedge \lambda = \emptyset$ then by rule $\mathcal{T}_{par}$ we have that $(E, \mathcal{T}, \lambda) \vdash Q_1 \mid Q_2$

Case $Q = !^i Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash Q'$ by rule $\mathcal{T}_{repl}$ $(E[i \mapsto (l, l)], \mathcal{S}, \lambda) \vdash Q' \wedge \lambda = \emptyset$ then by inductive hypothesis $(E[i \mapsto (i, i)], \mathcal{T}, \lambda) \vdash Q' \wedge \lambda = \emptyset$ then by rule $\mathcal{T}_{repl}$ we have that $(E, \mathcal{T}, \lambda) \vdash Q'$

Case $Q = \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'$ by rule $\mathcal{T}_{new}$ we have that
$(E[a \mapsto (a_0[E_1(M_1), \ldots, E_1(M_n)], a_0[E_2(M_1), \ldots, E_2(M_n)])], \mathcal{S}, \lambda) \vdash Q'$
and by inductive hypothesis

$(E[a \mapsto (a_0[E_1(M_1), \ldots, E_1(M_n)], a_0[E_2(M_1), \ldots, E_2(M_n)])], \mathcal{T}, \lambda) \vdash Q'$

then by rule $\mathcal{T}_{new}$ we have that $(E, \mathcal{T}, \lambda) \vdash \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'$

Case $Q = \texttt{in}(N, x); Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{in}(N, x); Q'$ then by rule $\mathcal{T}_{in}$ we have that

$$\bigwedge \begin{cases} (E[x \mapsto (p_1, p_2)], \mathcal{R}, \lambda) \vdash Q' \\ \forall p_1, p_2 \mid \begin{cases} \mathcal{R}\ (\mathcal{S} \leq \mathcal{R} \wedge \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid\ j \in \lambda] \wedge \\ msg(E_1(\bar{\mathcal{R}}_1), E_1(\texttt{fst}(N)), p_1, E_2(\bar{\mathcal{R}}_2), E_2(\texttt{snd}(N)), p_2) \in \mathcal{F}_{C'A0} \\ input(E_1(\bar{\mathcal{R}}_1), E_1(\texttt{fst}(N)), E_2(\bar{\mathcal{R}}_2), E_2(\texttt{snd}(N))) \in \mathcal{F}_{C'A0} \end{cases} \end{cases}$$

then by transitivity of $\leq$ and since $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}_j \mid j \in \lambda]$ we have that

$$\bigwedge \begin{cases} (E[x \mapsto (p_1, p_2)], \mathcal{R}, \lambda) \vdash Q' \\ \forall p_1, p_2 \mid \begin{cases} \mathcal{R}\ (\mathcal{T} \leq \mathcal{R} \wedge \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{T}(j) \mid\ j \in \lambda] \wedge \\ msg(E_1(\bar{\mathcal{R}}_1), E_1(\texttt{fst}(N)), p_1, E_2(\bar{\mathcal{R}}_2), E_2(\texttt{snd}(N)), p_2) \in \mathcal{F}_{C'A0} \\ input(E_1(\bar{\mathcal{R}}_1), E_1(\texttt{fst}(N)), E_2(\bar{\mathcal{R}}_2), E_2(\texttt{snd}(N))) \in \mathcal{F}_{C'A0} \end{cases} \end{cases}$$

then by rule $\mathcal{T}_{in}$ we have that $(E, \mathcal{T}, \lambda) \vdash \texttt{in}(N, x); Q'\ (= Q)$

Case $Q = \texttt{out}(N_1, N_2); Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{out}(N_1, N_2); Q'$, then by rule $\mathcal{T}_{out}$ we have that

$msg(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(N_1)), E_1(\texttt{fst}(N_2)), E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(N_1)),$
$E_2(\texttt{snd}(N_2))) \in \mathcal{F}_{C'A_0}, (E, \mathcal{S}, \lambda) \vdash Q'$, then by inductive hypothesis
$msg(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(N_1)), E_1(\texttt{fst}(N_2)), E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(N_1)),$
$E_2(\texttt{snd}(N_2))) \in \mathcal{F}_{C'A_0} \wedge (E, \mathcal{T}, \lambda) \vdash Q'$ and since $\mathcal{S} \leq \mathcal{T}$ we have that
$msg(E_1(\bar{\mathcal{T}}_1), E_1(\texttt{fst}(N_1)), E_1(\texttt{fst}(N_2)), E_2(\bar{\mathcal{T}}_2), E_2(\texttt{snd}(N_1)),$
$E_2(\texttt{snd}(N_2))) \in \mathcal{F}_{C'A_0} \wedge (E, \mathcal{T}, \lambda) \vdash Q'$ and by rule $\mathcal{T}_{out}$ we have that
$(E, \mathcal{T}, \lambda) \vdash \texttt{out}(N_1, N_2); Q'\ (= Q)$

Case $Q = \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2$, then by rule $\mathcal{T}_{let}$ we have that

$(\forall\ p_1, p_2\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2(E[x \mapsto (p1, p2)], \mathcal{S}, \lambda) \vdash Q_1)$

(if $\nexists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\nexists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $(E, \mathcal{S}, \lambda) \vdash Q_2)$

(if $\exists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\nexists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

(if $\nexists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\exists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

then by inductive hypothesis

$(\forall\ p_1, p_2\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$E_2'(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2(E[x \mapsto (p1, p2)], \mathcal{T}, \lambda) \vdash Q_1)$

(if $\nexists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\nexists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $(E, \mathcal{T}, \lambda) \vdash Q_2)$

(if $\exists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\nexists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

(if $\nexists\ p_1,\ E_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and

$\exists\ p_2,\ E_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ then $bad \in \mathcal{F}_{C'A_0}$)

and by rule $\mathcal{T}_{let}$ we have that $(E, \mathcal{T}, \lambda) \vdash \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2$ $(= Q)$

Case $Q = \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'$, then by rule $\mathcal{T}_{lock}$ we have that

$$\mathcal{R}\Big| \begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \cup \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

then by transitivity of $\leq$ and since $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}_j \mid j \in \lambda]$ we have that

$$\mathcal{R}\Big| \begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \cup \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{T} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{T}(j) |\ j \in \lambda] & \end{cases}$$

then by rule $\mathcal{T}_{lock}$ we have that $(E, \mathcal{T}, \lambda) \vdash \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'\ (= Q)$

Case $Q = \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'$, then by rule $\mathcal{T}_{unlock}$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

then by transitivity of $\leq$ and since $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}_j \mid j \in \lambda]$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E, \mathcal{R}, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q' \\ \mathcal{R} \leq \mathcal{T} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{T}(j) |\ j \in \lambda] & \end{cases}$$

then by rule $\mathcal{T}_{unlock}$ we have that $(E, \mathcal{T}, \lambda) \vdash \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'\ (= Q)$

Case $Q = s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q'$, then by rule $\mathcal{T}_{assign}$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & \mathcal{R} \leq \mathcal{R}[j_k \mapsto E(N_k) \mid 1 \leq k \leq m] \\ & (E, \mathcal{R}[j_k \mapsto E(N_k) \mid 1 \leq k \leq m], \lambda) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

then by transitivity of $\leq$ and since $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}_j \mid j \in \lambda]$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & \mathcal{R} \leq \mathcal{R}[j_k \mapsto E(N_k) \mid 1 \leq k \leq m] \\ & (E, \mathcal{R}[j_k \mapsto E(N_k) \mid 1 \leq k \leq m], \lambda) \vdash Q' \\ \mathcal{R} \leq \mathcal{T} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{T}(j) |\ j \in \lambda] & \end{cases}$$

by rule $\mathcal{T}_{assign}$ we have that

$(E, \mathcal{T}, \lambda) \vdash s_{j_1}, \ldots, s_{j_m} := N_1, \ldots, N_m; Q'\ (= Q)$

Case $Q = \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'$.

By hypothesis $(E, \mathcal{S}, \lambda) \vdash \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'$, then by rule $\mathcal{T}_{read}$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E \cup \{y_k \mapsto (E_1(\mathcal{R}_1[j_k]), E_2(\mathcal{R}_2[j_k]))\}, \mathcal{R}, \lambda) \vdash Q' \\ \mathcal{R} \leq \mathcal{S} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) |\ j \in \lambda] & \end{cases}$$

then by transitivity of $\leq$ and since $\mathcal{T} = \mathcal{T}[j \mapsto \mathcal{S}_j \mid j \in \lambda]$ we have that

$$\mathcal{R}| \begin{cases} \bigwedge & (E \cup \{y_k \mapsto (E_1(\mathcal{R}_1[j_k]), E_2(\mathcal{R}_2[j_k]))\}, \mathcal{R}, \lambda) \vdash Q' \\ \mathcal{R} \leq \mathcal{T} \wedge & \\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{T}(j) |\ j \in \lambda] & \end{cases}$$

and, by rule $\mathcal{T}_{read}$ we have that

$(E, \mathcal{T}, \lambda) \vdash \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ y_1, \ldots, y_m; Q'\ (= Q)$

□

C.7.3 Typability of the Adversary (C[])

Let $E0 = a \mapsto (a[], a[]) | a \in fn(C'[A'_0])$. Let $\mathcal{S}0 = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$

Lemma C.23. Typability of the adversary: *Let P' be a subprocess of C'. Let E be an environment (from names and variables to pairs of patterns) such that for all $a \in fn(P')$, $att(E_1(\bar{\mathcal{S}}0_1), E_1(a), E_2(\bar{\mathcal{S}}0_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ and for all $x \in fv(P')$, $att(E_1(\bar{\mathcal{S}}0_1), E_1(x), E_2(\bar{\mathcal{S}}0_2), E_2(x)) \in \mathcal{F}_{CA'_0}$. Let $\mathcal{S}$ be a state (from cell names to terms) and λ a set of cell indices. We show that $(E, \mathcal{S}0, \emptyset) \vdash P'$ by induction on P', similarly to [ARR11].*

Proof. We first prove by induction on the depth d of P' that, if

i $E0 \subseteq E$ and

ii $\mathcal{S}_0 \leq \mathcal{S}$ and

iii $(bn(P') \cup bv(P')) \cap dom(E) = \emptyset$ and

iv $\forall\, a \in fn(P'), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ and

v $\forall\, x \in fv(P'), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{CA'_0}$ and

vi $\forall\, i \in 1, \ldots, n, i \in \lambda$ if and only if P' is in the scope of a $lock \ldots s_i \ldots$ in C'

then

$$(E, \mathcal{S}, \lambda) \vdash P'$$

Base case ($d = 0$). In that case $P' = 0$ and according to our typing system

$$\frac{}{(E, \mathcal{S}, \lambda) \vdash 0 (= P')}\ \mathcal{T}_{nil}$$

Inductive case ($d > 0$). We proceed by case analysis on the structure of P'.

- Case $P' = Q_1 \mid Q_2$. First note that no parallel composition can occur in the scope of a lock, so $\lambda = \emptyset$.

  (i) By hypothesis, $E0 \subseteq E$.

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S}$.

  (iii) By definition of P' we have that $bn(Q_1) \cup bv(Q_1) \subseteq bn(P') \cup bv(P')$, and $bn(Q_2) \cup bv(Q_2) \subseteq bn(P') \cup bv(P')$ then (by hyp) we have that $(bn(Q_1) \cup bv(Q_1)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$ and $(bn(Q_2) \cup bv(Q_2)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

(iv) By definition of P' we have that $fn(Q_1) \subseteq fn(P')$ and $fn(Q_2) \subseteq fn(P')$ then (by hyp.) $\forall\, a \in fn(Q_1)$ we have that $att(E_1(\bar{\mathcal{S}}_1), E(a), E_2(\bar{\mathcal{S}}_2), E(a)) \in \mathcal{F}_{C'A_0}$ and $\forall\ a \in fn(Q_2)$ $att(E_1(\bar{\mathcal{S}}_1), E(a), E_2(\bar{\mathcal{S}}_2), E(a)) \in \mathcal{F}_{C'A_0}$

(v) By definition of P' we have that $fv(Q_1) \subseteq fv(P')$ and $fv(Q_2) \subseteq fv(P')$ then (by hyp.) $\forall\, x \in fv(Q_1)$ we have that $att(E_1(\bar{\mathcal{S}}_1), E(x), E_2(\bar{\mathcal{S}}_2), E(x)) \in \mathcal{F}_{C'A_0}$ and $\forall\ x \in fv(Q_2)$ $att(E_1(\bar{\mathcal{S}}_1), E(x), E_2(\bar{\mathcal{S}}_2), E(x)) \in \mathcal{F}_{C'A_0}$

(vi) P' is in the scope of a $lock \ldots s_i \ldots$ if and only if Q_1 and Q_2 are too, so Q_1, λ and Q_2, satisfy condition (vi) by hypothesis.

Thus $(Q_1, E, \mathcal{S}, \lambda)$ and $(Q_2, E, \mathcal{S}, \lambda)$ satisfy conditions $(i) - (vi)$, so we can apply our inductive hypothesis $(E, \mathcal{S}, \lambda) \vdash Q_1$ and $(E, \mathcal{S}, \lambda) \vdash Q_2$ and, according to our typing system

$$\frac{(E, \mathcal{S}, \lambda) \vdash Q_1 \qquad (E, \mathcal{S}, \lambda) \vdash Q_2}{(E, \mathcal{S}, \lambda) \vdash (Q_1 \mid Q_2) = P'}\ \mathcal{T}_{par}$$

- Case $P' = !^i Q$. First note that no parallel composition can occur in the scope of a lock, so $\lambda = \emptyset$.

  (i) By hypothesis, $E0 \subseteq E$. Let $E \cup \{i \mapsto (l, l)\} = E'$ we have that $E0 \subseteq E \subseteq E'$

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) = bn(P') \cup bv(P')$, then (by hyp) we have that $(bn(Q) \cup bv(Q)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) By definition of P' we have that $fn(Q) = fn(P')$ then (by hyp.) $\forall\, a \in fn(Q)$ we have that
  $att(E_1(\bar{\mathcal{S}}_1), E(a), E_2(\bar{\mathcal{S}}_2), E(a)) \in \mathcal{F}_{C'A_0}$ since $E(a) = E'(a)$ and $E(\bar{\mathcal{S}}) =$

$E'(\bar{\mathcal{S}})$ we have that
$att(E'_1(\bar{\mathcal{S}}_1), E'(a), E'_2(\bar{\mathcal{S}}_2), E'(a)) \in \mathcal{F}_{C'A_0}$

(v) By definition of P' we have that $fv(Q) = fv(P')$ then (by hyp.) $\forall\, x \in fv(Q)$ we have that
$att(E_1(\bar{\mathcal{S}}_1), E(x), E_2(\bar{\mathcal{S}}_2), E(x)) \in \mathcal{F}_{C'A_0}$ since $E(x) = E'(x)$ and $E(\bar{\mathcal{S}}) = E'(\bar{\mathcal{S}})$ we have that
$att(E'_1(\bar{\mathcal{S}}_1), E'(x), E'_2(\bar{\mathcal{S}}_2), E'(x)) \in \mathcal{F}_{C'A_0}$

(vi) P' is in the scope of a $lock \ldots s_i \ldots$ if and only if Q is too, so Q, λ satisfies condition (vi) by hypothesis.

Thus $(Q, E', \mathcal{S}, \lambda)$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E', \mathcal{S}, \lambda) \vdash Q$ and according to our typing system

$$\frac{\forall\, l \quad (E', \mathcal{S}, \lambda) \vdash Q}{(E, \mathcal{S}, \lambda) \vdash (!^i Q) = P'}\ \mathcal{T}_{repl}$$

- Case $P' = \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q$. By hypothesis on C', $bn(C') \cap bn(A_0) = \emptyset$, thus $a \notin bn(P')$. Let $E' = E \cup \{a \mapsto (a[E_1(\texttt{fst}(M_1)), \ldots, E_1(\texttt{fst}(M_n))], a[E_2(\texttt{snd}(M_1)), \ldots, E_2(\texttt{snd}(M_n))])\}$.

  (i) By hypothesis, $E0 \subseteq E$. We have that $E0 \subseteq E \subseteq E'$

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) \subseteq bn(P') \cup bv(P')$, then (by hyp) we have that $(bn(Q) \cup bv(Q)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) Let $b \in fn(Q)$. Then either $b \neq a$ or $b = a$.
  If $b \neq a$ then $b \in fn(P')$ and by hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(b), E_2(\bar{\mathcal{S}}_2), E_2(b)) \in \mathcal{F}_{C'A_0}$.
  If $b = a$ then by rule (RF) we have that $att(xs, a[E_1(M_1), \ldots, E_1(M_n)], xs', a[E_2(M_1), \ldots, E_2(M_n)]) \in \mathcal{F}_{C'A_0}$ for all xs, xs' hence $att(E'_1(\bar{\mathcal{S}}_1),$

$a[E_1(\texttt{fst}(M_1)), \ldots, E_1(\texttt{fst}(M_n))], E'_2(\bar{\mathcal{S}}_2), a[E_2(\texttt{snd}(M_1)), \ldots,$
$E_2(\texttt{snd}(M_n))]) \in \mathcal{F}_{C'A_0}$

(v) Let $x \in fv(Q)$ then $x \in fv(P')$ and by hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(x),$ $E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$, since $E(x) = E'(x)$ we have that $att(E'_1(\bar{\mathcal{S}}_1),$ $E'_1(x), E'_2(\bar{\mathcal{S}}_2), E'_2(x)) \in \mathcal{F}_{C'A_0}$

(vi) P' is in the scope of a $lock \ldots s_i \ldots$ if and only if Q is too, so Q, λ satisfies condition (vi) by hypothesis.

Thus $(Q, E', \mathcal{S}, \lambda)$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E', \mathcal{S}, \lambda) \vdash Q$ and according to our typing system

$$\frac{(E', \mathcal{S}, \lambda) \vdash Q}{(E, \mathcal{S}, \lambda) \vdash (\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q) = P'}\ \mathcal{T}_{new}$$

- Case $P' = \texttt{in}(M, x); Q$. Let $\mathcal{R}$ be a state such that $\mathcal{S} \leq \mathcal{R}$ and $\mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid j \in \lambda]$, let N be a term such that $E(N) = (p_1, p_2)$ and $msg(E_1(\mathcal{R}_1), E_1(\texttt{fst}(M)), p_1, E_2(\mathcal{R}_2), E_2(\texttt{snd}(M)), p2) \in \mathcal{F}_{C'A_0}, E' = E \cup \{x \mapsto (p_1, p_2)\}$.

  (i) By hypothesis, $E0 \subseteq E \subseteq (E \cup \{x \mapsto (p_1, p_2)\}) = E'$. Moreover, $bv(P') \cap dom(E) = \emptyset, x \notin dom(E)$. Thus E' is indeed an environment, i.e. a function from variables and names to pairs of patterns.

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) \subseteq bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) Let $a \in fn(Q)$ by definition $a \in fn(P')$ and by hypothesis $att(E_1(\bar{\mathcal{S}}_1),$ $E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$. Since $E(a) = E'(a)$ and $E(\bar{\mathcal{S}}) = E'(\bar{\mathcal{S}})$ then $att(E'_1(\bar{\mathcal{S}}_1), E'_1(a), E'_2(\bar{\mathcal{S}}_2), E'_2(a)) \in \mathcal{F}_{C'A_0}$. $\mathcal{S} \leq \mathcal{R}$ thus $att(E'_1(\bar{\mathcal{R}}_1),$ $E'_1(a), E'_2(\bar{\mathcal{R}}_2), E'_2(a)) \in \mathcal{F}_{C'A_0}$

  (v) Let $y \in fv(Q)$ then either $y \in fv(P')$ or $y = x$.
  If $y \in fv(P')$ then $att(E_1(\bar{\mathcal{S}}_1), E_1(y), E_2(\bar{\mathcal{S}}_2), E_2(y)) \in \mathcal{F}_{C'A_0}$.

Since $E(y) = E'(y)$ and $E(\bar{\mathcal{S}}) = E'(\bar{\mathcal{S}})$ we have that $att(E'_1(\bar{\mathcal{S}}_1), E'_1(x), E'_2(\bar{\mathcal{S}}_2), E'_2(x)) \in \mathcal{F}_{C'A_0}$. $\mathcal{S} \leq \mathcal{R}$ thus $att(E'_1(\bar{\mathcal{R}}_1), E'_1(y), E'_2(\bar{\mathcal{R}}_2), E'_2(y)) \in \mathcal{F}_{C'A_0}$.
If $y = x$ we have that $\forall\, u \in fn(M) \cup fv(M)$ $att(E_1(\bar{\mathcal{S}}_1), E_1(u), E_2(\bar{\mathcal{S}}_2), E_2(u)) \in \mathcal{F}_{C'A_0}$ and by Lemma C.20 $att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{S}}_2), E_2(\mathtt{snd}(M))) \in \mathcal{F}_{C'A_0}$. Since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(M))) \in \mathcal{F}_{C'A_0}$ and by rule RL $att(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(N)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(N))) \in \mathcal{F}_{C'A_0}$. Since $E'(y) = (p_1, p_2)$ and $E(\bar{\mathcal{R}}) = E'(\bar{\mathcal{R}})$ we have $att(E'_1(\bar{\mathcal{R}}_1), p_1, E'_2(\bar{\mathcal{R}}_1), p_2) \in \mathcal{F}_{C'A_0}$.

(vi) P' is in the scope of a $lock \ldots s_i \ldots$ if and only if Q is too, so Q, λ satisfies condition (vi) by hypothesis.

Thus $(Q, E', \mathcal{R}, \lambda)$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E', \mathcal{R}, \lambda) \vdash Q$. Since $att(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(M))) \in \mathcal{F}_{C'A_0}$ by rule RI we have that $input(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(M)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(M))) \in \mathcal{F}_{C'A_0}$ and according to our typing system

$$\frac{\begin{array}{c}\forall \mathcal{R}\ (\mathcal{S} \leq \mathcal{R}\ \wedge\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid\ j \in \lambda] \wedge \\ \forall\ N : E(N) = (p_1, p_2), \\ msg(\bar{\mathcal{R}}_1, E_1(\mathtt{fst}(M)), p_1, \bar{\mathcal{R}}_2, E_2(\mathtt{snd}(M)), p_2) \in \mathcal{F} \\ input(\bar{\mathcal{R}}_1, E_1(\mathtt{fst}(M)), \bar{\mathcal{R}}_2, E_2(\mathtt{snd}(M))) \in \mathcal{F}) \\ (E', \mathcal{R}, \lambda) \vdash Q\end{array}}{(E, \mathcal{S}, \lambda) \vdash (\mathtt{in}(M, x); Q) = P'}\ \mathcal{T}_{in}$$

- Case $P' = [s \mapsto M]$. This case cannot occur because by hypothesis no $[s \mapsto M]$ occurs in C'.

- Case $P' = \mathtt{out}(M, N); Q$.

  (i) By hypothesis, $E0 \subseteq E$.

(ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S}$.

(iii) By definition of P' we have that $bn(Q) \cup bv(Q) \subseteq bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

(iv) Let $a \in fn(Q)$ By definition $fn(Q) \subseteq fn(P')$ $a \in fn(P')$ and by hypothesis $\forall\, a \in fn(Q)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$.

(v) Let $x \in fv(Q)$ By definition $fv(Q) \subseteq fv(P')$ hence $x \in fv(P')$ and by hypothesis $\forall\, x \in fv(Q)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$.

(vi) P' is in the scope of a $lock \ldots s_i \ldots$ if and only if Q is too, so Q, λ satisfies condition (vi) by hypothesis.

Thus $(Q, E, \mathcal{S}, \lambda)$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E, \mathcal{S}, \lambda) \vdash Q$. Moreover, by hypothesis

- $\forall\ a \in fn(M) \cup fn(N), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ because $fn(M) \cup fn(N) \subseteq fn(P')$; and
- $\forall\ x \in fv(M) \cup fv(N), att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$ because $fv(M) \cup fv(N) \subseteq fv(P')$; and

Thus according to Lemma C.20 it is the case that $att(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(M)),$
$E_2(\bar{\mathcal{S}}_2),$
$E_2(\texttt{snd}(M))) \in \mathcal{F}_{C'A_0}$ and $att(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(N)), E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(N))) \in \mathcal{F}_{C'A_0}$ and because by (Rs) $att(xs, xc, xs', xc') \wedge$
$attacker(xs, xm, xs', xm') \rightarrow msg(xs, xc, xm, xs', xc', xm') \in \mathcal{F}_{C'A_0}$ we have by resolution that
$msg(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(M)), E_1(\texttt{fst}(N)), E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(M)), E_2(\texttt{snd}(N))) \in$

$\mathcal{F}_{C'A_0}$. Thus, according to our typing system

$$\frac{\begin{array}{c} msg(E_1(\bar{\mathcal{S}}_1), E_1(\texttt{fst}(M)), E_1(\texttt{fst}(N)), \\ E_2(\bar{\mathcal{S}}_2), E_2(\texttt{snd}(M)), E_2(\texttt{snd}(N))) \in \mathcal{F}_{C'A_0} \\ (E, \mathcal{S}, \lambda) \vdash Q \end{array}}{(E, \mathcal{S}, \lambda) \vdash (\texttt{out}(M, N); Q) = P'} \mathcal{T}_{out}$$

- Case $P' = \texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q$. Let $\mathcal{R}$ be a state such that $\mathcal{S} \leq \mathcal{R}$ and $\mathcal{R} = \mathcal{R}[k \mapsto \mathcal{S}(k) | k \in \lambda]$. Let $\lambda' = \lambda \cup \{j_1, \ldots, j_m\}$.

  (i) By hypothesis, $E0 \subseteq E$.

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) = bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) = (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) Let $a \in fn(Q)$ By definition $fn(Q) \subseteq fn(P')$ and $a \in fn(P')$ and by hypothesis $\forall\ a \in fn(Q)$,
  $att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}}_1), E_1(a), E_2(\bar{\mathcal{R}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$.

  (v) Let $x \in fv(Q)$ By definition $fv(Q) \subseteq fv(P')$ hence $x \in fv(P')$ and by hypothesis $\forall\ x \in fv(Q)$,
  $att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}}_1), E_1(x), E_2(\bar{\mathcal{R}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$.

  (vi) P' is in the scope of a $\texttt{lock}\ \ldots s_i \ldots$ if and only if Q is too, and P' is in the scope of a $\texttt{lock}\ \ldots s_i \ldots$ with $i \notin \{j_1, \ldots, j_m\}$ if and only if Q is too. So Q, λ' satisfies condition (vi) by hypothesis.

Thus $(Q, E, \mathcal{R}, \lambda')$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E, \mathcal{R}, \lambda') \vdash Q$. Thus, according to our typing system

$$\frac{\begin{array}{c}\forall \mathcal{R}\ (\mathcal{S} \leq \mathcal{R}\ \wedge\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid\ \ j \in \lambda] \\ (E, \mathcal{R}, \lambda \cup \{i_1, \ldots, i_m\}) \vdash Q)\end{array}}{(E, \mathcal{S}, \lambda) \vdash (\texttt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q) = P'}\ \mathcal{T}_{lock}$$

- Case $P' = \texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q$. Let $\mathcal{R}$ be a state such that $\mathcal{S} \leq \mathcal{R}$ and $\mathcal{R} = \mathcal{R}[k \mapsto \mathcal{S}(k) | k \in \lambda]$. Let $\lambda' = \lambda \setminus \{j_1, \ldots, j_m\}$.

  (i) By hypothesis, $E0 \subseteq E$.

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) = bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) = (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) Let $a \in fn(Q)$ By definition $fn(Q) \subseteq fn(P')$ and $a \in fn(P')$ and by hypothesis $\forall\ a \in fn(Q)$,
  $att(E_1(\bar{\mathcal{S}_1}), E_1(a), E_2(\bar{\mathcal{S}_2}), E_2(a)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}_1}), E_1(a), E_2(\bar{\mathcal{R}_2}), E_2(a)) \in \mathcal{F}_{C'A_0}$.

  (v) Let $x \in fv(Q)$ By definition $fv(Q) \subseteq fv(P')$ hence $x \in fv(P')$ and by hypothesis $\forall\ x \in fv(Q)$,
  $att(E_1(\bar{\mathcal{S}_1}), E_1(x), E_2(\bar{\mathcal{S}_2}), E_2(x)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}_1}), E_1(x), E_2(\bar{\mathcal{R}_2}), E_2(x)) \in \mathcal{F}_{C'A_0}$.

  (vi) P' is in the scope of a $\texttt{lock}\ \ldots s_i \ldots$ if and only if Q is too, and P' is in the scope of a $\texttt{lock}\ \ldots s_i \ldots$ with $i \notin \{j_1, \ldots, j_m\}$ if and only if Q is too. So Q, λ' satisfies condition (vi) by hypothesis.

Thus $(Q, E, \mathcal{R}, \lambda')$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E, \mathcal{R}, \lambda') \vdash Q$. Thus, according to our typing system

$$\frac{\begin{array}{c}\forall \mathcal{R}\ (\mathcal{S} \leq \mathcal{R}\ \wedge\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid\ j \in \lambda] \\ (E, \mathcal{R}, \lambda \setminus \{i_1, \ldots, i_m\}) \vdash Q)\end{array}}{(E, \mathcal{S}, \lambda) \vdash (\texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q) = P'}\ \mathcal{T}_{unlock}$$

- Case $P' = \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ x_1, \ldots, x_m; Q$. Let $\mathcal{R}$ be a state such that $\mathcal{S} \leq \mathcal{R}$ and $\mathcal{R} = \mathcal{R}[k \mapsto \mathcal{S}(k) | k \in \lambda]$. Let $E' = E \cup \{x_k \mapsto (E_1(\texttt{fst}(\mathcal{R}[j_k]), E_2(\texttt{snd}(\mathcal{R}[j_k])))) | 1 \leq k \leq m\}$.

  (i) By hypothesis, $E0 \subseteq E \subseteq E'$. Moreover, $bv(P') \cap dom(E) = \emptyset, x \notin dom(E)$. Thus E is indeed an environment, i.e. a function from variables and names to patterns.

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R}$.

  (iii) By definition of P' we have that $bn(Q) \cup bv(Q) \subseteq bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) Let $a \in fn(Q)$ by definition $a \in fn(P')$ and by hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$. Since $E(a) = E'(a)$ and $E(\bar{\mathcal{S}}) = E'(\bar{\mathcal{S}})$ we have that $att(E'_1(\bar{\mathcal{S}}_1), E'_1(a), E'_2(\bar{\mathcal{S}}_2), E'_2(a)) \in \mathcal{F}_{C'A_0}$. $\mathcal{S} \leq \mathcal{R}$ thus $att(E'_1(\bar{\mathcal{R}}_1), E'_1(a), E'_2(\bar{\mathcal{R}}_2), E'_2(a)) \in \mathcal{F}_{C'A_0}$

  (v) Let $y \in fv(Q)$ then either $y \in fv(P')$ or $y \in \{x_1, \ldots, x_m\}$.
  If $y \in fv(P')$ then $att(E_1(\bar{\mathcal{S}}_1), E_1(y), E_2(\bar{\mathcal{S}}_2), E_2(y)) \in \mathcal{F}_{C'A_0}$. Since $E(y) = E'(y)$ and $E(\bar{\mathcal{S}}) = E'(\bar{\mathcal{S}})$ we have that $att(E'_1(\bar{\mathcal{S}}_1), E'_1(x), E'_2(\bar{\mathcal{S}}_2), E'_2(x)) \in \mathcal{F}_{C'A_0}$. $\mathcal{S} \leq \mathcal{R}$ thus $att(E'_1(\bar{\mathcal{R}}_1), E'_1(y), E'_2(\bar{\mathcal{R}}_2), E'_2(y)) \in \mathcal{F}_{C'A_0}$.
  If $y = x_k$ for some $k \in \{1, \ldots, m\}$ by hypothesis $cells(C') \subseteq fn(A_0)$, thus by definition of $E0$ we have that $cells(C') \subseteq dom(E0)$ and by construction of $\mathcal{F}_{C'A_0}$ we have that $\forall\ i \in \{1, \ldots, n\}$
  $C_i = msg((xs_1, \ldots, xs_n), xc, xm, (xs'_1, \ldots, xs'_n), xc', xm') \wedge$

$att((xs_1,\ldots,xs_n),E_{0_1}(s_i),(xs'_1,\ldots,xs'_n),E_{0_1}(s_i)) \rightarrow$

$att((xs_1,\ldots,xs_n),xs_i,(xs'_1,\ldots,xs'_n),xs'_i) \in \mathcal{C}_{C'A_0}$.

By definition $\{s_{j_1},\ldots,s_{j_m}\} \in fn(P')$ and by hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(s_{j_k},E_2(\bar{\mathcal{S}}_2),E_2(s_{j_k})) \in \mathcal{F}_{C'A_0}$. Since $E0(s_{j_k}) = E(s_{j_k})$ we have that $att(E_{0_1}(\bar{\mathcal{S}}_1),E_{0_1}(s_{j_k}),E_{0_2}(\bar{\mathcal{S}}_2),E_{0_2}(s_{j_k})) \in \mathcal{F}_{C'A_0}$. Since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_{0_1}(\bar{\mathcal{R}}_1),E_{0_1}(s_{j_k}),E_{0_2}(\bar{\mathcal{R}}_2),E_{0_2}(s_{j_k})) \in \mathcal{F}_{C'A_0}$. Since $C_{j_k} \in \mathcal{C}_{C'A_0}$ we have that
$att(E_{0_1}(\bar{\mathcal{R}}_1),E_1(\bar{\mathcal{R}}_{1_{j_k}}),E_{0_2}(\bar{\mathcal{R}}_2),E_2(\bar{\mathcal{R}}_{2_{j_k}})) \in \mathcal{F}_{C'A_0}$, $\bar{\mathcal{R}}_{j_k} = \bar{\mathcal{R}}[j_k]$ thus
$att(E_{0_1}(\bar{\mathcal{R}}_1),E_1(\bar{\mathcal{R}}_1[j_k]),E_{0_2}(\bar{\mathcal{R}}_2),E_2(\bar{\mathcal{R}}_2[j_k])) \in \mathcal{F}_{C'A_0}$. Finally, since $E'(y) = E'(\mathcal{R}[j_k])$ we have that
$att(E_{0_1}(\bar{\mathcal{R}}_1),E'_1(y),E_{0_2}(\bar{\mathcal{R}}_2),E'_2(y)) \in \mathcal{F}_{C'A_0}$

(vi) P' is in the scope of a `lock` $\ldots s_i \ldots$ if and only if Q is too. So Q,λ satisfies condition (vi) by hypothesis.

Thus $(Q,E',\mathcal{R},\lambda)$ satisfies conditions $(i)-(vi)$, and we can apply our inductive hypothesis $(E,\mathcal{R},\lambda) \vdash Q$. Thus, according to our typing system

$$\frac{\begin{array}{c}\forall \mathcal{R}\ (\mathcal{S} \leq \mathcal{R}\ \wedge\ \mathcal{R} = \mathcal{R}[j \mapsto \mathcal{S}(j) \mid\ j \in \lambda] \\ (E \cup \{x_k \mapsto (E_1(\texttt{fst}(\mathcal{R}[i_k])),E_2(\texttt{snd}(\mathcal{R}[i_k]))) \mid 1 \leq k \leq m\}, \\ \mathcal{R},\lambda) \vdash Q)\end{array}}{(E,\mathcal{S},\lambda) \vdash (\texttt{read}\ s_{i_1},\ldots,s_{i_m}\ \texttt{as}\ x_1,\ldots,x_m;Q) = P'}\ \mathcal{T}_{read}$$

- Case $P' = s_{j_1},\ldots,s_{j_m} := M_1,\ldots,M_m;Q$. Let $\mathcal{R}$ be a state such that $\mathcal{S} \leq \mathcal{R}$ and $\mathcal{R} = \mathcal{R}[k \mapsto \mathcal{S}(k) | k \in \lambda]$. Let $\mathcal{R}' = \mathcal{R}[j_k \mapsto M_k \mid 1 \leq k \leq m]$. We first show that $\mathcal{R} \leq \mathcal{R}'$. By hypothesis $cells(C') \subseteq fn(A_0)$. Thus by construction of $E0$, $cells(C') \subseteq dom(E0)$, and by construction of $\mathcal{C}_{A_0}$ for all $i \in \{1,\ldots,n\}$
$\mathcal{C}_{i_1} = msg((xs_1,\ldots,xs_n),xc,xm,(xs'_1,\ldots,xs'_n),xc',xm') \wedge$
$att((xs_1,\ldots,xs_n),E0_1(s_i),(xs'_1,\ldots,xs'_n),E0_1(s'_i)) \wedge$
$att((xs_1,\ldots,xs_n),ys_i,(xs'_1,\ldots,xs'_n),ys'_i) \rightarrow$
$msg((xs_1,\ldots,ys_i,\ldots,xs_n),xc,xm,(xs'_1,\ldots,ys'_i,\ldots,xs'_n),xc',xm') \in \mathcal{C}_{A_0}$

$\mathcal{C}_{i_2} = att((xs_1, \ldots, xs_n), xm, (xs'_1, \ldots, xs'_n), xm') \wedge$
$att((xs_1, \ldots, xs_n), E0_1(s_i)(xs'_1, \ldots, xs'_n), E0_2(s'_i)) \wedge$
$att((xs_1, \ldots, xs_n), ys_i, (xs'_1, \ldots, xs'_n), ys'_i) \rightarrow$
$att((xs_1, \ldots, ys_i, \ldots, xs_n), xm, (xs'_1, \ldots, ys'_i, \ldots, xs'_n), xm') \in \mathcal{C}_{A_0}$.
By definition $s_{j_1}, \ldots, s_{j_m} \in fn(Q)$ and by hypothesis
$\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{S}}_1), E_1(s_{j_k}), E_2(\bar{\mathcal{S}}_2), E_2(s_{j_k})) \in \mathcal{F}_{C'A_0}$ since
$\mathcal{S} \leq \mathcal{R}$ we have $\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{R}}_1), E_1(s_{j_k}), E_2(\bar{\mathcal{R}}_2), E_2(s_{j_k})) \in \mathcal{F}_{C'A_0}$.
$E0(s_{j_k}) = E(s_{j_k})$ then
$\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{R}}_1), E0_1(s_{j_k}), E_2(\bar{\mathcal{R}}_2), E0_2(s_{j_k})) \in \mathcal{F}_{C'A_0}$.
By hypothesis $\forall\, u \in fn(M_k) \cup fv(M_k)$
$\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{S}}_1), E_1(u), E_2(\bar{\mathcal{S}}_2), E_2(u)) \in \mathcal{F}_{C'A_0}$ then by Lemma C.20
$\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{S}}_1), E_1(\mathtt{fst}(M_k)), E_2(\bar{\mathcal{S}}_2), E_2(\mathtt{snd}(M_k))) \in \mathcal{F}_{C'A_0}$, since $\mathcal{S} \leq \mathcal{R}$ we have that
$\bigwedge_{1 \leq k \leq m} att(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(M_k)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(M_k))) \in \mathcal{F}_{C'A_0}$. Finally, we can infer that
$\forall\, K, L\ msg(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(K)), E_1(\mathtt{fst}(L)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(K)),$
$E_2(\mathtt{snd}(L))) \in \mathcal{F}_{C'A_0} \Rightarrow$
$msg(E_1(\bar{\mathcal{R}}'_1), E_1(\mathtt{fst}(K)), E_1(\mathtt{fst}(L)), E_2(\bar{\mathcal{R}}'_2), E_2(\mathtt{snd}(K)), E_2(\mathtt{snd}(L))) \in$
$\mathcal{F}_{C'A_0}$ and
$\forall\, K\ att(E_1(\bar{\mathcal{R}}_1), E_1(\mathtt{fst}(K)), E_2(\bar{\mathcal{R}}_2), E_2(\mathtt{snd}(K))) \in \mathcal{F}_{C'A_0} \Rightarrow$
$att(E_1(\bar{\mathcal{R}}'_1), E_1(\mathtt{fst}(K)), E_2(\bar{\mathcal{R}}'_2), E_2(\mathtt{snd}(K)))$ hence by definition $\bar{\mathcal{R}} \leq \bar{\mathcal{R}}'$

(i) By hypothesis, $E0 \subseteq E$

(ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R} \leq \mathcal{R}'$.

(iii) By definition of P' we have that $bn(Q) \cup bv(Q) = bn(P') \cup bv(P')$, then we have that $(bn(Q) \cup bv(Q)) \cap dom(E) = (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

(iv) Let $a \in fn(Q)$ By definition $fn(Q) \subseteq fn(P')$ and $a \in fn(P')$ and by hypothesis $\forall\, a \in fn(Q)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that

$att(E_1(\bar{\mathcal{R}}_1), E_1(a), E_2(\bar{\mathcal{R}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ and since $\mathcal{R} \leq \mathcal{R}'$ we have that $att(E_1(\bar{\mathcal{R}}'_1), E_1(a), E_2(\bar{\mathcal{R}}'_2), E_2(a)) \in \mathcal{F}_{C'A_0}$.

(v) Let $x \in fv(Q)$ By definition $fv(Q) \subseteq fv(P')$ hence $x \in fv(P')$ and by hypothesis $\forall\, x \in fv(Q)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$ since $\mathcal{S} \leq \mathcal{R}$ we have that $att(E_1(\bar{\mathcal{R}}_1), E_1(x), E_2(\bar{\mathcal{R}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$ and since $\mathcal{R} \leq \mathcal{R}'$ we have that $att(E_1(\bar{\mathcal{R}}'_1), E_1(x), E_2(\bar{\mathcal{R}}'_2), E_2(x)) \in \mathcal{F}_{C'A_0}$.

(vi) P' is in the scope of a `lock` $\ldots s_i \ldots$ if and only if Q is too. So Q, λ satisfies condition (vi) by hypothesis.

Thus $(Q, E, \mathcal{R}', \lambda)$ satisfies conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E, \mathcal{R}', \lambda) \vdash Q$. Thus, according to our typing system

$$\frac{\begin{array}{c}\forall \mathcal{R}\ (\mathcal{S} \leq \mathcal{R}, \mathcal{R} = \mathcal{R}[k \mapsto \mathcal{S}(k) \mid k \in \lambda] \wedge \\ \mathcal{R} \leq \mathcal{R}' = \mathcal{R}[j_k \mapsto M_k], 1 \leq k \leq m), \\ (E, \mathcal{R}[j_k \mapsto M_k], \lambda) \vdash Q\end{array}}{(E, \mathcal{S}, \lambda) \vdash (s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; Q) = P'}\ \mathcal{T}_{assign}$$

- Case $P' =$ `let` $x = D$ `in` Q_1 `else` Q_2. Let $E' = E \cup \{x \mapsto (p_1, p_2)\}$.

  (i) By hypothesis, $E0 \subseteq E \subseteq E'$

  (ii) By hypothesis, $\mathcal{S}_0 \leq \mathcal{S} \leq \mathcal{R}$.

  (iii) By definition of P' we have that $bn(Q_1) \cup bv(Q_1) \subseteq bn(P') \cup bv(P')$ and $bn(Q_2) \cup bv(Q_2) \subseteq bn(P') \cup bv(P')$, then we have that $(bn(Q_1) \cup bv(Q_1)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$ and $(bn(Q_2) \cup bv(Q_2)) \cap dom(E) \subseteq (bn(P') \cup bv(P')) \cap dom(E) = \emptyset$

  (iv) By definition $fn(Q_1) \subseteq fn(P')$ and by hypothesis $\forall\, a \in fn(Q_1)$, $att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$. By definition $fn(Q_2) \subseteq fn(P')$ and by hypothesis $\forall\, a \in fn(Q_2), att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$.

(v) By definition $fv(Q_1) \subseteq fv(P') \cup \{x\}$. Let $y \in fv(Q_1)$ then either $y \in fv(P')$ or $y = x$.
Case $y \in fv(P')$: by hypothesis $att(E_1(\bar{\mathcal{S}}_1), E_1(y), E_2(\bar{\mathcal{S}}_2), E_2(y)) \in \mathcal{F}_{C'A_0}$ since $E(y) = E'(y)$ and $E(\mathcal{S}) = E'(\mathcal{S})$ we have $att(E'_1(\bar{\mathcal{S}}_1), E'_1(y), E'_2(\bar{\mathcal{S}}_2), E'_2(y)) \in \mathcal{F}_{C'A_0}$ and since $\mathcal{S} \leq \mathcal{R}$ we have that
$att(E'_1(\bar{\mathcal{R}}_1), E'_1(y), E'_2(\bar{\mathcal{R}}_2), E'_2(y)) \in \mathcal{F}_{C'A_0}$
Case $y = x$: let p_1, p_2 such that $E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and $E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2$ since by hypothesis $\forall\, u \in fn(D) \cup fv(D)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(u), E_2(\bar{\mathcal{S}}_2), E_2(u)) \in \mathcal{F}_{C'A_0}$ and $fn(p_1) \cup fv(p_1) \subseteq fn(D) \cup fv(D), fn(p_2) \cup fv(p_2) \subseteq fn(D) \cup fv(D)$ by Lemma C.20 we have that $att(E_1(\bar{\mathcal{S}}_1), p_1, E_2(\bar{\mathcal{S}}_2), p_2) \in \mathcal{F}_{C'A_0}$. Since $\mathcal{S} \leq \mathcal{R}$ we have $att(E_1(\bar{\mathcal{R}}_1), p_1, E_2(\bar{\mathcal{R}}_2), p_2) \in \mathcal{F}_{C'A_0}$ and finally since $E'(y) = (p_1, p_2)$ we have
$att(E'_1(\bar{\mathcal{R}}_1), E'_1(y), E_2(\bar{\mathcal{R}}_2), E'_2(y)) \in \mathcal{F}_{C'A_0}$
By definition $fv(Q_2) \subseteq fv(Q) \Rightarrow \forall\, x \in fv(Q_2)$,
$att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$

(vi) P' is in the scope of a `lock` $\ldots s_i \ldots$ if and only if Q_1 and Q_2 are too. So Q_1, λ and Q_2, λ satisfy condition (vi) by hypothesis.

Thus $(Q_1, E', \mathcal{R}, \lambda)$ and $(Q_2, E, \mathcal{S}, \lambda)$ satisfy conditions $(i) - (vi)$, and we can apply our inductive hypothesis $(E', \mathcal{R}, \lambda) \vdash Q_1$, $(E, \mathcal{S}, \lambda) \vdash Q_2$.
In order to show the desired property, we have to show that if for all $a \in fn(D)$, $att(E_1(\bar{\mathcal{S}}_1), E_1(a), E_2(\bar{\mathcal{S}}_2), E_2(a)) \in \mathcal{F}_{C'A_0}$ and for all $x \in fv(D)$, $att(E_1(\bar{\mathcal{S}}_1), E_1(x), E_2(\bar{\mathcal{S}}_2), E_2(x)) \in \mathcal{F}_{C'A_0}$, then we have:

(a) if $E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and $\nexists p_2, E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2$, then $bad \in \mathcal{F}_{C'A_0}$; symmetrically, if $E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2$ and $\nexists p_1, E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1$, then $bad \in \mathcal{F}_{C'A_0}$.

The proof is by induction on D.

– **Case** $D = \mathtt{diff}[a, a]$: we have $E_1(\mathtt{fst}(D)) = E_1(a) \Downarrow_{\Sigma'} E_1(a)$ and $E_2(\mathtt{snd}(D)) = E_2(a) \Downarrow_{\Sigma'} E_2(a)$, so Property (a) holds.

– **Case** $D = x$: This case is similar to that for $D = \mathtt{diff}[a, a]$.

– **Case** $D = \mathtt{eval}\ h(D_1, \ldots, D_n)$: we prove the first part of Property (a). The second part of Property (a) follows by symmetry. Since $E_1(D) \Downarrow_{\Sigma'} p_1$, there exist $h(N_1, \ldots, N_n) \to N \in def_{\Sigma'}(h), p_1, p_{1,1}, \ldots, p_{1,n}$, and σ such that $E_1(\mathtt{fst}(D_i)) \Downarrow_{\Sigma'} p_{1,i}$ for all $i \in \{1, \ldots, n\}, p_1 = \sigma N$, and $p_{1,i} = \sigma N_i$ for all $i \in \{1, \ldots, n\}$. Since there exists no p_2 such that $E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma} p_2$, either for some $i \in \{1, \ldots, n\}$ there exists no $p_{2,i}$ such that $E_2(\mathtt{snd}(D_i)) \Downarrow_{\Sigma} p_{2,i}$ (and $bad \in \mathcal{F}_{C'A_0}$ by induction hypothesis), or for all $i \in \{1, \ldots, n\}$ there exists $p_{2,i}$ such that $E_2(\mathtt{snd}(D_i)) \Downarrow_{\Sigma} p_{2,i}$, and there exist no $h(N'_1, \ldots, N'_n) \to N' \in def_{\Sigma}(h)$ and σ such that for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash p_{2,i} = \sigma N'_i$. Hence, h must be a destructor.

By Property S2, there exists an environment E' such that $\Sigma \vdash E'(a) = E(a)$ for all $a \in fn(D)$, $\Sigma \vdash E'(x) = E(x)$ for all $x \in fv(D)$, and $nf_{S,\Sigma}(E')$. By Lemma C.16, $att(E'_1(\mathcal{S}_1), E'_1(a), E'_2(\mathcal{S}_2), E'_2(a)) \in \mathcal{F}_{C'A_0}$ for all $a \in fn(D)$ and $att(E'_1(\mathcal{S}_1), E'_1(x), E'_2(\mathcal{S}_2), E'_2(x)) \in \mathcal{F}_{C'A_0}$ for all $x \in fv(D)$. We have $nf_{S,\Sigma}(E'(D_i))$ and $\Sigma \vdash E'_2(\mathtt{snd}(D_i)) = E_2(\mathtt{snd}(D_i))$.

By Property S2, there exist $p'_{2,1}, \ldots, p'_{2,n}$ such that $\Sigma p'_{2,i} = p_{2,i}$ for all $i \in \{1, \ldots, n\}$ and $nf_{S,\Sigma}(E', p'_{2,1}, \ldots, p'_{2,n})$. By a variant of LemmaC.4 for patterns instead of terms, $E'_2(\mathtt{snd}(D_i)) \Downarrow_{\Sigma'} p'_{2,i}$ for all $i \in \{1, \ldots, n\}$. By a variant of Lemma C.7 for patterns instead of terms, $E'_1(\mathtt{fst}(D_i)) \Downarrow_{\Sigma} p'_{1,i}$ for some $p'_{1,i}$ such that $\Sigma \vdash p'_{1,i} = p_{1,i}$. By Property S2, there exist $p''_{1,1}, \ldots, p''_{1,n}, p''_1$ such that $\Sigma \vdash p''_{1,i} = p_{1,i}$ for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash p''_1 = p_1$, and $nf_{S,\Sigma}(E', p'_{2,1}, \ldots, p'_{2,n}, p''_{1,1}, \ldots, p''_{1,n}, p''_1)$. By a variant of Lemma C.4 for patterns instead of terms, $E_1(\mathtt{fst}(D_i)) \Downarrow_{\Sigma'} p''_{1,i}$ for all $i \in \{1, \ldots, n\}$. By Property (v), we obtain $att(E'_1(\mathcal{S}_1), p''_{1,i}, E'_2(\mathcal{S}_2), p'_{2,i}) \in \mathcal{F}_{C'A_0}$ for all $i \in \{1, \ldots, n\}$. Since $\Sigma \vdash p'_{2,i} = p_{2,i}$, there exist no σ and $h(N'_1, \ldots, N'_n) \to N' \in def_{\Sigma}(h)$ such that for all $i \in \{1, \ldots, n\}$,

$\Sigma \vdash p'_{2,i} = \sigma N'_i$. By Lemma C.5, there exist no σ and $h(N'_1, \ldots, N'_n) \to N' \in def_{\Sigma'}(h)$ such that for all $i \in \{1, \ldots, n\}$, $\Sigma \vdash p'_{2,i} = \sigma N'_i$, that is, we have

$$\bigwedge_{h(N'_1,\ldots,N'_n)\to N' \in def_{\Sigma'}(h)} nounif((p'_{2,1}, \ldots, p'_{2,n}), GVar(N'_1, \ldots, N'_n))$$

Since $\Sigma \vdash p''_{1,i} = p_{1,i}$, $\Sigma \vdash p''_1 = p_1$, and $nf_{S,\Sigma}(p''_{1,1}, \ldots, p''_{1,n}, p''_1)$, by Lemma C.5 and a variant of Lemma C.3 for patterns instead of terms, there exist $h(N_1, \ldots, N_n) \to N$ in $def_{\Sigma'}(h)$ and σ such that $p''_{1,i} = \sigma N_i$ for all $i \in \{1, \ldots, n\}$ and $p_1 = \sigma N$. Hence, by Clause (Rt), $bad \in \mathcal{F}_{C'A_0}$.

$$\frac{\begin{array}{c}(\forall\, p_1, p_2\ E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \\ (E[x \mapsto (p1, p2)], \mathcal{S}, \lambda) \vdash Q_1) \\ (\text{if } \nexists\, p_1,\ E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ (E, \mathcal{S}, \lambda) \vdash Q_2) \\ (\text{if } \exists\, p_1,\ E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ bad \in \mathcal{F}) \\ (\text{if } \nexists\, p_1,\ E_1(\mathtt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \exists\, p_2,\ E_2(\mathtt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ bad \in \mathcal{F})\end{array}}{(E, \mathcal{S}, \lambda) \vdash (\mathtt{let}\ x = D\ \mathtt{in}\ Q_1\ \mathtt{else}\ Q_2) = P'}\ \mathcal{T}_{let}$$

To conclude the proof we need to show that $C', E0, S0$, and $\emptyset$ satisfy conditions (i)- (vi).

(i) By definition $E0 \subseteq E0$.

(ii) By definition $S0 \leq S0$.

(iii) By hypotheses, $dom(E0) = fn(A_0) \cup fn(C') \cup cells(A_0)$ and $(bn(C') \cup bv(C')) \cap (fn(A_0) \cup fn(P) \cup cells(P)) = \emptyset$, thus $(bn(C') \cup bv(C')) \cap dom(E0) = \emptyset$.

(iv) By construction, we have that $\forall a \in fn(C')\ E0(a) = (a[x], a[x])$, and $att(E0_1(\mathcal{S}0_1), a[x], E0_2(\mathcal{S}0_2), a[x]) \in \mathcal{C}_{A_0}$. Thus $\forall a \in fn(A)$, $att(E0_1(\mathcal{S}0_1), E0_1(a), E0_2(\mathcal{S}0_2), E0_2(a)) \in \mathcal{F}_{C'A_0}$.

(v) C' is an Init-adversary, so it is a closed process, and $fv(C') = \emptyset$.

(vi) C' is by definition under no lock in C', thus by definition $C', \emptyset$ satisfy condition (vi)

We can then apply the preliminary result we just established to conclude that $(E0, S0, \emptyset) \vdash C'$. □

C.7.4 Typability of the Protocol (A_0)

Let $E0 = a \mapsto (a[], a[]) | a \in fn(C'[A'_0])$. Let $\mathcal{S}0 = \{s_1 \mapsto M_1, \ldots, s_n \mapsto M_n\}$

Lemma C.24. Typability of the protocol: *We show that* $(E0, \mathcal{S}0, \emptyset) \vdash P$*, similarly to [ARR11].*

Proof. Let Q be a subprocess of P and σ, ρ, H, φ, and λ. We first prove by induction on the size of Q, that if

(i) ρ binds all the free names and variables of Q, H, and φ;

(ii) $(bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$;

(iii) σ is a closed substitution;

(iv) $i \in \lambda$ if and only if Q is in the scope of `lock` $\ldots s_i \ldots$ in P;

(v) $\mathcal{C}_{A_0} \supseteq \|Q\|\rho\omega_1\omega_2 H\varphi_1\varphi_2\lambda$;

(vi) $\forall\, C \in H$, σC can be derived from $\mathcal{C}_{A_0}$.

then $(\rho\sigma, \varphi\sigma, \lambda) \vdash Q$.

Base case ($|Q| = 0$). In this case $Q = 0$, and thus according to the rule $\mathcal{T}_{nil}$ of our type system we have that

$$\frac{}{(\rho\sigma, \varphi\sigma, \lambda) \vdash 0 (= Q)}\ \mathcal{T}_{nil}$$

Inductive case ($|Q| > 0$). We proceed by case analysis on the structure of Q.

- **Case** $Q = Q_1 \mid Q_2$. In this case, $\lambda = \emptyset$ because no parallel composition can occur under a lock. We will show that $(Q_1, \sigma, \rho, H, \varphi, \lambda)$ and $(Q_2, \sigma, \rho, H, \varphi, \lambda)$ satisfy conditions (i)-(vi)

  (i) By definition, $fv(Q_1) \cup fv(Q_2) = fv(Q)$ and $fn(Q_1) \cup fn(Q_2) = fn(Q)$. Thus if ρ binds the free names and variables of Q, it also binds the free names and variables of Q_1 and Q_2.

  (ii) By definition $bn(Q_1) \cup bv(Q_1) \subseteq bn(Q) \cup bv(Q)$ and $bn(Q_2) \cup bv(Q_2) \subseteq bn(Q) \cup bv(Q)$ then by hypothesis $(bn(Q_1) \cup bv(Q_1)) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ and $(bn(Q_2) \cup bv(Q_2)) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$

  (iii) By hypothesis σ is a closed substitution.

  (iv) By definition Q is under a `lock` $\ldots s_i \ldots$ in P if and only if Q_1 and Q_2 are also in the scope of a `lock` $\ldots s_i \ldots$ in P, thus condition (iv) is satisfied by hypothesis.

  (v) By hypothesis $\mathcal{C}_{A_0} \supseteq \|Q_1 \mid Q_2\|\rho H\varphi_1\varphi_2\lambda \stackrel{def}{=} \|Q_1\|\rho H\varphi_1\varphi_2\lambda \cup \|Q_2\|\rho H \varphi_1\varphi_2\lambda$

(vi) Let $C \in H$. By hypothesis, we know that σC is derivable from $\mathcal{C}_{A_0}$.

We can thus apply our induction hypothesis to infer that $(\rho\sigma, \varphi\sigma, \lambda) \vdash Q_1$ and $(\rho\sigma, \varphi\sigma, \lambda) \vdash Q_2$. Hence according to our type system

$$\frac{(\rho\sigma, \varphi\sigma, \lambda) \vdash Q_1 \qquad (\rho\sigma, \varphi\sigma, \lambda) \vdash Q_2}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (Q_1 \mid Q_2) = Q}\ \mathcal{T}_{par}$$

- **Case** $Q = !^i Q'$. In this case, $\lambda = \emptyset$ because no replication can occur under a lock. We will show that $(Q', \sigma, \rho[i \mapsto (l, l)], H, \varphi, \lambda)$ satisfy conditions (i)-(vi)

  (i) By definition, $fv(Q') = fv(Q)$ and $fn(Q') = fn(Q)$. Thus if ρ binds the free names and variables of Q, it also binds the free names and variables of Q'.

  (ii) By definition we have that $bn(Q') \cup bv(Q') = bn(Q) \cup bv(Q)$ and by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) = (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$

  (iii) By hypothesis σ is a closed substitution.

  (iv) By definition Q is under a $\texttt{lock}\ \ldots s_i \ldots$ in P if and only if Q' is also under a $\texttt{lock}\ \ldots s_i \ldots$ in P, thus condition (iv) is satisfied by hypothesis.

  (v) By hypothesis $\mathcal{C}_{A_0} \supseteq \|!^i Q'\| \rho H \varphi_1 \varphi_2 \lambda = \|Q'\| \rho[i \mapsto (l, l)] H \varphi_1 \varphi_2 \lambda$

  (vi) Let $C \in H$. By hypothesis, we know that σC is derivable from $\mathcal{C}_{A_0}$.

  We can thus apply the induction hypothesis to infer that $(\rho[i \mapsto (l, l)]\sigma, \varphi\sigma, \lambda) \vdash Q'$ and hence according to our type system

$$\frac{\forall\, l \quad (\rho[i \mapsto (l, l)]\sigma, \varphi\sigma, \lambda) \vdash Q'}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (!^i Q') = Q}\ \mathcal{T}_{repl}$$

- **Case** $Q = \texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'$. Note that Q being a sub-process of P implies that $a \in bn(P)$. Let $\rho' = \rho\ \cup \{a \mapsto (a[\rho_1(\texttt{fst}(M_1)), \ldots, \rho_1(\texttt{fst}(M_n))], a[\rho_2(\texttt{snd}(M_1)), \ldots, \rho_2(\texttt{snd}(M_n))])\}$. We first show that $(Q', \sigma\rho', H, \varphi, \lambda)$ satisfy conditions (i)-(vi)

(i) By definition we have that $fv(Q') \cup fn(Q') \cup fn(M_1) \cup \cdots \cup fn(M_n) \cup fv(M_1) \cup \cdots \cup fv(M_n) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) = fv(Q) \cup fn(Q) \cup \{a\} \cup fn(M_1) \cup \cdots \cup fn(M_n) \cup fv(M_1) \cup \cdots \cup fv(M_n) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi)$. By hypothesis $fv(Q') \cup fn(Q') \cup \cup fn(M_1) \cup \cdots \cup fn(M_n) \cup fv(M_1) \cup \cdots \cup fv(M_n) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) \subseteq dom(\rho) \cup \{a\}$. Since $dom(\rho') = dom(\rho) \cup \{a\}$ we have that $fv(Q') \cup fn(Q') \cup \cup fn(M_1) \cup \cdots \cup fn(M_n) \cup fv(M_1) \cup \cdots \cup fv(M_n) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) \subseteq dom(\rho')$

(ii) By definition $(bn(Q') \cup bv(Q')) \subset (bn(Q) \cup bv(Q))$ and by hypothesis $(bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$. Hence, $(bn(Q') \cup bv(Q')) \cap dom(\rho) \subset (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$, since $a \notin bn(Q')$ we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho) = \emptyset$

(iii) By hypothesis, σ is a closed substitution.

(iv) By definition Q is under a $\texttt{lock} \ldots s_i \ldots$ in P if and only if Q' is under a $lock \ldots s_i \ldots$, so condition (iv) is satisfied by hypothesis.

(v) $\mathcal{C}_{A_0} \supseteq \|\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q'\|\rho H \varphi_1 \varphi_2 \lambda \stackrel{def}{=}$
$\|Q'\|(\rho \cup \{a \mapsto (a[\rho_1(\texttt{fst}(M_1)), \ldots, \rho_1(\texttt{fst}(M_n))],$
$a[\rho_2(\texttt{snd}(M_1)), \ldots, \rho_2(\texttt{snd}(M_n))])\})H\varphi_1\varphi_2\lambda$ since
$\rho' = \rho \cup \{a \mapsto (a[\rho_1(\texttt{fst}(M_1)), \ldots, \rho_1(\texttt{fst}(M_n))],$
$a[\rho_2(\texttt{snd}(M_1)), \ldots, \rho_2(\texttt{snd}(M_n))])\}$ we have that $\mathcal{C}_{A_0} \supseteq \|Q'\|\rho' H \varphi_1 \varphi_2 \lambda$

(vi) Let $C \in H$. By hypothesis $sigmaC$ is derivable from $\mathcal{C}_{A_0}$.

We can thus apply the induction hypothesis to infer that $(\rho'\sigma, \varphi\sigma, \lambda) \vdash Q'$ and since $a \notin dom(\sigma)$ we have that $(\rho\sigma \cup \{a \mapsto (a[\rho_1(\texttt{fst}(M_1)), \ldots, \rho_1(\texttt{fst}(M_n))], a[\rho_2(\texttt{snd}(M_1)), \ldots, \rho_2(\texttt{snd}(M_n))])\}, \varphi\sigma, \lambda) \vdash Q'$ and then according to our typing system

$$\frac{(\rho\sigma \cup \{a \mapsto (a[\rho_1(\texttt{fst}(M_1)), \ldots, \rho_1(\texttt{fst}(M_n))], a[\rho_2(\texttt{snd}(M_1)), \ldots, \rho_2(\texttt{snd}(M_n))])\}, \varphi\sigma, \lambda) \vdash Q'}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q') = Q}\ \mathcal{T}_{new}$$

- **Case** $Q = \texttt{out}(M, N)); Q'$.

  (i) By definition $fn(Q') \subseteq fn(Q)$ and $fv(Q') \subseteq fv(Q)$ and by hypothesis ρ binds the free names and variables of Q, hence it also binds the free names and variables of Q'.

  (ii) By definition $bn(Q') \cup bv(Q') \subseteq bn(Q) \cup bv(Q)$ and by hypothesis$(bn(Q') \cup bv(Q')) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$

  (iii) By hypothesis, σ is a closed substitution.

  (iv) By definition Q is under a $\texttt{lock}\ \ldots s_i \ldots$ in P if and only if Q' is under a $\texttt{lock}\ \ldots s_i \ldots$ in P. Thus Q', λ satisfy condition (iv) because by hypothesis Q, λ satisfy it.

  (v) $\mathcal{C}_{A_0} \supseteq \|\texttt{out}(M, N); Q' \| \rho H \varphi_1 \varphi_2 \lambda \supseteq \|Q' \| \rho H \varphi_1 \varphi_2 \lambda$

  (vi) Let $C \in H$. By hypothesis, we know that σC is derivable from $\mathcal{C}_{A_0}$.

  We can thus apply the induction hypothesis to infer that $(\rho\sigma, \varphi\sigma, \lambda) \vdash Q'$. Moreover, by definition of our translation we have that $\|Q\| \rho H \varphi_1 \varphi_2 \lambda = \{H \rightarrow msg(\rho_1(\varphi_1), \rho_1(\texttt{fst}(M)), \rho_1(\texttt{fst}(N)), \rho_2(\varphi_2), \rho_2(\texttt{snd}(M)), \rho_2(\texttt{snd}(N)))\} \cup \|Q'\| \rho H \varphi_1 \varphi_2 \lambda$ with H and σ satisfying condition (vi), *i.e.* $H\sigma$ is derivable from $\mathcal{C}_{A_0}$. So by resolution we can derive $msg(\rho_1(\varphi_1)\sigma, \rho_1(\texttt{fst}(M))\sigma, \rho_1(\texttt{fst}(N))\sigma, \rho_2(\varphi_2)\sigma, \rho_2(\texttt{snd}(M))\sigma, \rho_2(\texttt{snd}(N))\sigma)$ and since $fn(Q) \subseteq dom(\rho)$ we have that $msg(\rho_1(\varphi_1)\sigma, \rho_1(\texttt{fst}(M))\sigma, \rho_1(\texttt{fst}(N))\sigma, \rho_2(\varphi_2)\sigma, \rho_2(\texttt{snd}(M))\sigma, \rho_2(\texttt{snd}(N))\sigma) = msg(\rho_1\sigma(\varphi_1), \rho_1\sigma(\texttt{fst}(M)), \rho_1\sigma(\texttt{fst}(N)), \rho_2\sigma(\varphi_2), \rho_2\sigma(\texttt{snd}(M)), \rho_2\sigma(\texttt{snd}(N)))$. Finally, according to our typing system

$$\frac{\begin{array}{c} msg(\rho_1\sigma(\varphi_1), \rho_1\sigma(\texttt{fst}(M)), \rho_1\sigma(\texttt{fst}(N)), \\ \rho_2\sigma(\varphi_2), \rho_2\sigma(\texttt{snd}(M)), \rho_2\sigma(\texttt{snd}(N))) \in \mathcal{F}_{C'A_0} \\ (\rho\sigma, \varphi\sigma, \lambda) \vdash Q' \end{array}}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\texttt{out}(M, N); Q') = Q} \ \mathcal{T}_{out}$$

Case $Q = \mathtt{in}(M, x); Q'$. Let ψ be a state such that $\varphi\sigma \leq \psi$ and $\psi = \psi[j \mapsto \varphi\sigma(j) | j \in \lambda]$. Let p_1, p_2 be a pattern such that $msg(\rho_1(\psi_1)), (\rho_1\sigma)(K), p_1, \rho_2(\psi_2)), (\rho_2\sigma)(K), p_2) \in \mathcal{C}_{A_0}$. Let $\rho' = \rho \cup \{x \mapsto (x', x'')\} \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}, \sigma' = \sigma \cup \{x' \mapsto p_1, x'' \mapsto p_2\} \cup \{vs'_j \mapsto \rho_1(\psi_1(j)), vs''_j \mapsto \rho_2(\psi_2(j)) | j \notin \lambda\}, \varphi' = \varphi[j \mapsto vs_j | j \notin \lambda], H' = H \wedge msg(\rho_1(\varphi_1), \rho_1(\mathtt{fst}(M)), x', \rho_2(\varphi_2), \rho_2(\mathtt{snd}(M)), x'')$, for $vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ fresh. We show that $(Q', \sigma', \rho', H, \varphi', \lambda)$ satisfy conditions (i)-(vi).

(i) By definition $fv(Q') \cup fn(Q) \cup fv(H') \cup fn(H') \cup fv(\varphi') \cup fn(\varphi') \subseteq fv(Q) \cup \{x\} \cup fn(Q) \cup fv(H) \cup fn(H) \cup fv(\varphi) \cup fn(\varphi) \cup \{vs_j | j \in \lambda\}$ then by hypothesis $fv(Q') \cup fn(Q') \cup fv(H') \cup fn(H') \cup fv(\varphi') \cup fn(\varphi') \subseteq dom(\rho) \cup \{x\} \cup \{vs_j | j \in \lambda\}$ since $dom(\rho) \cup \{x\} \cup \{vs_j | j \in \lambda\} \subseteq dom(\rho')$ we have that $fv(Q') \cup fn(Q') \cup fv(H') \cup fn(H') \cup fv(\varphi') \cup fn(\varphi') \subseteq dom(\rho')$, hence ρ' binds the free names and variables of Q', H', φ'

(ii) By definition $bn(Q') \cup bv(Q') \subseteq bn(Q) \cup bv(Q)$ then by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $x \in bv(Q')$ and $vs_1, \ldots, vs_n$ are fresh, we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho') = \emptyset$.

(iii) By hypothesis $p_1, p_2, \psi(1), \ldots, \psi(n)$ are ground then $\{x \mapsto (p_1, p_2)\} \cup \{vs'_j \mapsto \rho_1(\psi_2(j)), vs''_j \mapsto \rho_1(\psi_2(j)) | j \notin \lambda\}$ is closed and since σ is closed $\sigma \cup \{x \mapsto (p_1, p_2)\} \cup \{vs'_j \mapsto \rho_1(\psi_2(j)), vs''_j \mapsto \rho_1(\psi_2(j)) | j \notin \lambda\}$ is closed, and since $\sigma' = \sigma \cup \{x \mapsto (p_1, p_2)\} \cup \{vs'_j \mapsto \rho_1(\psi_2(j)), vs''_j \mapsto \rho_1(\psi_2(j)) | j \notin \lambda\}$ we have that σ' is closed.

(iv) By definition Q' is under a $\mathtt{lock} \ldots s_i \ldots$ in P if and only if Q is under a $\mathtt{lock} \ldots s_i \ldots$, so condition (iv) is satisfied by hypothesis.

(v) $\mathcal{C}_{A_0} \supseteq \| \in Mx; Q' \| \rho H \varphi_1 \varphi_2 \lambda \stackrel{def}{=} \|Q'\|(\rho \cup \{x \to (x', x'')\} \cup \{vs_j \to (vs'_j, vs''_j) | j \in \lambda\})(H \wedge msg(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), x', \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)), x'))\varphi'\lambda = \|Q'\|\rho' H' \varphi' \lambda \cup \{input(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)))\}$

(vi) Let $msg(xs', M', N', xs'', M'', N'') \in H'$. Then either
$msg(xs', M', N', xs'', M'', N'') \in H$ or $msg(xs', M', N', xs'', M'', N'') = msg(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), x', \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)), x'')$.
If $msg(xs', M', N', xs'', M'', N'') \in H$ by hypothesis
$msg(xs'\sigma, M'\sigma, N'\sigma, xs''\sigma, M''\sigma, N''\sigma,)$ is derivable from $\mathcal{C}_{A_0}$ since $x' \notin fn(xs') \cup fn(M') \cup fn(N'), x'' \notin fn(xs'') \cup fn(M'') \cup fn(N'')$ and $vs_1, \ldots, vs_n$ are fresh, we have that
$msg(xs'\sigma', M'\sigma', N'\sigma', xs''\sigma', M''\sigma', N''\sigma')$ is derivable from $\mathcal{C}_{A_0}$.
If $msg(xs', M', N', xs'', M'', N'') = msg(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), x', \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)), x'')$ by hypothesis
$msg(\rho_1(\psi_1), \rho_1(\mathtt{fst}(M))\sigma, p_1, \rho_2(\psi_2), \rho_2(\mathtt{snd}(M))\sigma, p_2)$ is derivable from $\mathcal{C}_{A_0}$ since $x' \notin fn(\rho_1(\mathtt{fst}(M)))$,
$x'' \notin fn(\rho_2(\mathtt{snd}(M)))$ and $vs_1, \ldots, vs_n$ are fresh, we have that
$msg(\rho_1(\psi_1), \rho_1(\mathtt{fst}(M))\sigma, p_1, \rho_2(\psi_2), \rho_2(\mathtt{snd}(M))\sigma, p_2)$ is derivable from $\mathcal{C}_{A_0}$. Since $\sigma'(x') = p_1, \sigma'(x'') = p_2$ we have that
$msg(\rho_1(\psi_1), \rho_1(\mathtt{fst}(M))\sigma, x'\sigma', \rho_2(\psi_2), \rho_2(\mathtt{snd}(M))\sigma, x''\sigma')$ is derivable from $\mathcal{C}_{A_0}$.
$\psi = \varphi'\sigma'$ hence $msg(\rho_1(\varphi'_1\sigma'), \rho_1(\mathtt{fst}(M))\sigma, x'\sigma',$
$\rho_2(\varphi'_2\sigma'), \rho_2(\mathtt{snd}(M))\sigma, x''\sigma')$ is derivable from $\mathcal{C}_{A_0}$. Finally, since
$msg(xs', M', N', xs'', M'', N'') = msg(\rho_1(\varphi'_1), \rho_1(\mathtt{fst}(M)), x', \rho_2(\varphi'_2), \rho_2(\mathtt{snd}(M)), x'')$ we have that
$msg(xs'\sigma', M'\sigma', N'\sigma', xs''\sigma', M''\sigma', N''\sigma')$ is derivable from $\mathcal{C}_{A_0}$. Hence $\forall C \in H$, σC is derivable from $\mathcal{C}_{A_0}$.

We can thus apply the induction hypothesis to infer that $(\rho'\sigma', \varphi'\sigma', \lambda) \vdash Q'$ since $\psi = \varphi'\sigma'$, $(\rho'\sigma', \psi, \lambda) \vdash Q'$. $\rho'\sigma' = \rho\sigma' \cup \{x \mapsto (p_1, p_2)\}$ hence $(\rho\sigma' \cup \{x \mapsto (p_1, p_2)\}, \varphi'\sigma', \lambda) \vdash Q'$ and since $x \notin \rho$ we have $(\rho\sigma \cup \{x \mapsto$

$(p_1, p_2)\}, \varphi'\sigma', \lambda) \vdash Q'$. Thus according to our typing system

$$\frac{\begin{array}{c}\forall \psi\ (\varphi\sigma \leq \psi\ \wedge\ \psi = \psi[j \mapsto \varphi(j)\sigma \mid\ j \in \lambda] \wedge \\ \forall\ N : \rho\sigma(N) = (p_1, p_2), \\ msg(\rho_1(\bar{\psi}_1), \rho_1\sigma(\mathtt{fst}(M)), p_1, \rho_2(\bar{\psi}_2), \rho_2\sigma(\mathtt{snd}(M)), p_2) \in \mathcal{F} \\ input(\rho_1(\bar{\psi}_1), \rho_1\sigma(\mathtt{fst}(M)), \rho_2(\bar{\psi}_2), \rho_2\sigma(\mathtt{snd}(M))) \in \mathcal{F}) \\ (\rho\sigma \cup \{x \mapsto (p_1, p_2)\}, \psi, \lambda) \vdash Q'\end{array}}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\mathtt{in}(M, x); Q') = Q}\ \mathcal{T}_{in}$$

- **Case** $Q = \mathtt{lock}\ s_{j_1}, \ldots, s_{j_m}; Q'$. Let ψ such that $\varphi\sigma \leq \psi$ and $\psi = \psi[j \mapsto \varphi\sigma(j) | j \in \lambda]$. Let $\rho' = \rho \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}, \sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}, \varphi' = \varphi[j \mapsto vs_j | j \notin \lambda], \lambda' = \lambda \cup \{j_1, \ldots, j_m\}$, for some $vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ fresh. We will show that $(Q', \sigma', \rho', H, \varphi', \lambda')$ satisfy conditions (i)-(vi).

  (i) By definition $fn(Q') \cup fv(Q') \cup fn(H) \cup fv(H) \cup fn(\varphi') \cup fv(\varphi') \subseteq fn(Q) \cup fv(Q) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) \cup \{vs_j | j \notin \lambda\}$ By hypothesis $fn(Q') \cup fv(Q') \cup fn(H) \cup fv(H) \cup fn(\varphi') \cup fv(\varphi') \subseteq dom(\rho) \cup \{vs_j | j \notin \lambda\}$ since $\rho' = dom(\rho) \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}$ we have $fn(Q') \cup fv(Q') \subseteq dom(\rho')$.

  (ii) By definition $bn(Q') \cup bv(Q') = bn(Q) \cup bv(Q)$ and by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) = (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $vs_1, \ldots, vs_n \notin bv(Q')$ we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho') = \emptyset$

  (iii) by hypothesis $\psi(1), \ldots, \psi(n)$ are ground and by definition $\{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed. Since σ is closed, we have that $\sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed. Thus $\sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed.

  (iv) By definition Q is under a $\mathtt{lock}\ \ldots s_i \ldots$ if and only if either $s_i \in \{s_{j_1}, \ldots, s_{j_m}\}$ or Q is under a $\mathtt{lock}\ \ldots s_i \ldots$, so condition (iv) is satisfied by construction of λ.

(v) $\mathcal{C}_{A_0} \supseteq \|\texttt{lock}\ s_{j_1},\ldots,s_{j_m};Q'\|\rho H\varphi_1\varphi_2\lambda \stackrel{def}{=} \|Q'\|(\rho \cup \{vs_j \mapsto (vs'_j, vs''_j)|$ $j \notin \lambda\})H(\varphi_1[j \mapsto vs_j|j \notin \lambda])(\varphi_2[j \mapsto vs_j|j \notin \lambda])(\lambda \cup \{j_1,\ldots,j_m\}) \stackrel{def}{=}$ $\|Q'\|\rho' H\varphi'_1\varphi'_2\lambda'$

(vi) Let $msg(xs, M, N, xs', M', N') \in H$. By hypothesis $msg(xs\sigma, M\sigma, N\sigma,$ $xs'\sigma, M'\sigma, N'\sigma)$ is derivable from $\mathcal{C}_{A_0}$ since $vs_1,\ldots,vs_n, vs'_1,\ldots,vs'_n,$ $vs''_1,\ldots,vs''_n \notin fv(xs)\cup fv(M)\cup fv(N)\cup fv(xs')\cup fv(M')\cup fv(N')$ we have that $msg(xs\sigma', M\sigma', N\sigma', xs'\sigma', M'\sigma', N'\sigma')$ is derivable from $\mathcal{C}_{A_0}$. Hence $\forall C \in H$, σC is derivable from $\mathcal{C}_{A_0}$.

We can thus apply our induction hypothesis to infer that $(\rho'\sigma', \varphi'\sigma', \lambda') \vdash Q'$ and by definition $(\rho\sigma \cup \{vs_j \mapsto (\rho_1(\psi(j)_1),$ $\rho_2(\psi(j)_2))|j \notin \lambda\}, \psi, \lambda \cup \{j_1,\ldots,j_m\}) \vdash Q'$. Lemma C.21 $(\rho\sigma, \psi, \lambda \cup$ $\{j_1,\ldots,j_m\}) \vdash Q'\{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2)|j \notin \lambda\}$, since $vs'_1,\ldots,vs'_n, vs''_1,\ldots,vs''_n \notin fv(Q')$ we have that $(\rho\sigma, \psi, \lambda\cup\{j_1,\ldots,j_m\}) \vdash Q'$ But then according to our typing system

$$\frac{\begin{array}{c}\forall\psi\ (\varphi\sigma \leq \psi\ \wedge\ \psi = \psi[j \mapsto \varphi\sigma(j) \mid\ j \in \lambda] \\ (\rho\sigma, \psi, \lambda \cup \{i_1,\ldots,i_m\}) \vdash Q')\end{array}}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\texttt{lock}\ s_{j_1},\ldots,s_{j_m};Q') = Q}\ \mathcal{T}_{lock}$$

- **Case** $Q = \texttt{unlock}\ s_{j_1},\ldots,s_{j_m};Q'$. Let ψ such that $\varphi\sigma \leq \psi$ and $\psi = \psi[j \mapsto \varphi\sigma(j)|j \in \lambda]$. Let $\rho' = \rho \cup \{vs_j \mapsto (vs'_j, vs''_j)|j \notin \lambda\}, \sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2)|j \notin \lambda\}, \varphi' = \varphi[j \mapsto diff[vs'_j, vs''_j]|j \notin \lambda], \lambda' = \lambda \setminus \{j_1,\ldots,j_m\}$, for some $vs_1,\ldots,vs_n$ fresh. We will show that $(Q', \sigma', \rho', H, \varphi', \lambda')$ satisfy conditions (i)-(vi).

  (i) By definition $fn(Q') \cup fv(Q') \cup fn(H) \cup fv(H) \cup fn(\varphi') \cup fv(\varphi') \subseteq$ $fn(Q) \cup fv(Q) \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) \cup \{vs_j|j \notin \lambda\}$ and by hypothesis $fn(Q') \cup fv(Q') \cup fn(H) \cup fv(H) \cup fn(\varphi') \cup fv(\varphi') \subseteq$ $dom(\rho) \cup \{vs_j|j \notin \lambda\}$ since $\rho = dom(\rho) \cup \{vs_j \mapsto vs_j|j \in \lambda\}$ we have that $fn(Q') \cup fv(Q') \subseteq dom(\rho')$

(ii) By definition $bn(Q') \cup bv(Q') = bn(Q) \cup bv(Q)$ by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) = (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $vs_1, \ldots, vs_n \notin bv(Q')$ we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho') = \emptyset$

(iii) $\psi(1), \ldots, \psi(n)$ are ground and by definition $\{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed and since σ is closed $\sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed then $\sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}$ is closed

(iv) By definition Q is not under a `lock` $\ldots s_i \ldots$ if and only if either $s_i \in \{s_{j_1}, \ldots, s_{j_m}\}$ or Q is not under a `lock` $\ldots s_i \ldots$, so condition (iv) is satisfied by construction of λ

(v) $\mathcal{C}_{A_0} \subseteq \|\texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q'\| \rho H \varphi_1 \varphi_2 \lambda \stackrel{def}{=} \|Q'\|(\rho \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}) H (\varphi_1[j \mapsto vs_j | j \notin \lambda)(\varphi_2[j \mapsto vs_j | j \notin \lambda)(\lambda \setminus \{j_1, \ldots, j_m\}) \stackrel{def}{=} \|Q'\| \rho' H \varphi' \lambda'$

(vi) Let $msg(xs, M, N, xs', M', N') \in H$. By hypothesis $msg(xs\sigma, M\sigma, N\sigma, xs'\sigma, M'\sigma, N'\sigma)$ is derivable from $\mathcal{C}_{A_0}$ since $vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n \notin fv(xs) \cup fv(M) \cup fv(N) \cup fv(xs') \cup fv(M') \cup fv(N')$ we have that
$msg(xs\sigma', M\sigma', N\sigma', xs'\sigma', M'\sigma', N'\sigma')$ is derivable from $\mathcal{C}_{A_0}$.
Hence $\forall C \in H$, σC is derivable from $\mathcal{C}_{A_0}$.

We can thus apply our induction hypothesis to infer that $(\rho'\sigma', \varphi'\sigma', \lambda') \vdash Q'$ and by definition $(\rho\sigma \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}, \psi, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q'$ and by Lemma C.21 $(\rho\sigma, \psi, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q'\{vs'_j \mapsto \rho_1(\psi_1(j)), vs''_j \mapsto \rho_2(\psi_2(j)) | j \notin \lambda\}$ since $\{vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n\} \notin fv(Q')$ we have $(\rho\sigma, \psi, \lambda \setminus \{j_1, \ldots, j_m\}) \vdash Q'$. But then according to our typing system

$$\frac{\forall \psi\ (\varphi\sigma \leq \psi\ \wedge\ \psi = \psi[j \mapsto \varphi\sigma(j) \mid\ j \in \lambda] \quad (\rho\sigma, \psi, \lambda \setminus \{i_1, \ldots, i_m\}) \vdash Q')}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\texttt{unlock}\ s_{j_1}, \ldots, s_{j_m}; Q') = Q}\ \mathcal{T}_{unlock}$$

- **Case** $Q = s_{j_1}, \ldots, s_{j_m} := M_1, \ldots, M_m; Q'$. Let ψ such that $\varphi\sigma \leq \psi$ and $\psi = \psi[j \mapsto \varphi\sigma(j) | j \in \lambda]$. Let $\rho' = \rho \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}, \sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}, \varphi' = \varphi[j \mapsto vs_j | j \notin \lambda]$, and $\varphi'' = \varphi'[j_k \mapsto M_k | 1 \leq k \leq m]$, for some $vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ fresh. We will show that $(Q', \sigma', \rho', H, \varphi', \lambda)$ satisfy conditions (i)-(vi). We have that $\varphi\sigma = \psi \leq \psi[j_k \mapsto \sigma(M_k) | 1 \leq k \leq m] = \varphi''\sigma'$. By definition of our translation $\|s_{j_1}, \ldots, s_{j_m} := M_1, \ldots, M_m; Q'\|\rho H\varphi\lambda = \|Q'\|(\rho \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\} H\varphi''\lambda \cup \{H \wedge msg(\rho_1(\varphi'_1), wc, wm, \rho_2(\varphi'_2), wc', wm') \rightarrow msg(\rho_1(\varphi''_1), wc, wm, \rho_2(\varphi''_2), wc', wm')\} (= \mathcal{C}_1) \cup$
  $\{H \wedge att(\rho_1(\varphi'_1), wm, \rho_2(\varphi'_2), wm') \rightarrow att(\rho_1(\varphi'_1), wm, \rho_2(\varphi'_2), wm')\} (= \mathcal{C}_2)$
  for some wc, wm, wc', wm' fresh.

  Let $msg(\rho_1(\psi_1), M, N, \rho_2(\psi_2), M', N') \in \mathcal{F}_{A_0}$. By hypothesis $H\sigma$ is derivable from $\mathcal{C}_{A_0}$ and $msg(\rho_1(\psi_1), M, N, \rho_2(\psi_2), M', N') \in \mathcal{F}_{A_0}$ since $vs_1, \ldots, vs_n$, $vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ are fresh $H\sigma'$ is derivable from $\mathcal{C}_{A_0} \wedge msg(\rho_1(\psi_1), M, N, \rho_2(\psi_2), M', N') \in \mathcal{F}_{A_0}$ and since $\psi = \varphi'\sigma'$ we have $H\sigma'$ is derivable from $\mathcal{C}_{A_0} \wedge msg(\varphi'_1\sigma', M, N, \varphi'_2, M', N') \in \mathcal{F}_P$. $\mathcal{C}_1 \in \mathcal{C}_{A_0}$ thus $msg(\varphi''_1\sigma', M, N, \varphi''_2\sigma' M', N') \in \mathcal{F}_{A_0}$ since $\psi[j_k \mapsto \sigma(M_k) | 1 \leq k \leq m] = \varphi''\sigma'$ we have that $msg(\rho_1(\psi_1[j_k \mapsto \sigma(\mathtt{fst}(M_k)) | 1 \leq k \leq m]), M, N, \rho_2(\psi_2[j_k \mapsto \sigma(\mathtt{snd}(M_k)) | 1 \leq k \leq m]) M', N') \in \mathcal{F}_{A_0}$

  Let $att(\rho_1(\psi_1), M, \rho_2(\psi_2), M') \in \mathcal{F}_{A_0}$. By hypothesis $H\sigma$ is derivable from $\mathcal{C}_{A_0} \wedge att(\rho_1(\psi_1), M, \rho_2(\psi_2), M') \in \mathcal{F}_{A_0}$ since $vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n$, $vs''_1, \ldots, vs''_n$ are fresh $H\sigma$ is derivable from $\mathcal{C}_{A_0} \wedge att(\rho_1(\psi_1), M, \rho_2(\psi_2), M') \in \mathcal{F}_{A_0}$ and $\psi = \varphi'\sigma'$ we have that $\mathcal{C}_{A_0} \wedge att(\rho_1(\varphi'_1\sigma'), M, \rho_2(\varphi'_2\sigma'), M') \in \mathcal{F}_{A_0}$. $\mathcal{C}_2 \in \mathcal{C}_{A_0}$ hence $att(\rho_1(\varphi''_1\sigma'), M, \rho_2(\varphi''_2\sigma'), M') \in \mathcal{F}_{A_0}$ and since $\psi[j_k \mapsto \sigma(M_k) | 1 \leq k \leq m] = \varphi''\sigma'$ we have that $att(\rho_1(\psi_1[j_k \mapsto \sigma(\mathtt{fst}(M_k)]), M, \rho_2(\psi_2[j_k \mapsto \sigma(\mathtt{snd}(M_k)]), M') \in \mathcal{F}_{A_0}$.

  Thus $\varphi'\sigma' = \psi \leq \psi[j_k \mapsto M_k | 1 \leq k \leq m] = \varphi''\sigma'$

  We will show that $(Q', \sigma', \rho', H, \varphi', \lambda)$ satisfy conditions (i)-(vi).

(i) $fv(Q') \cup fn(Q) \cup fv(H) \cup fn(H) \cup fv(\varphi') \cup fn(\varphi') \subseteq fv(Q) \cup fn(Q) \cup fv(H) \cup fn(H) \cup fv(\varphi) \cup fn(\varphi) \cup \{vs_j | j \notin \lambda\}$ and by hypothesis $fv(Q') \cup fn(Q') \cup \cup fv(H) \cup fn(H) \cup fv(\varphi') \cup fn(\varphi') \subseteq dom(\rho) \cup \{vs_j | j \notin \lambda\}$ since $\rho' = \rho \cup \{vsj \mapsto (vs'_j, vs''_j | j \notin \lambda\}$ we have that $fv(Q') \cup fn(Q') \subseteq dom(\rho')$

(ii) By definition $bn(Q') \cup bv(Q') = bn(Q) \cup bv(Q)$ by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) = (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $vs_1, \ldots, vs_m$ are fresh we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho') = \emptyset$

(iii) By hypothesis $\psi(1), \ldots, \psi(n)$ are ground and by definition $\{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}$ is closed. Since σ is closed $\sigma \cup \{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}$ is closed, thus $\sigma' = \sigma \cup \{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}$ is closed

(iv) By definition Q is under a `lock` $\ldots s_i \ldots$ in P if and only if Q' is under a `lock` $\ldots s_i \ldots$ in P, so condition (iv) is satisfied by hypothesis.

(v) By hypothesis $\mathcal{C}_P \supseteq \|s_{j_1}, \ldots, s_{j_m} := M_1, \ldots, M_m; Q' \| \rho H \varphi \lambda \stackrel{def}{=} \|Q' \| \rho' H \varphi'' \lambda$

(vi) Let $msg(xs, M', N', xs', M'', N'') \in H$. By hypothesis $msg(xs\sigma, M'\sigma, N'\sigma, xs'\sigma, M''\sigma, N''\sigma)$ is derivable from $\mathcal{C}_{A_0}$, and because $vs_1, \ldots vs_n$ are fresh, i.e. not in $fn(xs) \cup fn(M') \cup fn(N') \cup fn(xs') \cup fn(M'') \cup fn(N'')$, $msg(xs\sigma', M'\sigma', N'\sigma') = msg(xs\sigma, M'\sigma, N'\sigma)$ is derivable from $\mathcal{C}_{A_0}$.

We can thus apply our inductive hypothesis to infer that $(\rho'\sigma', \varphi''\sigma', \lambda) \vdash Q'$ since $\rho'\sigma' = \rho\sigma \cup \{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}$ and $\varphi''\sigma' = \psi[j_k \mapsto diff[(\rho_1\sigma)(\texttt{fst}(M_k)), (\rho_2\sigma)(\texttt{snd}(M_k))] | 1 \leq k \leq m]$ we have $(\rho\sigma \cup \{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}, \psi[j_k \mapsto diff[(\rho_1\sigma)(\texttt{fst}(M_k)), (\rho_2\sigma)(\texttt{snd}(M_k))] | 1 \leq k \leq m], \lambda) \vdash Q'$. By Lemma C.21 we have that $(\rho\sigma, \psi[j_k \mapsto diff[(\rho_1\sigma)(\texttt{fst}(M_k)), (\rho_2\sigma)(\texttt{snd}(M_k))] | 1 \leq k \leq m], \lambda) \vdash Q'\{vs'_j \mapsto \psi_1(j), vs''_j \mapsto \psi_2(j) | j \notin \lambda\}$ and since $vs'_1, \ldots vs'_n, vs''_1, \ldots vs''_n \notin fv(Q') \subseteq dom(\rho)$ we have $(\rho\sigma, \psi[j_k \mapsto diff[(\rho_1\sigma)(\texttt{fst}(M_k)), (\rho_2\sigma)(\texttt{snd}(M_k))] | 1 \leq k \leq m], \lambda) \vdash Q'$.

But then according to our typing system

$$\frac{\begin{array}{c}\forall\psi\ (\rho\sigma \leq \psi, \psi = \psi[k \mapsto \rho\sigma(k) \mid k \in \lambda] \wedge \\ \psi \leq \psi' = \psi[j_k \mapsto M_k], 1 \leq k \leq m), \\ (\rho\sigma, \psi[j_k \mapsto M_k], \lambda) \vdash Q'\end{array}}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; Q') = Q}\ \mathcal{T}_{assign}$$

- **Case** $Q = \texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ x_1, \ldots, x_m; Q'$. Let ψ be a state such that $\varphi\sigma \leq \psi$ and $\psi = \psi[j \mapsto \varphi\sigma(j) | j \in \lambda]$. Let $\rho' = \rho \cup \{vc \mapsto (vc', vc''), vm \mapsto (vm', vm'')\} \cup \{xk \mapsto (\rho_1(\varphi_1[j_k]), \rho_2(\varphi_2[j_k]))\} \cup \{vs_j \mapsto (vs'_j, vs''_j) | j \notin \lambda\}, \sigma' = \sigma \cup \{vs'_j \mapsto \rho_1(\psi(j)_1), vs''_j \mapsto \rho_2(\psi(j)_2) | j \notin \lambda\}, \varphi' = \varphi[j \mapsto vs_j | j \notin \lambda]$, and $H' = H \wedge msg(\rho_1(\varphi_1), vc', vm', \rho_2(\varphi_2), vc'', vm'')$ for some $vc, vm, vc', vm', vc'', vm'', vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ fresh. We show that $(Q', \sigma', \rho', H', \varphi', \lambda)$ satisfy conditions (i)-(vi).

  (i) By definition $fv(Q') \cup fn(Q') \cup fv(H') \cup fn(H') \cup fv(\varphi') \cup fn(\varphi') \subseteq fv(Q) \cup \{x_1, \ldots, x_m\} \cup fn(Q) \cup fv(H) \cup fn(H) \cup \{vc, vm\} \cup fv(\varphi) \cup fn(\varphi)\{vs_j | j \notin \lambda\}$ and by hypothesis $fv(Q') \cup fn(Q') \cup fv(H') \cup fn(H') \cup fv(\varphi')fn(\varphi') \subseteq dom(\rho) \cup \{x_1, \ldots, x_m\} \cup \{vc, vm, vc', vm', vc'', vm''\} \cup \{vs_j, vs'_j, vs''_j | j \notin \lambda\}$ since $dom(\rho') = dom(\rho) \cup \{x_1, \ldots, x_m\} \cup \{vc, vm, vc', vm', vc'', vm''\} \cup \{vs_j, vs'_j, vs''_j | j \notin \lambda\}$ we have $fv(Q') \cup fn(Q') \cup fv(H') \cup fn(H') \cup fv(\varphi') \cup fn(\varphi') \subseteq dom(\rho')$

  (ii) By definition $bn(Q') \cup bv(Q') \subseteq bn(Q) \cup bv(Q)$ and by hypothesis $(bn(Q') \cup bv(Q')) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $\{x_1, \ldots, x_m\} \in bv(Q')$ and $vc, vm, vc', vm', vc'', vm'', vs_1, \ldots, vs_n, vs'_1, \ldots, vs'_n, vs''_1, \ldots, vs''_n$ are fresh, we have that $(bn(Q') \cup bv(Q')) \cap dom(\rho') = \emptyset$

  (iii) By hypothesis $\psi(1), \ldots, \psi(n)$ are ground and by definition $\{vs_j\psi(j) | j \notin \lambda\} \cup \{vc \mapsto (vc', vc''), vm \mapsto (vm', vm'')\}$ is closed. Since σ is closed $\sigma \cup \{vs_j\psi(j) | j \notin \lambda\} \cup \{vc \mapsto (vc', vc''), vm \mapsto (vm', vm'')\}$ is closed, and

hence $\sigma' = \sigma \cup \{vs_j\psi(j) | j \notin \lambda\} \cup \{vc \mapsto (vc', vc''), vm \mapsto (vm', vm'')\}$ is closed.

(iv) By definition Q is under a `lock` $\ldots s_i \ldots$ in P if and only if Q' is under a $lock \ldots s_i \ldots$, so condition (iv) is satisfied by hypothesis.

(v) By hypothesis $\mathcal{C}_{A_0} \supseteq \|\texttt{read}\ s_{j_1}, \ldots, s_{j_m}\ \texttt{as}\ x_1, \ldots, x_m; Q'\|\rho H\varphi\lambda \stackrel{def}{=} \|Q'\|(\rho \cup \{x_j \mapsto (\rho_1\varphi_1'[j_k], \rho_2\varphi_2'[j_k]) | 1 \leq k \leq m\} \cup \{vs_j \mapsto (vs_j', vs_j'') | j \notin \lambda\})(H \wedge message(\varphi_1', vc', vm', \varphi_2', vc'', vm''))\varphi'\lambda \stackrel{def}{=} \|Q'\|\rho' H'\varphi'\lambda$

(vi) Let $msg(xs, M, N, xs', M', N') \in H'$. Then either $msg(xs, M, N, xs', M', N') \in H$ or $msg(xs, M, N, xs', M', N') = msg(\rho_1(\varphi_1'), vc', vm', \rho_2(\varphi_2'), vc'', vm'')$.
If $msg(xs, M, N, xs', M', N') \in H$ by hypothesis $msg(xs\sigma, M\sigma, N\sigma, xs'\sigma, M'\sigma, N'\sigma)$ is derivable from $\mathcal{C}_{A_0}$ since $x_1, \ldots, x_m \notin fn(xs') \cup fn(xs'') \cup fn(M) \cup fn(M') \cup fn(N) \cup fn(N')$ and $vc, vm, vc', vm', vc'', vm'', vs_1, \ldots, vs_n, vs_1', \ldots, vs_n', vs_1'', \ldots, vs_n''$ are fresh, we have that $msg(xs\sigma', M\sigma', N\sigma', xs'\sigma', M'\sigma', N'\sigma')$ is derivable from $\mathcal{C}_{A_0}$.
If $msg(xs, M, N, xs', M', N') = msg(\rho_1(\varphi_1'), vc', vm', \rho_2(\varphi_2'), vc'', vm'')$. By hypothesis $\forall a, a' \notin bn(A_0) \cup bv(A_0)$ $att(\rho_1(\varphi_1\sigma), a, \rho_2(\varphi_2\sigma), a')$ is derivable from $\mathcal{C}_{A_0}$, hence $att(\rho_1(\varphi_1\sigma), vc', \rho_2(\varphi_2\sigma), vc'')$, $att(\rho_1(\varphi_1\sigma), vm', \rho_2(\varphi_2\sigma), vm'')$ are derivable from $\mathcal{C}_{A_0}$. By definition of $\mathcal{C}_{A_0}$, $msg(\rho_1(\varphi_1\sigma), vc', vm', \rho_2(\varphi_2\sigma), vc'', vm'')$ is derivable from $\mathcal{C}_{A_0}$, since $\varphi\sigma \leq \psi$ $msg(\rho_1(\psi_1), vc', vm', \rho_2(\psi_2), vc'', vm'')$ is derivable from $\mathcal{C}_{A_0}$. $\psi = \varphi'\sigma'$ thus $msg(\rho_1(\varphi_1'\sigma'), vc', vm', \rho_2(\varphi_2'\sigma'), vc'', vm'')$ is derivable from $\mathcal{C}_{A_0}$ and since $msg(xs, M, N, xs', M', N') = msg(\rho_1(\varphi_1'), vc', vm', \rho_2(\varphi_2'), vc'', vm'')$ we have that $msg(xs\sigma', M\sigma', N\sigma', xs'\sigma', M'\sigma', N'\sigma')$ is derivable from $\mathcal{C}_{A_0}$, and $\forall C \in H$ we have that σC is derivable from $\mathcal{C}_{A_0}$.

We can thus apply our induction hypothesis to infer that $(\rho'\sigma', \varphi'\sigma', \lambda) \vdash Q'$. Since $\{x_1, \ldots, x_m\} \cap dom(\rho) = \emptyset$ and $vc, vm, vc', vm', vc'', vm'', vs_1, \ldots, vs_n, vs_1', \ldots, vs_n', vs_1'', \ldots, vs_n''$ are fresh and by Lemma C.21 ($\rho\sigma \cup \{x_k \mapsto$

$(\rho_1(\varphi'_1\sigma'(jk)),\ \rho_2(\varphi'_2\sigma'(jk)))|1 \leq k \leq m\}, \varphi'\sigma', \lambda) \vdash Q'$. Since $\psi = \varphi'\sigma'$ we have $(\rho\sigma \cup \{x_k \mapsto (\rho_1(\psi_1(jk)), \rho_2(\psi_2(jk)))|1 \leq k \leq m\}, \psi, \lambda)$ Thus according to our typing system

$$\frac{\begin{array}{c}\forall\psi\ (\varphi\sigma \leq \psi\ \wedge\ \psi = \psi[j \mapsto \varphi\sigma(j) \mid j \in \lambda]),\\ (\rho\sigma \cup \{x_k \mapsto (\rho_1(\psi_1(j_k)), \rho_2(\psi_2(j_k))) \mid 1 \leq k \leq m\}, \psi, \lambda) \vdash Q')\end{array}}{(\rho\sigma, \varphi\sigma, \lambda) \vdash (\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; Q') = Q}\ \mathcal{T}_{read}$$

- **Case** $Q = \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2$. Let p_1, p_2 be patterns such that $\sigma\rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and $\sigma\rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_2$ Let $\rho' = \rho\sigma \cup \{x \mapsto (p_1, p_2)\}$, $H' = H\sigma, \varphi' = \varphi\sigma, H'' = H \wedge \rho_1(fails(\texttt{fst}(D))) \wedge \rho_2(fails(\texttt{snd}(D)))$ where $fails(D) =_{\sigma|D\overset{\wedge}{\Downarrow'}(p,\sigma)} nounif(D, GVar(\sigma D))$ We show that $(Q_1, \sigma, \rho', H', \varphi', \lambda)$ and $(Q_2, \sigma, \rho, H'', \varphi, \lambda)$ satisfy conditions (i)-(vi).

  (i) by definition $fn(Q_1) \cup fv(Q_1) \cup fn(H') \cup fv(H') \cup fn(\varphi') \cup fv(\varphi') \subseteq fn(Q) \cup fv(Q) \cup \{x\} \cup fn(H\sigma) \cup fv(H\sigma) \cup fn(\varphi\sigma) \cup fv(\varphi\sigma)$ and $fn(Q_1) \cup fv(Q_1) \cup fn(H') \cup fv(H') \cup fn(\varphi') \cup fv(\varphi') \subseteq fn(Q) \cup fv(Q) \cup \{x\} \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi) \cup fn(D) \cup fv(D)$ and by definition $fn(Q_1) \cup fv(Q_1) \cup fn(H') \cup fv(H') \cup fn(\varphi') \cup fv(\varphi') \subseteq fn(Q) \cup fv(Q) \cup \{x\} \cup fn(H) \cup fv(H) \cup fn(\varphi) \cup fv(\varphi)$. By hypothesis $fn(Q_1) \cup fv(Q_1) \subseteq dom(\rho) \cup \{x\} = dom(\rho')$ hence ρ binds the free names and variables of Q_1, H', φ'.

  By definition $fn(Q_2) \cup fv(Q_2) \subseteq fn(Q) \cup fv(Q)$ and by hypothesis $fn(Q_2) \cup fv(Q_2) \subseteq dom(\rho)$ then ρ binds the free names and variables of Q_2, H'', φ

  (ii) By definition $bn(Q_1) \cup bv(Q1) \subseteq bn(Q) \cup bv(Q)$ and $(bn(Q1) \cup bv(Q1)) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$ since $x \notin bv(Q_1)$ we have that $(bn(Q1) \cup bv(Q1)) \cap dom(\rho') = \emptyset$.

  By definition $bn(Q_2) \cup bv(Q_2) \subseteq bn(Q) \cup bv(Q)$ and $(bn(Q_2) \cup bv(Q_2)) \cap dom(\rho) \subseteq (bn(Q) \cup bv(Q)) \cap dom(\rho) = \emptyset$

(iii) By hypothesis σ is closed.

(iv) By definition Q is under a $\texttt{lock}\ \ldots s_i \ldots$ in P if and only if Q_1 and Q_2 are under a $\texttt{lock}\ \ldots s_i \ldots$, so condition (iv) is satisfied by hypothesis.

(v) By hypothesis $\mathcal{C}_{A_0} \supseteq \|\texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2\|\rho H\varphi_1\varphi_2\lambda \stackrel{def}{=}$ $\{\|Q_1\|\rho\sigma \cup \{x \mapsto (p_1, p_2)\}H\sigma\varphi_1\sigma\varphi_2\sigma\lambda \mid$ $(\rho_1(\texttt{fst}(D)), \rho_2(\texttt{fst}(D))) \Downarrow' ((p, p'), \sigma)\} \cup \|Q_2\|\rho H''\varphi_1\varphi_2\lambda \cup \{\sigma H$ $\wedge\sigma\rho_2(fails(\texttt{snd}(D))) \rightarrow bad|\rho_1(\texttt{fst}(D)) \Downarrow' (p, \sigma)\} \cup \{\sigma H$ $\wedge\sigma\rho_1(fails(\texttt{fst}(D))) \rightarrow bad|\rho_2(\texttt{snd}(D)) \Downarrow' (p', \sigma)\}$ where $fails(D) =$ $\bigwedge_{\sigma|D\Downarrow'(p,\sigma)} nounif(D, GVar(\sigma D))$

(vi) Let $msg(xs_1, M_1, N_1, xs_2, M_2, N_2) \in H'$. Since $H' = H\sigma$ we have that $msg(xs_1, M_1, N_1, xs_2, M_2, N_2) \in H\sigma$ and by definition $msg(xs'_1, M'_1, N'_1,$ $xs'_2, M'_2, N'_2) \in H \wedge xs_1 = xs'_1\sigma \wedge M_1 = M'_1\sigma \wedge N_1 = N'_1\sigma \wedge xs_2 =$ $xs'_2\sigma \wedge M_2 = M'_2\sigma \wedge N_2 = N'_2\sigma$ by hypothesis $msg(xs_1\sigma, M_1\sigma, N_1\sigma, xs_2\sigma,$ $M_2\sigma, N_2\sigma)$ is derivable from $\mathcal{C}_{A_0}$.
Let $msg(xs_1, M_1, N_1, xs_2, M_2, N_2) \in H$. By hypothesis,
$msg(xs_1\sigma, M_1\sigma, N_1\sigma, xs_2\sigma, M_2\sigma, N_2\sigma) \in H'$ is derivable from $\mathcal{C}_{A_0}$.
If $\nexists p_1$ such that $\sigma\rho_1(\texttt{fst}(D)) \Downarrow_\Sigma p_1$ and $\nexists p_2$ such that $\sigma\rho_2(\texttt{snd}(D)) \Downarrow_\Sigma$ p_2 by Lemma C.18 $\sigma\rho_1(fails(\texttt{fst}(D))$ and $\sigma\rho_2(fails(\texttt{snd}(D))$ are true so $\sigma(H \wedge \rho_1(fails(\texttt{fst}(D)) \wedge \sigma\rho_2(fails(\texttt{snd}(D)))$ can be derived from $\mathcal{C}_{A_0}$

We can now apply our induction hypothesis to infer that $(\rho', \varphi', \lambda) \vdash Q_1$ and $(\rho, \varphi, \lambda) \vdash Q_2$. Since $(\rho', \varphi', \lambda) \vdash Q_1$ by definition $(\rho\sigma \cup \{x \mapsto (p_1, p_2)\}, \varphi\sigma, \lambda)$ $\vdash Q_1$ If $\exists p_1$ such that $\sigma\rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1$ and $\nexists p_2$ such that $\sigma\rho_2(\texttt{snd}(D)) \Downarrow_\Sigma$ p_2, then $bad \in \mathcal{F}_{A_0}$. By a variant of Lemma C.2 for patterns instead of terms, $\exists p'_1, \sigma'$ and σ'' such that $\rho_1(\texttt{fst}(D)) \Downarrow' (p'_1, \sigma'), p_1 = \sigma''p'_1$, and $\sigma = \sigma''\sigma'$ except on the fresh variables introduced in the computation of $\rho_1(\texttt{fst}(D)) \Downarrow' (p'_1, \sigma')$. There exists no p_2 such that $\sigma''\sigma'\rho_2(\texttt{snd}(D)) \Downarrow_\Sigma p_2$, so by Lemma C.18, $\sigma''\sigma'\rho_2(fails(\texttt{snd}(D)))$ holds, hence $\sigma''(\sigma' H \wedge$

$\sigma'\rho_2(fails(\texttt{snd}(D))))$ can be derived from $\mathcal{C}_{A_0}$.

If $\exists p_2$ such that $\sigma\rho_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2$ and $\nexists p_1$ such that $\sigma\rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma} p_1$, then $bad \in \mathcal{F}_{A_0}$. This property follows from the previous one by symmetry.

Hence according to our typing system:

$$\frac{\begin{array}{c}(\forall\ p_1, p_2\ \rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \rho_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2 \\ (\rho[x \mapsto (p1, p2)], \varphi', \lambda) \vdash Q_1) \\ (\text{if } \nexists\, p_1,\ \rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ \rho_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ (\rho, \varphi, \lambda) \vdash Q_2) \\ (\text{if } \exists\, p_1,\ \rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \nexists\, p_2,\ \rho_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ bad \in \mathcal{F}_{A_0}) \\ (\text{if } \nexists\, p_1,\ \rho_1(\texttt{fst}(D)) \Downarrow_{\Sigma'} p_1 \text{ and } \exists\, p_2,\ \rho_2(\texttt{snd}(D)) \Downarrow_{\Sigma'} p_2 \text{ then} \\ bad \in \mathcal{F}_{A_0})\end{array}}{(\rho, \varphi, \lambda) \vdash \texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2}\ T_{let}$$

To conclude the proof of Lemma C.24 we then need to show that $\rho = E0$, σ such that $dom(\sigma) = \emptyset, H = true, \varphi = S0$ and $\lambda = \emptyset$ satisfy conditions (i)-(vii).

(i) Since by hypotheses $fv(P) = \emptyset$ and $fn(P) \subseteq dom(E0)$ by construction, ρ binds the free names and variables of P, H and φ.

(ii) By definition $(bn(P) \cup bv(P)) \cap dom(\rho) = \emptyset$.

(iii) By definition σ is a closed substitution.

(iv) P is not under any lock in P , thus $\emptyset$ satisfies condition (iv).

(v) By definition $\mathcal{C}_{A_0} \supseteq \|P'\|\rho H\varphi_1\varphi_2\lambda$.

(vi) By definition $H\sigma = true$, and thus $H\sigma$ can trivially be derived from $\mathcal{C}_{A_0}$.

Thus, $P, \rho, \sigma, H, \varphi, \lambda$ satisfy the conditions of our induction result according to which $(E0, S0, \lambda) \vdash P$. □

C.7.5 Typability of $C[A_0]$

Let $A_0 = (\mathcal{E}_0, \mathcal{S}_0, \{(P_0, \emptyset)\})$ be an initial biprocess configuration.

Lemma C.25. Typability of $C[A_0]$ *We show that* $E_0 \vdash \Lambda_0; (\mathcal{E}_0, \mathcal{S}_0, \{(C' \mid P_0, \emptyset)\})$

Proof. In order to prove that $E_0 \vdash \Lambda_0; (\mathcal{E}_0, \mathcal{S}_0, \{(P_0, \emptyset)\})$ we show that $(E_0, \mathcal{S}_0, \emptyset) \vdash P_0$ by induction on C'. If $C' = \_$ the result follows from the typability of the protocol, Lemma C.24. If $C' = \texttt{new}\ a : a[]; C''$ the result follows by induction hypothesis and the type rule $\mathcal{T}_{new}$. If $C' = C'' \mid Q$ the result follows from the typability of the adversary and rule $\mathcal{T}_{par}$. □

C.7.6 Subject Reduction

Lemma C.26. (Subject reduction)*: Let* $\Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ *be an instrumented biprocess configuration such that no* $[s \mapsto M]$ *occurs in* $\mathcal{P}$*, names and variables are bound at most once in* $\mathcal{P}$*, and* $cells(\mathcal{P}) \subseteq \{s_1, \ldots, s_n\}$*. Let* E *be a mapping from names and variables to pair of patterns. If* $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ *and* $\Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}) \to \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$*, then* $E' \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$*.*

Proof. We prove by induction on the derivation rule R applied for the transition $\Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}) \to \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$ that if $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P})$ and $\Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P} \cup \{(P, \lambda)\}) \to \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q}' \cup \{(Q, \lambda')\})$ then $(E', \mathcal{S}', \lambda') \vdash Q$, and $E' \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q}' \cup \{(Q, \lambda')\})$.

- **Case** R = Red Nil. In this case, $\Lambda' = \Lambda, \mathcal{E} = \mathcal{E}', \mathcal{S} = \mathcal{S}', \mathcal{P} = \mathcal{P}' \cup \{(0, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{Q})$ since $E = \mathcal{E}', \mathcal{S} = \mathcal{S}', \Lambda' = \Lambda$, we have that $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$.
- **Case** R = Red Repl. In this case, $\Lambda' = \Lambda \setminus \{l\}, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(!^i P, \emptyset)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(!^i P \mid P\{^l/_i\}, \emptyset)\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then

either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') = (!^i P \mid P\{^l/_i\}, \emptyset)$

If $(Q', \lambda'') \in \mathcal{P}'$ since by hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ and $E = \mathcal{E}', \mathcal{S} = \mathcal{S}'$, we have that $E \vdash \Lambda; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$ Let $E' = E \cup \{i \mapsto (l, l)\}$ since i does not appear in Q', $E' \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$.

If $(Q', \lambda'') = (!^i P \mid P\{^l/_i\}, \emptyset)$ by hypothesis $(E, \mathcal{S}, \emptyset) \vdash !^i P$ since $\mathcal{S}' = \mathcal{S}$, we have $(E, \mathcal{S}', \emptyset) \vdash P$. Let $E' = E \cup \{i \mapsto (l, l)\}$ by rule $\mathcal{T}_{repl}$, $(E', \mathcal{S}, \emptyset) \vdash P$ since $\mathcal{S}' = \mathcal{S}$, we have $(E', \mathcal{S}', \emptyset) \vdash P$. By Lemma C.21 we have that $(E, \mathcal{S}', \emptyset) \vdash P\{^l/_i\}$ and by rule $\mathcal{T}_{par}$ we have that $(E, \mathcal{S}', \emptyset) \vdash (!^i P \mid P\{^l/_i\}, \emptyset)$. By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(!^i P) = \emptyset$ hence $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P}) = Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P}') \cup Label(!^i P) = Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P}')$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}) = Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}') \cup Label(!^i P) = Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}')$ contain no duplicates. Since $Label(P) \cap Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P}') = Label(P) \cap Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}') = \emptyset$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P}' \cup \{(P, \emptyset)\})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}' \cup \{(P, \emptyset)\})$ contain no duplicates. $Label(\Lambda') \subseteq Label(\Lambda)$ hence $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P}' \cup \{(P, \emptyset)\})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P}' \cup \{(P, \emptyset)\})$ contain no duplicates. Since $\mathcal{Q} = \mathcal{P}' \cup \{(!^i P \mid P\{^l/_i\}, \emptyset)\}$ we have that $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = Red Par. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup (P_1 | P_2, \emptyset)$, and $\mathcal{Q} = \mathcal{P}' \cup \{(P_1, \emptyset), (P_2, \emptyset)\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') \in \{(P_1, \emptyset), (P_2, \emptyset)\}$.

  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}$ we have that $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.

  If $(Q', \lambda'') = (P_1, \emptyset)$ by rule $\mathcal{T}_{par}$ we have that $(\mathcal{E}, \mathcal{S}, \emptyset) \vdash P_1$ since $\mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}$ we have that $(\mathcal{E}', \mathcal{S}', \emptyset) \vdash P_1$ If $(Q', \lambda'') = (P_2, \emptyset)$ by rule $\mathcal{T}_{par}$ we have that $(\mathcal{E}, \mathcal{S}, \emptyset) \vdash P_2$ since $\mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}$ we have that $(\mathcal{E}', \mathcal{S}', \emptyset) \vdash P_2$. By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$

contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. $\mathcal{P} = \mathcal{P}' \cup \{(P_1 \mid P_2, \emptyset)\}$ hence $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P}') \cup Label(\{(P_1 \mid P_2, \emptyset)\})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}) \cup Label(\{(P_1 \mid P_2, \emptyset)\})$ contain no duplicates. By definition $Label(\{(P_1 \mid P_2, \emptyset)\}) = Label(\{(P_1, \emptyset)\}) \cup Label(\{(P_2, \emptyset)\})$ hence $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P}') \cup Label(\{(P_1, \emptyset)\}) \cup Label(\{(P_2, \emptyset)\})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P}) \cup Label(\{(P_1, \emptyset)\}) \cup Label(\{(P_2, \emptyset)\})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = Red New. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E} \cup \{a_0[M_1, \ldots, M_n]\}$, $\mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q\{^{a_0[M_1,\ldots,M_n]}/_a\}, \lambda)\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') = (Q\{^{a_0[M_1,\ldots,M_n]}/_a\}, \lambda)$.
  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}$ and $a_0[M_1, \ldots, M_n]$ is fresh, in fact by hypothesis $Label(\mathcal{P})$ does not contain duplicates, we have that $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.
  If $(Q', \lambda'') = (Q\{^{a_0[M_1,\ldots,M_n]}/_a\}, \lambda)$ Let $E' = E[a \mapsto (a_0[E_1(\texttt{fst}(M_1)), \ldots, E_1(\texttt{fst}(M_n))], a_0[E_2(\texttt{snd}(M_1)), \ldots, E_2(\texttt{snd}(M_n))])]$ then by rule $\mathcal{T}_{new}$, $(E', \mathcal{S}, \lambda) \vdash Q$ and by Lemma C.21 $(E, \mathcal{S}, \lambda) \vdash Q\{^{a_0[M_1,\ldots,M_n]}/_a\}$ since $\mathcal{S}' = \mathcal{S}$ we have that $(E, \mathcal{S}', \lambda) \vdash Q\{^{a_0[M_1,\ldots,M_n]}/_a\}$. By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates.
  By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label(\texttt{new}\ a : a_0[M_1, \ldots, M_n]; Q) = Label(\mathcal{P}') \cup \{(a_0, last(M_1, \ldots, M_n)\} \cup Label(Q) \supseteq Label(\mathcal{P}') \cup Label(Q) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = Red Fun1. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{\texttt{let}\ x = D\ \texttt{in}\ Q_1\ \texttt{else}\ Q_2\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q_1\{^{\texttt{diff}[M,M']}/_x\}, \lambda)\}$ and $\texttt{fst}(D) \Downarrow_\sigma M, \texttt{snd}(D) \Downarrow_\sigma M'$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or

$(Q', \lambda'') = (Q_1\{^{\texttt{diff}[M,M']}/_x\}, \lambda)$.

If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.

If $(Q', \lambda'') = (Q_1\{^{\texttt{diff}[M,M']}/_x\}, \lambda)$ then by rule $\mathcal{T}_{let}$, $(E, \mathcal{S}, \lambda) \vdash Q_1\{^{\texttt{diff}[M,M']}/_x\}$ and since $\mathcal{S}' = \mathcal{S}, E' = E$ we have that $(E', \mathcal{S}', \lambda) \vdash Q_1\{^{\texttt{diff}[M,M']}/_x\}$. By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label(\texttt{let } x = D \texttt{ in } Q_1 \texttt{ else } Q_2) = Label(\mathcal{P}') \cup Label(Q_1) \cup Label(Q_2) \supseteq Label(\mathcal{P}') \cup Label(Q_1\{^{\texttt{diff}[M,M']}/_x\}) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED FUN2. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{\texttt{let } x = D \texttt{ in } Q_1 \texttt{ else } Q_2\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q_2, \lambda)\}$ and $\nexists M, M'$ such that $\texttt{fst}(D) \Downarrow_\sigma M, \texttt{snd}(D) \Downarrow_\sigma M'$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') = (Q_2, \lambda)$.

  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.

  If $(Q', \lambda'') = (Q_2, \lambda)$ then by rule $\mathcal{T}_{let}$, $(E, \mathcal{S}, \lambda) \vdash Q_2$ and since $\mathcal{S}' = \mathcal{S}, E' = E$ we have that $(E', \mathcal{S}', \lambda) \vdash Q_2$. By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label(\texttt{let } x = D \texttt{ in } Q_1 \texttt{ else } Q_2) = Label(\mathcal{P}') \cup Label(Q_1) \cup Label(Q_2) \supseteq Label(\mathcal{P}') \cup Label(Q_2) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED COMM. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(\mathtt{out}(M, N); Q_1, \lambda_1), (\mathtt{in}(M', x); Q_2, \lambda_2)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q_1, \lambda_1), (Q_2\{^N/_x\}, \lambda_2)\}$ and $\Sigma \vdash \mathtt{fst}(M) = \mathtt{fst}(M'), \Sigma \vdash \mathtt{snd}(M) = \mathtt{snd}(M')$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') \in \{(Q_1, \lambda_1), (Q_2\{^N/_x\}, \lambda_2)\}$.
  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.
  If $(Q', \lambda'') = (Q_1, \lambda_1)$ then by rule $\mathcal{T}_{out}$, $(E, \mathcal{S}, \lambda) \vdash Q_1$ and since $\mathcal{S}' = \mathcal{S}$ we have that $(E, \mathcal{S}', \lambda) \vdash Q_1$.
  If $(Q', \lambda'') = (Q_2\{^N/_x\}, \lambda_2)$ let $E' = E[x \mapsto (E_1(\mathtt{fst}(N)), E_2(\mathtt{snd}(N)))]$ then by rule $\mathcal{T}_{in}$, $(E', \mathcal{S}, \lambda_2) \vdash Q_2$ and since $\mathcal{S}' = \mathcal{S}$ we have that $(E', \mathcal{S}', \lambda_2) \vdash Q_2$ by Lemma C.21 we have that $(E, \mathcal{S}', \lambda_2) \vdash Q_2\{^N/_x\}$.
  By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label((\mathtt{out}(M, N); Q_1, \lambda_1), (\mathtt{in}(M', x); Q_2, \lambda_2)) = Label(\mathcal{P}') \cup Label((\mathtt{out}(M, N); Q_1, \lambda_1)) \cup Label((\mathtt{in}(M', x); Q_2, \lambda_2)) \supseteq Label(\mathcal{P}') \cup Label((Q_1, \lambda_1), (Q_2\{^N/_x\}, \lambda_2)) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED STATE. This case cannot happen since $[s \mapsto M]$ does not occur in $\mathcal{P}$

- **Case** R = RED LOCK. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(\mathtt{lock}\ s_{i_1}, \ldots, s_{i_m}; Q, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q, \lambda')\}, \lambda' = \lambda \cup \{s_{i_1}, \ldots, s_{i_m}\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') \in \{(Q, \lambda')\}$.
  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.
  If $(Q', \lambda'') = (Q, \lambda')$ then by rule $\mathcal{T}_{lock}$, $(E, \mathcal{S}, \lambda') \vdash Q$ and since $\mathcal{S}' = \mathcal{S}, E' = E$ we have that $(E', \mathcal{S}', \lambda') \vdash Q_1$.
  By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup$

$Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label((\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; Q, \lambda)) = Label(\mathcal{P}') \cup Label((\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}, \lambda)) \cup Label((Q, \lambda')) \supseteq Label(\mathcal{P}') \cup Label((Q, \lambda')) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED UNLOCK. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(\texttt{unlock}\ s_{i_1}, \ldots, s_{i_m}; Q, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q, \lambda')\}, \lambda' = \lambda \setminus \{s_{i_1}, \ldots, s_{i_m}\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') \in \{(Q, \lambda')\}$.
  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.
  If $(Q', \lambda'') = (Q, \lambda')$ then by rule $\mathcal{T}_{unlock}$, $(E, \mathcal{S}, \lambda') \vdash Q$ and since $\mathcal{S}' = \mathcal{S}, E' = E$ we have that $(E', \mathcal{S}', \lambda') \vdash Q$.
  By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label((\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}; Q, \lambda)) = Label(\mathcal{P}') \cup Label((\texttt{lock}\ s_{i_1}, \ldots, s_{i_m}, \lambda)) \cup Label((Q, \lambda')) \supseteq Label(\mathcal{P}') \cup Label((Q, \lambda')) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED READ. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}, \mathcal{P} = \mathcal{P}' \cup \{(\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; Q, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, {}^{\mathcal{S}(s_{i_m})}/_{x_m}\}, \lambda)\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') = (Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, {}^{\mathcal{S}(s_{i_m})}/_{x_m}\}, \lambda)$.
  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{S}' = \mathcal{S}, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.
  If $(Q', \lambda'') = (Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, {}^{\mathcal{S}(s_{i_m})}/_{x_m}\}, \lambda)$. Let $E' = E[x_1 \mapsto (E_1(\mathcal{S}_1(s_{i_1})), E_2(\mathcal{S}_2(s_{i_1}))), \ldots, x_m \mapsto (E_1(\mathcal{S}_1(s_{i_m})),$

$E_2(\mathcal{S}_2(s_{i_m})))]$ then by rule $\mathcal{T}_{read}$, $(E', \mathcal{S}, \lambda) \vdash Q$ and since $\mathcal{S}' = \mathcal{S}$ we have that $(E', \mathcal{S}', \lambda) \vdash Q$ and by Lemma C.21 we have that $(E, \mathcal{S}', \lambda) \vdash Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, ^{\mathcal{S}(s_{i_m})}/_{x_m}\}$.

By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label((\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m; Q, \lambda))$ $= Label(\mathcal{P}') \cup Label((\texttt{read}\ s_{i_1}, \ldots, s_{i_m}\ \texttt{as}\ x_1, \ldots, x_m, \lambda)) \cup Label((Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, ^{\mathcal{S}(s_{i_m})}/_{x_m}\}, \lambda)) \supseteq Label(\mathcal{P}') \cup Label((Q\{^{\mathcal{S}(s_{i_1})}/_{x_1}, \ldots, ^{\mathcal{S}(s_{i_m})}/_{x_m}\}, \lambda)) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

- **Case** R = RED ASSIGN. In this case, $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}, \mathcal{S}' = \mathcal{S}[s_{i_1} \mapsto M_1, \ldots, s_{i_m} \mapsto M_m], \mathcal{P} = \mathcal{P}' \cup \{(s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; Q, \lambda)\}$, and $\mathcal{Q} = \mathcal{P}' \cup \{(Q, \lambda)\}$. Let $(Q', \lambda'') \in \mathcal{Q}$, then either $(Q', \lambda'') \in \mathcal{P}'$ or $(Q', \lambda'') = (Q, \lambda)$.

  If $(Q', \lambda'') \in \mathcal{P}'$. By hypothesis $E \vdash \Lambda; (\mathcal{E}, \mathcal{S}, \mathcal{P}')$ since $\Lambda' = \Lambda, \mathcal{E}' = \mathcal{E}$ we have $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}, \mathcal{P}')$ since by rule RED ASSIGN $\forall (Q', \lambda') \in \mathcal{P}', \lambda' \cap \{i_1, \ldots\ i_m\} = \emptyset$ we have that $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{P}')$.

  If $(Q', \lambda'') = (Q, \lambda)$ then by rule $\mathcal{T}_{assign}$, $(E, \mathcal{S}', \lambda) \vdash Q$.

  By hypothesis $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{P})$ contain no duplicates. Since $\Lambda' = \Lambda$ we have $Label(E_1) \cup Label(\Lambda') \cup Label(\mathcal{P})$ and $Label(E_2) \cup Label(\Lambda') \cup Label(\mathcal{P})$ contain no duplicates. By definition $Label(\mathcal{P}) = Label(\mathcal{P}') \cup Label((s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m; Q, \lambda)) = Label(\mathcal{P}') \cup Label((s_{i_1}, \ldots, s_{i_m} := M_1, \ldots, M_m, \lambda)) \cup Label((Q, \lambda)) \supseteq Label(\mathcal{P}') \cup Label((Q, \lambda)) = Label(\mathcal{Q})$ we have that $Label(E_1) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ and $Label(E_2) \cup Label(\Lambda) \cup Label(\mathcal{Q})$ contain no duplicates. Hence $E \vdash \Lambda'; (\mathcal{E}', \mathcal{S}', \mathcal{Q})$

□

Bibliography

[3GP01] 3GPP. Technical specification group services and system aspects; 3G security; formal analysis of the 3G authentication protocol (release 4), TS 33.902. Technical Specification TR 33.902, V4.0.0, 3rd Generation Partnership Project, 2001.

[3GP10a] 3GPP. Generic Access Network (GAN); Mobile GAN interface layer 3 specification, TS 44.318. Technical Specification TS 44.318 v9.2.0, 3rd Generation Partnership Project, 2010.

[3GP10b] 3GPP. Generic Access Network (GAN); Stage 2, TS 43.318. Technical Specification TS 43.318 v9.0.0, 3rd Generation Partnership Project, 2010.

[3GP10c] 3GPP. Technical specification group core network and terminals; mobile radio interface layer 3 specification; core network protocols; stage 3 (release 9), TS 24.008. Technical Report TS 24.008 V9.4.0, 3rd Generation Partnership Project, 2010.

[3GP10d] 3GPP. Technical specification group services and system aspects; 3G security; security architecture (release 9), TS 33.102. Technical Specification TS 33.102 V9.3.0, 3rd Generation Partnership Project, 2010.

[3GP10e] 3GPP. Technical specification group services and system aspects; security aspects (release 4), TS 42.009. Technical Specification TS 42.009 V4.3.0, 3rd Generation Partnership Project, 2010.

[3GP11a] 3GPP. Security of Home Node B (HNB) / Home evolved Node B (HeNB), TS 33.302. Technical Specification TS 33.302 v11.2.0, 3rd Generation Partnership Project, 2011.

[3GP11b] 3GPP. Technical specification group radio access network; radio resource control (RRC); protocol specification (release 10), TS 25.331. Technical Report TS 25.331, V10.5.0, 3rd Generation Partnership Project, 2011.

[3GP11c] 3GPP. Technical specification group services and system aspects; 3G security; cryptographic algorithm requirements (release 10), TS 33.105. Technical Specification TS 33.105 V10.0.0, 3rd Generation Partnership Project, 2011.

[AB05] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *Journal of the ACM*, 52(1):102–146, January 2005.

[ABB+05] Alessandro Armando, David Basin, Yohan Boichut, Yannick Chevalier, Luca Compagna, Jorge Cuellar, P. Hankes Drielsma, P. C. Heám, O. Kouchnarenko, J. Mantovani, S. Mödersheim, D. von Oheimb, M. Rusinowitch, J. Santiago, M. Turuani, L. Viganò, and L. Vigneron. The avispa tool for the automated validation of internet security protocols and applications. In *International Conference on Computer Aided Verification*, CAV'05, pages 281–285, Berlin, Heidelberg, 2005. Springer-Verlag.

[ABR12] Myrto Arapinis, Sergiu Bursuc, and Mark Ryan. Privacy supporting cloud computing: Confichair, a case study. In Pierpaolo Degano and JoshuaD. Guttman, editors, *Principles of Security and Trust, POST*, volume 7215 of *Lecture Notes in Computer Science*, pages 89–108. Springer Berlin Heidelberg, 2012.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1-2):2–32, 2006.

[ACC+08a] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuellar, and Llanos Tobarra. Formal analysis of saml 2.0 web browser single sign-on: Breaking the saml-based single sign-on for google apps. In *ACM Workshop on Formal Methods in Security Engineering*, FMSE '08, pages 1–10, New York, NY, USA, 2008. ACM.

[ACC+08b] Alessandro Armando, Roberto Carbone, Luca Compagna, Jorge Cuéllar, and M. Llanos Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. In *ACM Workshop on Formal Methods in Security Engineering*, FMSE, 2008.

[ACC14] Alessandro Armando, Roberto Carbone, and Luca Compagna. Satmc: a sat-based model checker for security-critical systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, pages 31–45. Springer, 2014.

[ACRR09] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Untraceability in the applied pi calculus. In *International Workshop on RFID Security and Cryptography*, 2009.

[ACRR10] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Computer Security Foundations Symposium, CSF*, CSF, 2010.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL, 2001.

[AG97] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: The spi calculus. In *Conference on Computer and Communications Security, CCS*, CCS, pages 36–47, New York, NY, USA, April 1997. ACM.

[ALRR14] Myrto Arapinis, Jia Liu, Eike Ritter, and Mark D. Ryan. Stateful applied pi calculus. In Martn Abadi and Steve Kremer, editors, *Principles of Security and Trust, POST*, volume 8414 of *Lecture Notes in Computer Science*, pages 22–41. Springer Berlin Heidelberg, 2014.

[AMR+12] Myrto Arapinis, Loretta Mancini, Eike Ritter, Mark Ryan, Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. New privacy issues in mobile telephony: Fix and verification. In *Conference on Computer and Communications Security*, CCS, pages 205–216. ACM, 2012.

[AMRR14] Myrto Arapinis, Loretta Ilaria Mancini, Eike Ritter, and Mark Ryan. Privacy through pseudonymity in mobile telephony systems. In *Network and Distributed System Security Symposium, NDSS*, 2014.

[AO05] Gildas Avoine and Philippe Oechslin. RFID Traceability: A Multilayer Problem. In *Financial Cryptography*, FC, 2005.

[AR00] Martin Abadi and Phillip Rogaway. Reconciling two views of cryptography. In Jan van Leeuwen, Osamu Watanabe, Masami Hagiya, PeterD. Mosses, and Takayasu Ito, editors, *Theoretical Computer Science: Exploring New Frontiers of Theoretical Informatics*, volume 1872 of *Lecture Notes in Computer Science*, pages 3–22. Springer Berlin Heidelberg, 2000.

[ARR11] Myrto Arapinis, Eike Ritter, and Mark Ryan. Statverif: Verification of stateful processes. In *Computer Security Foundations Symposium, CSF*, CSF, 2011.

[ASS09] Zahra Ahmadian, Somayeh Salimi, and Ahmad Salahi. New attacks on UMTS network access. In *Conference on Wireless Telecommunications Symposium*, WTS'09, 2009.

[BAF05] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated Verification of Selected Equivalences for Security Protocols. In *Symposium on Logic in Computer Science*, LICS, 2005.

[BAF08] Bruno Blanchet, Martín Abadi, and Cédric Fournet. Automated verification of selected equivalences for security protocols. *Journal of Logic and Algebraic Programming*, 75(1):3–51, 2008.

[BAN90] Michael Burrows, Martín Abadi, and Roger M. Needham. A logic of authentication. *ACM Transactions on Computer System*, 8(1):18–36, 1990.

[Bau05] Mathieu Baudet. Deciding security of protocols against off-line guessing attacks. In *Conference on Computer and Communications Security, CCS*, CCS, 2005.

[BBDM13] Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. Keys to the cloud: Formal analysis and concrete attacks on encrypted web storage. In *Principles of Security and Trust, POST*, POST, pages 126–146, 2013.

[BBDM14] Chetan Bansal, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, and Sergio Maffeis. Discovering concrete attacks on website authorization by formal analysis. *Journal of Computer Security*, 22(4):601–657, 2014.

[BBM12] Chetan Bansal, Karthikeyan Bhargavan, and Sergio Maffeis. Discovering concrete attacks on website authorization by formal analysis. In *Computer Security Foundations Symposium, CSF*, CSF, pages 247–262, 2012.

[BCD09] Mathieu Baudet, Véronique Cortier, and Stéphanie Delaune. Yapa: A generic tool for computing intruder knowledge. In Ralf Treinen, editor, *Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 148–163. Springer Berlin Heidelberg, 2009.

[BCdH10] Mayla Brusò, Konstantinos Chatzikokolakis, and Jerry den Hartog. Formal verification of privacy for rfid systems. In *Computer Security Foundations Symposium, CSF*, CSF, pages 75–88, 2010.

[BCFS10] Matteo Bortolozzo, Matteo Centenaro, Riccardo Focardi, and Graham Steel. Attacking and fixing PKCS#11 security tokens. In *Conference on Computer and Communications Security, CCS*, CCS, 2010.

[BDE09] Murat A. Bayir, Murat Demirbas, and Nathan Eagle. Discovering spatiotemporal mobility profiles of cellphone users. In *World of Wireless, Mobile and Multimedia Networks Workshops, 2009. WoWMoM 2009*, pages 1–9, june 2009.

[ben] Benetton Boycott Campaign. `http://www.boycottbenetton.com/`.

[BHKO04] Yohan Boichut, Pierre-Cyrille Héam, Olga Kouchnarenko, and F. Oehl. Improvements on the Genet and Klay technique to automatically verify security protocols. In *International Workshop on Automated Verification of Infinite-State Systems (AVIS'2004), joint to ETAPS'04*, pages 1–11, Barcelona, Spain, April 2004.

[BJPV11] Jesper Bengtson, Magnus Johansson, Joachim Parrow, and BjÃ Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *arXiv preprint arXiv:1101.3262*, 2011.

[blaa] BladeRF. `http://nuand.com/`.

[Blab] Bruno Blanchet. Proverif: Cryptographic protocol verifier in the formal model. `http://www.proverif.ens.fr/`.

[Bla01] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *Computer Security Foundations Symposium, CSF*, CSFW, 2001.

[Bla02] Bruno Blanchet. From secrecy to authenticity in security protocols. In *International Symposium on Static Analysis*, SAS '02, pages 342–359, London, UK, UK, 2002. Springer-Verlag.

[Bla09] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, July 2009.

[BMV03] David Basin, Sebastian Mdersheim, and Luca Vigan. An on-the-fly model-checker for security protocol analysis. In Einar Snekkenes and Dieter Gollmann, editors, *European Symposium on Research in Computer Security, ESORICS*, volume 2808 of *Lecture Notes in Computer Science*, pages 253–270. Springer Berlin Heidelberg, 2003.

[BS03] Alastair R. Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, January 2003.

[BSW00] Alex Biryukov, Adi Shamir, and David Wagner. Real time cryptanalysis of a5/1 on a pc. In *International Workshop on Fast Software Encryption*, page 118, 2000.

[Bur] Burgess, David A. and Samra, Harvind and et al. OpenBTS. `http://openbts.sourceforge.net/`.

[BXR13] Ian Batten, Shiwei Xu, and Mark Ryan. Dynamic measurement and protected execution: model and analysis. In *International Symposium on Trustworthy Global Computing, TGC*, August 2013. to appear.

[BZj06] Michael Barbaro and Tom Zeller jr. A face is exposed for aol searcher no. 4417749. *The New York Times*, 2006.

[Cas05] Johann. Cas. Privacy in pervasive computing environments - a contradiction in terms? *Technology and Society Magazine, IEEE*, 24(1):24–33, Spring 2005.

[CB13] Vincent Cheval and Bruno Blanchet. Proving more observational equivalences with proverif. In David Basin and John Mitchell, editors, *Principles of Security and Trust, POST*, Lecture Notes in Computer Science, pages 226–246, Roma, Italy, March 2013. Springer.

[CCLD10] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. Automating security analysis: Symbolic equivalence of constraint systems. In *International Joint Conference on Automated Reasoning*, IJCAR, 2010.

[CD09] Véronique Cortier and Stéphanie Delaune. A method for proving observational equivalence. In *Computer Security Foundations Symposium, CSF*, CSF, 2009.

[CDK09] Ştefan Ciobâcă, Stéphanie Delaune, and Steve Kremer. Computing knowledge in security protocols under convergent equational theories. In Renate Schmidt, editor, *International Conference on Automated Deduction (CADE'09)*, Lecture Notes in Artificial Intelligence, pages 355–370, Montreal, Canada, Aug 2009. Springer.

[CJS+08] Iliano Cervesato, Aaron D. Jaggard, Andre Scedrov, Joe-Kai Tsay, and Christopher Walstad. Breaking and fixing public-key kerberos. *Inf. Comput.*, 206:402–424, February 2008.

[CLCZ10] Hubert Comon-Lundh, Véronique Cortier, and Eugen Zălinescu. Deciding security properties for cryptographic protocols. application to key cycles. *ACM Transactions on Computational Logic*, 11(2):1–42, 2010.

[CLN09] Cas J.F. Cremers, Pascal Lafourcade, and Philippe Nadeau. Comparing state spaces in automatic security protocol analysis. In Véronique Cortier, Claude Kirchner, Mitsuhiro Okada, and Hideki Sakurada, editors, *Formal to Practical Security*, volume 5458 of *Lecture Notes in Computer Science*, pages 70–94. Springer Berlin Heidelberg, 2009.

[Com] Gerald Combs. Wireshark. http://www.wireshark.org.

[Cre08] Cas J.F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In Aarti Gupta and Sharad Malik, editors, *Computer Aided Verification*, volume 5123 of *Lecture Notes in Computer Science*, pages 414–418. Springer Berlin Heidelberg, 2008.

[CS10] Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *International Conference on Financial Cryptography and Data Security*, FC, 2010.

[CS12] Véronique Cortier and Ben Smyth. Attacking and fixing helios: An analysis of ballot secrecy. *Journal of Computer Security*, 2012.

[CW12a] Véronique Cortier and Cyrille Wiedling. A formal analysis of the norwegian e-voting protocol. In *Principles of Security and Trust, POST*, volume 7215 of *Lecture Notes in Computer Science*, pages 109–128, Tallinn, Estonia, March 2012. Springer.

[CW12b] Véronique Cortier and Cyrille Wiedling. A formal analysis of the norwegian e-voting protocol. In Pierpaolo Degano and Joshua D. Guttman, editors, *Principles of Security and Trust, POST*, volume 7215 of *Lecture Notes in Computer Science*, pages 109–128. Springer Berlin Heidelberg, 2012.

[DKRS11] Stéphanie Delaune, Steve Kremer, MarkD. Ryan, and Graham Steel. A formal analysis of authentication in the tpm. In Pierpaolo Degano,

Sandro Etalle, and Joshua Guttman, editors, *Formal Aspects of Security and Trust, FAST*, volume 6561 of *Lecture Notes in Computer Science*, pages 111–125. Springer Berlin Heidelberg, 2011.

[DLMS99] Nancy A. Durgin, Patrick Lincoln, John C. Mitchell, and Andre Scedrov. Undecidability of bounded security protocols. In *Workshop on Formal Methods and Security Protocols*, FMSP'99, Trento, Italy, July 1999.

[DRS08] Stéphanie Delaune, Mark D. Ryan, and Ben Smyth. Automatic verification of privacy properties in the applied pi-calculus. In *Conferences on Privacy, Trust Management and Security*, volume 263 of *International Federation for Information Processing (IFIP)*, pages 263–278. Springer, 2008.

[DSHJ10] Nitish Dalal, Jenny Shah, Khushboo Hisaria, and Devesh Jinwala. A comparative analysis of tools for verification of security protocols. *Int'l J. of Communications, Network and System Sciences*, 3(10):779–787, 2010.

[DY81] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *Annual IEEE Symposium on Foundations of Computer Science*, 0:350–357, 1981.

[Eng] Tobias Engel. Locating mobile phones using signalling system 7. `http://events.ccc.de/congress/2008/Fahrplan/attachments/1262_25c3-locating-mobile-phones.pdf`. 25th Chaos Communication Congress (25C3).

[Ett09] Ettus. USRP. `http://www.ettus.com/products`, 2009.

[Fox97] Dirk Fox. IMSI-Catcher. *Datenschutz und Datensicherheit (DuD)*, 21:539–539, 1997.

[Gad] Great Scott Gadgets. HackRF. `http://greatscottgadgets.com/hackrf/`.

[GHB08] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453:779–782, 2008.

[GHT05] Andreas Grlach, Andreas Heinemann, and WesleyW. Terpstra. Survey on location privacy in pervasive computing. In Philip Robinson, Harald Vogt, and Waleed Wagealla, editors, *Privacy, Security and Trust within the Context of Pervasive Computing*, volume 780 of *The International Series in Engineering and Computer Science*, pages 23–34. Springer US, 2005.

[gnu06] GNU Radio. `http://gnuradio.org`, 2006.

[GRB12] Nico Golde, Kevin Redon, and Ravishankar Borgaonkar. Weaponizing femtocells: The effect of rogue devices on mobile telecommunications. In *Network and Distributed System Security Symposium, NDSS*, NDSS, 2012.

[GRBR13] Gurchetan S. Grewal, Mark D. Ryan, Sergiu Bursuc, and Peter Y. A. Ryan. Caveat coercitor: Coercion-evidence in electronic voting. In *IEEE Symposium on Security and Privacy*, SP '13, pages 367–381, Washington, DC, USA, 2013. IEEE Computer Society.

[Gut12] Joshua D. Guttman. State and progress in strand spaces: Proving fair exchange. *Journal of Automated Reasoning*, 48(2):159–195, 2012.

[H02] Hans Httel. Deciding framed bisimilarity. *Electronic Notes in Theoretical Computer Science*, (6):1–18, 2002.

[Her06] J. Herzog. Applying protocol analysis to security device interfaces. *IEEE Security Privacy*, 4(4):84–87, 2006.

[Int97] Lucas D. Introna. Privacy and the computer: Why we need privacy in the information society. *Metaphilosophy*, 28(3):259–275, 1997.

[KFD10] Ioannis Krontiris, Felix C. Freiling, and Tassos Dimitriou. Location privacy in urban sensing networks: research challenges and directions [security and privacy in emerging wireless networks]. *Wireless Communications, IEEE*, 17(5):30–35, October 2010.

[Kin10] Kineto Wireless Inc. official Unlicensed Mobile Access presentation webiste. `http://www.smart-wi-fi.com/`, June 2010.

[KKHK12] Denis Foo Kune, John Koelndorfer, Nicholas Hopper, and Yongdae Kim. Location leaks over the GSM air interface. In *Network and Distributed System Security Symposium, NDSS*, NDSS, 2012.

[KO06] Geir Koien and Vladimir Oleshchuk. Location privacy for cellular systems; analysis and solution. In *Privacy Enhancing Technologies Symposium, PET*, volume 3856, 2006.

[KR05] Steve Kremer and Mark D. Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *European Conference on Programming Languages and Systems*, ESOP'05, pages 186–200, Berlin, Heidelberg, 2005. Springer-Verlag.

[Kru09] John Krumm. A survey of computational location privacy. *Personal Ubiquitous Comput.*, 13(6):391–399, August 2009.

[KSR08] Petr Klus, Ben Smyth, and Mark D. Ryan. Proswapper, 2008.

[KT09] Ralf Küsters and Tomasz Truderung. Using proverif to analyze protocols with diffie-hellman exponentiation. In *Computer Security Foundations Symposium, CSF*, CSF, pages 157–171, Washington, DC, USA, 2009. IEEE Computer Society.

[KT11] Ralf Küsters and Tomasz Truderung. Reducing protocol analysis with xor to the xor-free case in the horn theory based approach. *J. Autom. Reason.*, 46(3-4):325–352, April 2011.

[Liu11] Jia Liu. A proof of coincidence of labeled bisimilarity and observational equivalence in applied pi-calculus, 2011.

[Low96a] Gavin Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, TACAs '96, pages 147–166, London, UK, UK, 1996. Springer-Verlag.

[Low96b] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using fdr. In *Tools and Algorithms for the Construction and Analysis of Systems*, TACAS, 1996.

[LTV10] Pascal Lafourcade, Vanessa Terrade, and Sylvain Vigier. Comparison of cryptographic verification tools dealing with algebraic properties. In Pierpaolo Degano and Joshua D. Guttman, editors, *Formal Aspects in Security and Trust*, volume 5983 of *Lecture Notes in Computer Science*, pages 173–185. Springer Berlin Heidelberg, 2010.

[MM] Sam May and Luca Melette. Distributed gsm security analysis. `https://program.sigint.ccc.de/fahrplan/system/attachments/21/original/120518.GSMMAP-SIGINT.pdf`. SIGINT 2012.

[Mod10] Sebastian Alexander Modersheim. Abstraction by set-membership: verifying security protocols and web services with databases. In *Conference on Computer and Communications Security, CCS*, CCS, pages 351–360, New York, NY, USA, 2010. ACM.

[MW04] Ulrike Meyer and Susanne Wetzel. A man-in-the-middle attack on UMTS. In *ACM Workshop on Wireless Security*, WiSe, 2004.

[NMa] Karsten Nohl and Luca Melette. Defending mobile phones. `http://events.ccc.de/congress/2011/Fahrplan/attachments/1994_111217.SRLabs-28C3-Defending_mobile_phones.pdf`. 28th Chaos Communication Congress (28C3).

[NMb] Karsten Nohl and Sylvain Munaut. Wideband gsm sniffing. `http://events.ccc.de/congress/2010/Fahrplan/attachments/1783_101228.27C3.GSM-Sniffing.Nohl_Munaut.pdf`.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Commun. ACM*, 21(12):993–999, December 1978.

[NS06] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *arXiv preprint arXiv:0610105*, 2006.

[NS09] Arvind Narayanan and Vitaly Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, SP '09, pages 173–187, Washington, DC, USA, 2009. IEEE Computer Society.

[Pag10] Chris Paget. Practical cellphone spying. Def Con 18 Hacking Conference, 2010.

[pat] `http://www.pathintelligence.com`. Path Intelligence Ltd. (2010) FootPath.

[PM08] Juan Carlos López Pimentel and Raul Monroy. Formal Support to Security Protocol Development: A Survey. *Computación y Sistemas*, 12:89 – 108, 09 2008.

[PR11] M Peters and Pieter Rogaar. A review of proverif as an automatic security protocol verifier, 2011.

[Pro] AVISPA Project. AVISPA protocol library. `http://www.avispa-project.org`.

[PVs] `http://www.markryan.eu/research/UMTS/`.

[RT03] Michaël Rusinowitch and Mathieu Turuani. Protocol insecurity with a finite number of sessions and composed keys is np-complete. *Theor. Comput. Sci.*, 299(1-3):451–475, April 2003.

[Rya11] Mark D. Ryan. Cloud computing privacy concerns on our doorstep. *Commun. ACM*, 54(1):36–38, January 2011.

[Shi09] Katie Shilton. Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection. *Commun. ACM*, 52(11):48–53, November 2009.

[SMCB12] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated analysis of diffie-hellman protocols and advanced security properties. In *Computer Security Foundations Symposium, CSF*, CSF, pages 78–94. IEEE, 2012.

[Str07] Daehyun Strobel. IMSI Catcher, 2007. Seminar Work, Ruhr-Universitat Bochum.

[Tom13] Tom Ritter and Doug DePerry and Andrew Rahimi. I Can Hear You Now: Traffic Interception and Remote Mobile Phone Cloning with a Compromised CDMA Femtocell. `https://www.blackhat.com/us-13/briefings.html`, july 2013. Blackhat.

[tra] `http://www.markryan.eu/research/mobile/`.

[WB90] Samuel D. Warren and Louis D. Brandeis. The right to privacy. *Harvard Law Review*, IV(5), 1890.

[Weh09] Dennis Wehrle. Open Source IMSI-Catcher für GSM. Master Thesis, October 2009.

[WFS+] Harald Welte, Holger Freyther, Dieter Spaar, Stefan Schmidt, Daniel Willmann, Jan Luebbe, Thomas Seiler, and Andreas Eversberg. OpenBSC. `http://openbsc.osmocom.org`.

[WMEoc] Harald Welte, Sylvain Munaut, Andreas Eversberg, and other contributors. OsmocomBB. `http://bb.osmocom.org`.

[ZdlR09] Jie Zhang and Guillaume de la Roche. *Femtocells: Technologies and Deployment*. John Wiley & Sons, Ltd, 2009.

[ZF05] Muxiang Zhang and Yuguang Fang. Security analysis and enhancements of 3GPP authentication and key agreement protocol. *IEEE Transactions on Wireless Communications*, 4(2):734–742, 2005.