PhD-FSTM-2023-041

The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 15/06/2023 in Esch-sur-Alzette

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG EN INFORMATIQUE

by

Semen Yurkov

Born on 30 April 1987 in Novokuznetsk (USSR)

ANALYSIS OF SMARTCARD-BASED PAYMENT PROTOCOLS IN THE APPLIED π-CALCULUS USING QUASI-OPEN BISIMILARITY

Dissertation defence committee

Dr. Peter Y. A. RYAN, Chairman

Professor, Université du Luxembourg

Dr. Ross James HORNE, Vice chairman

Postdoctoral researcher, Université du Luxembourg

Dr. Sjouke MAUW, Supervisor

Professor, Université du Luxembourg

Dr. Sebastian MÖDERSHEIM

Associate Professor, Danmarks Tekniske Universitet

Dr. Ioana BOUREANU

Professor, University of Surrey

"Today abstraction is no longer that of the map, the double, the mirror, or the concept. Simulation is no longer that of a territory, a referential being, or a substance. It is the generation by models of a real without origin or reality: a hyperreal. The territory no longer precedes the map, nor does it survive it. It is nevertheless the map that precedes the territory – precession of simulacra – that engenders the territory."

Jean Baudrillard

Abstract

Analysis of Smartcard-based Payment Protocols in the Applied π-calculus using Quasi-Open Bisimilarity

by Semyon YURKOV

Cryptographic protocols are instructions explaining how the communication between agents should be done. Critical infrastructure sectors, such as communication networks, financial services, information technology, transportation, etc., use security protocols at their very core to establish the information exchange between the components of the system. Symbolic verification is a discipline that investigates whether a given protocol satisfies the initial requirements and delivers exactly what it intends to deliver. An immediate goal of symbolic verification is to improve the reliability of existing systems – if a protocol is vulnerable, actions must be taken asap before a malicious attacker exploits it; a far-reaching goal is to improve the system design practices – when creating a new protocol, it must be proven correct before the implementation.

Properties of cryptographic protocols roughly fall into two categories. Either reachability-based, i.e. that a system can or cannot reach a state satisfying some condition, or equivalence-based, i.e. that a system is indistinguishable from its idealised version, where the desired property trivially holds. Security properties are often formulated as a reachability problem and privacy properties as an equivalence problem. While the study of security properties is relatively settled, and powerful tools like Tamarin and ProVerif, where it is possible to check reachability queries, exist, the study of privacy properties expressed as equivalence only starts gaining momentum. Tools like DeepSec, Akiss and, again, ProVerif offer only limited support when it comes to indistinguishability. This is partially due to the question of "What is an attacker capable of?" is not answered definitively in the second case.

The widely-accepted default attacker, when it comes to security, is the so-called Dolev-Yao attacker, which has full control of the communication network; however, there is no default attacker who attempts to break the privacy of a protocol. The capabilities of such an attacker are reflected in the equivalence relation used to define a privacy property; hence the choice of such relation is crucial.

This dissertation justifies a particular equivalence relation called quasi-open bisimilarity which satisfies several natural requirements. It has sound and complete modal logic characterisation, meaning that any attack on privacy has a practical interpretation; it enables compositional reasoning, meaning that if a privacy property of a system automatically extends to a bigger system having the initial one as a component, and, it captures the capability of an attacker to make decisions dynamically during the execution of the protocol.

We not only explain the notion of quasi-open bisimilarity, but we also employ it to study real-world protocols. The first protocol, UBDH, is an authenticated key agreement suitable for card payments, and the second protocol, UTX, is a smartcard-based payment protocol. Using quasi-open bisimilarity, we define the target privacy property of unlinkability, namely that it is impossible to link protocol sessions made with the same card and prove that it holds for UBDH and

UTX. The proofs that UBDH and UTX satisfy their privacy requirements to our knowledge are the first ones that demonstrate that a privacy property of a security protocol, defined as bisimilarity equivalence, is satisfied for an unbounded number of protocol sessions. Moreover, these proofs illustrate the methodology that could be employed to study the privacy of other protocols.

Acknowledgements

This dissertation started in September 2018, when I was in Georgia preparing for the SaToSS research seminar during my holiday. In the picture, you can see my laptop and the mount Kazbegi (5034m). At that time, my long-term friend Olga Gadyatskaya invited me to visit Luxembourg to assess my suitability as a doctoral researcher and to introduce me to Sjouke Mauw and Ross Horne, who were looking for a new colleague. I am glad it worked out and that Sjouke and Ross gave me the opportunity to make my way toward the degree under their supervision. Without these three people, this dissertation would have never happened.

If I could give myself from the past a single piece of advice, it would sound like this: "If it is not written down, it doesn't exist. Write down what you want to preserve." I learned this from Sjouke, whom I want to thank for his kindness and wisdom. I owe immense thanks to Ross; it was truly a heroic effort of his to teach me how to write a paper end-to-end.

I want to thank Ioana Boureanu and Peter Y. A. Ryan, members of my CET (comité d'encadrement de thèse) committee, for their guidance and patience. They were the first to see the majority of the material that went into this work, and without their insights, the dissertation would not be what it is now.

This dissertation is by no means a single-person effort. It is a product of the environment the University of Luxembourg and the wider research community have managed to build – from tiny internal events like brainstorms to summer schools and conferences. I thank every SaToSS group member who was always supportive and sharing with a dedicated credit to Sergiu Bursuc for his expertise in verification tools. I am also grateful for my teaching experience during the doctorate; I thank the students I taught since there is no better way to learn something than to explain it to others.

I want to thank the FNR (Fonds National de la Recherche), which Pride program has given me the freedom to manage the budget; the dissertation would not happen without their financial support.

Lastly, I would like to thank the smartest person I know, my wife Daria, for helping me not to overthink. It could have been a much harder journey without her.

Semyon Yurkov[1]

[1]Semen Yurkov is supported by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared

Contents

List of Figures

Chapter 1

Introduction

The majority of useful systems have a cryptographic (or, interchangeably, security) protocol at their core, a set of instructions that specifies how the communication between the components of the system should be done. Whether the system is a web service comprising servers and clients, or a payment system comprising payment cards, point-of-sale terminals, and banks, the participants of the communication aim at reaching certain goals. For instance, the client might want to be sure that the data she receives is indeed the data the server sends to her and that this data was not altered on the way due to the, e.g. noise in the communication channel; or, the bank issuing the card might want to guarantee that it is impossible to identify whether two transactions are made with the same card if all three participants honestly follow the protocol.

Symbolic verification studies whether a particular protocol satisfies a specific property under the assumption that cryptography is perfect. In a way, this is the study that answers the high-level question: "Is my protocol designed correctly?" If the answer is negative, it is not worth investigating such a protocol further or implementing it. Hence, when designing a new protocol, its symbolic verification is the first guard that the protocol should pass on its way to deployment. However, practice shows that protocols are deployed without any such verification. For instance, the EMV [emv11] payment protocol that payment systems such as Visa and Mastercard implement has only recently been symbolically verified [BST21b], resulting in attacks allowing to bypass the PIN for high-value purchases. Moreover, even "proven" protocols might be found vulnerable later due to the modelling issues, e.g. the famous Needham-Schroeder (NS) authentication protocol [NS78] was considered to be secure in the closed environment, where agents know each other's identities, while later in an enhanced model with anyone allowed to connect, Lowe found that the protocol is broken [Low96]. In this example, the attacker's capabilities were initially underestimated; hence, what an attacker can do affects the verification outcome, and it is important to identify the relevant attacker model. It is always safer to overestimate as if a property holds in a stronger setting, there is no attack in a weaker setting, while proving in a weaker setting leaves room for a more powerful realistic attacker, as the example with the Needham-Schroeder protocol teaches.

The concept of a labelled transition system is often used to describe the potential behaviour of the protocol and to formulate its properties that roughly fall into two categories.

- Reachability, i.e. the property holds if an attacker interacting with the system cannot force the system reaching a "bad" state where the property is violated.
- Indistinguishability, i.e. the property holds if an attacker interacting with the system cannot distinguish between the real-world system *impl* and the idealised system *spec*, where the target property definitely holds, written schematically as $impl \sim spec$.

In symbolic verification argot, the first category is informally called security properties which include secrecy and authentication, and the second category is called privacy properties which include anonymity and unlinkability. The theory of verifying reachability properties is well-established, and state-of-the-art tools like Tamarin [MSCB13] and ProVerif [B+01] allowing to reason about complex protocols automatically exist. At the same, the methodology of reasoning about the indistinguishability properties is still in development, as a story similar to the NS protocol has been recently repeated.

Indistinguishability relies on the notion of equivalence $\sim$ between two labelled transition systems that reflects the attacker's capabilities of distinguishing between the real-world and idealised systems [Del18, ACRR10, HM21, KR05, CS11]. In 2016 Hirschi, Baelde and Delaune proved that in the trace equivalence-based model [HBD16] in the BAC protocol used in biometric passports is unlinkable, it is impossible to identify whether two sessions are with the same document. However, later in 2019, a practical attack allowing to identify sessions with the same document was discovered in a stronger bisimilarity-based model [FHMS19] by Filimonov, Horne, Mauw, and Smith, thereby actualising the question of which equivalence notion reflects a realistic attacker. Though methods for checking trace equivalence and tools like DeepSec [CKR18] and Akiss [BAF08] also exist, the BAC study [FHMS19, HM21] demonstrates that, generally, trace equivalence underestimates the attacker's capabilities.

The above leads us to research questions we will aim to answer in this dissertation.

Research Question 1

Can we identify the requirements for an equivalence notion suitable for modelling indistinguishability properties of security protocols?

We aim at a small set of natural demands for an equivalence notion coming both from the verification perspective and from the practical considerations; for instance, it is already clear from this introduction that being finer than trace equivalence is among such demands.

Research Question 2

Can we identify a canonical equivalence notion satisfying the demands identified?

Moreover, we are interested in the explicit attacker's capabilities for distinguishing between the real and the idealised worlds such equivalence notion might capture.

Research Question 3

Can we reason effectively about security protocols using the equivalence identified?

Our goal is to demonstrate how to use the identified equivalence to define (privacy) properties of security protocols, express attacks and, more importantly, prove that a protocol in question indeed satisfied the desired property.

It is clear from the research questions above that our main ambition is to propose a method and apply it to some case study, thereby justifying it. Our method of reasoning about indistinguishability properties of protocols in this dissertation is based on the applied π-calculus formalism [ABF17], which is also used in the ProVerif tool we rely on to analyse reachability properties. The central concept of the applied π-calculus is a process, a concise way to define the labelled transition system corresponding to the protocol; hence the equivalence relation we are aiming at is the equivalence between two processes, and the attacker's capabilities are also expressed in terms of how an attacker can interact with the processes in an attempt to distinguish between the real and the idealised versions of the same protocol. Chapter 2 is dedicated entirely to the mathematical background.

Our case study is smartcard-based payments as such payments have a curious aspect – since the card is a passive device without any power source, a distant attacker can activate a contactless card and run the protocol remotely without the cardholder's awareness using an unauthorised device that is not connected to any payment infrastructure. The privacy dimension of this scenario is straightforward. If the card exposes its long-term identity to any device asking, it is easy to track the cardholder by the fact that she is holding the card in her pocket – enough to install antennas capable of activating the card at, e.g. doorways inside the building and observe where each card is. This is exactly the case with the current EMV protocol deployed worldwide since the card gives away its card number PAN. Surprisingly or not, in 2013, the developers of the EMV protocol, in their proposal of enhancing the privacy of payments [rfc12], underestimated the attacker's capabilities. They assumed an attacker that could only eavesdrop on communications but could not interact, which led to a Blinded Diffie-Hellman authenticated key establishment protocol that satisfies their privacy requirements in the weaker model but does not satisfy them in the stronger model as we explain in Chapter 3 where we apply the formalism developed in the first chapter. As in 2019, efforts to enhance the privacy of payments were officially abandoned [emv19] In Chapter 4, we take the opportunity to give a lesson in protocol design, i.e. we develop a smartcard-based payment protocol by identifying functional, security and privacy requirements, the realistic attacker model, and then verify both the security and privacy of the protocol using ProVerif and the methodology developed in this dissertation.

1.1 Contributions

Below we summarise the main contributions of this dissertation.

- We identify a concise set of requirements for the equivalence in the applied π-calculus suitable for modelling indistinguishability properties of cryptographic protocols. Firstly, such equivalence should have sound and complete modal logic characterisation, i.e. if $impl \not\sim spec$, then there is a formula ϕ that holds on one side but fails for the opposite, and if $impl \sim spec$ both sides satisfy the same formulas. Secondly, such equivalence should also be a congruence relation, i.e. if a property holds for a smaller system, it should also hold for a larger system containing the smaller one as a subsystem. Thirdly, the target equivalence should be some form of bisimilarity, i.e. an attacker should be able to make decisions dynamically during the execution.
- We present an equivalence in the applied π-calculus satisfying the requirements above, called quasi-open bisimilarity, which is the coarsest among such relations, and explain the attacker's capabilities it captures. The attacker behind quasi-open bisimilarity can "access" different universes where the execution takes place by manipulating variables in his control, can compare two states using the messages output so far by honest participants, and can make dynamic decisions.
- We analyse unlinkability, i.e. the impossibility of determining whether two payments were made with the same card, of the Blinded Diffie-Hellman (BDH) authenticated key establishment protocol proposed by the developers of the EMV payment protocol to enhance the privacy of payments. We propose an unlinkability definition based on the notion of quasi-open bisimilarity that accounts for active attackers and employs the compositionality aspect allowing us to reduce the amount of work needed for verification. We demonstrate a modal logic formula representing an attack on the unlinkability of BDH. Further, we propose a fixed version of BDH called UBDH (Unlinkable BHD) and verify it against our definition. The proof of the unlinkability of the UBDH key agreement protocol is the first published proof of a bisimilarity-based indistinguishability problem for an unbounded number of protocol sessions.
- We identify the functional, privacy and security requirements for an unlinkable smartcard-based payment protocol, we design such a protocol called UTX (Unlinkable Transactions), and prove it satisfies the identified requirements yet again providing a detailed proof for the unbounded case of indistinguishability between the real-world implementation and the idealised specification of the protocol comprising several roles.

1.2 Layout and the author's contribution

The dissertation comprises three chapters.

- Chapter 2 is dedicated to the background. In the introduction to the chapter, we explain three main requirements that an equivalence suitable for reasoning

about protocols should satisfy. Then we present what constitutes the applied π-calculus: the message theory axiomatising cryptographic operations, process syntax and the associated labelled transition system. Next, we introduce the notion of quasi-open bisimilarity and the surrounding concepts that include the "accessibility" mechanism an attacker may use to consider different universes where the execution of the protocol may take place. After that, we state the main results about quasi-open bisimilarity, i.e. that it is sound and complete with respect to certain modal logic, thereby there is always a formula describing attack whenever the equivalence fails; that it is preserved in any context, thereby we can use compositional reasoning and reduce the amount of work needed for verification; and that it is the coarsest bisimilarity congruence for the applied π-calculus making quasi-open bisimilarity the canonical choice among the relations satisfying our requirements. The presented material contains a variety of examples illustrating each concept introduced. The chapter is based on the following work.

  - *Compositional Analysis of Protocol Equivalence in the Applied π-Calculus Using Quasi-open Bisimilarity* by Horne, Mauw, and Yurkov published in 2021 in the proceedings of International Colloquium on Theoretical Aspects of Computing (ICTAC) conference.
  - *Whenever a privacy property fails, a formula describes an attack: A complete and Compositional Verification Method for Applied π-calculus* by Horne, Mauw, and Yurkov published in 2023 in Theoretical Computer Science (TCS) journal.

  It should be made clear that the author of this dissertation neither invented quasi-open bisimilarity, nor proved its properties, yet he has contributed a few technical results (e.g., Theorem 4, Sec. 2.2.6) that made it possible to publish main results (Theorems 1 - 3) established by Ross Horne.

- In Chapter 3, we investigate EMV, a payment method accepted worldwide. We introduce the main stages of the EMV transaction, discuss the security and privacy threats of EMV and demonstrate an insecure version of the EMV protocol allowed by the EMV standard, a set of technical documents specifying how the cards and the terminals should be implemented. Then we study the Blinded Diffie-Hellman (BDH) authenticated key establishment protocol proposed by the developers of EMV to encrypt the communication between the card and the terminal. We apply the theory developed in Chapter 2 to show that the BDH protocol does not satisfy the official anti-tracking requirement in the presence of a realistic active attacker that can force the card to interact using an unauthorised device such as an NFC-capable smartphone. We finally propose a minor fix to the BDH protocol and verify that our fix indeed brings the desired anti-tracking in an enhanced attacker's model without breaking the initial functional BDH requirement of providing authentication. The chapter is based on the following work.

  - *Unlinkability of an Improved Key Agreement Protocol for EMV 2nd Gen Payments* by Horne, Mauw, and Yurkov published in 2022 in the proceedings of The Computer Security Foundations Symposium (CSF) conference.

- In Chapter 4, we extract from the existing EMV standard the set of functional and security requirements that smartcard-based payments should satisfy and introduce a new privacy requirement, the unlinkability of payments in the presence of active attackers. We design a new protocol, UTX explaining how the sensitive information is hidden from the most powerful attacker, an attacker pretending to be a terminal. Finally, we verify that UTX satisfies our requirements. The chapter is based on the following work.

  - *Designing an Unlinkable Smartcard-based Payment Protocol* by Bursuc, Horne, Mauw, and Yurkov rejected at the 2023 Symposium on Security and Privacy (S&P) conference, in preparation for the submission to the 2023 Computer and Communications Security (CCS) conference.

- In Chapter 5, we conclude the dissertation by revising the research questions posed in the introduction and outlining directions for future work.

Chapter 2

Background: quasi-open bisimilarity

In this chapter, we identify the requirements for an equivalence relation suitable for modelling indistinguishability properties of security protocols, we then give a state-of-the-art formulation of the applied π-calculus [AF01], a language for modelling concurrent processes and their interactions, and finally, we define the equivalence relation that satisfies our requirements called *quasi-open bisimilarity*.

An equivalence-based approach can be employed to formulate both security and privacy properties of cryptographic protocols, and, as Ryan and Schneider explain in [RS99, RSG+01] there is no universally "correct" notion of equivalence that suits all contexts and applications. Hence the equivalence we discuss in this dissertation is the correct one for the requirements we identify below. We are also clear from the start that the default application for the equivalence notion we introduce here is privacy; thus, by property, we always mean privacy property if not specified otherwise.

To start discussing requirements, we introduce some context by presenting the running example of indistinguishability property in this dissertation, a privacy property called *unlinkability*. Paraphrasing the definition given in Common Criteria for Information Technology Security Evaluation (ISO/IEC 15408) [cc17], the protocol is unlinkable if an attacker cannot determine if two sessions of the protocol were executed with the same user. To illustrate this definition, let us consider a protocol in which a document carries a digital certificate issued by a trusted authority and presents this certificate to a reader in different locations. Protocols such as bicycle-sharing systems, COVID-19 green pass, access cards, and smartcard payments are instances of such protocol.

Consider two (wireless) sessions made with the same document depicted in blue in the left part of Fig. 2.1. If an attacker observing the protocol sessions can link them, they can track the document owner. On the contrary, if an attacker cannot link sessions, i.e. a document executing the protocol in each session appears to an attacker as a new document that an attacker has never seen before, the tracking is impossible, and we say that the protocol is unlinkable. This impossibility of linking sessions is manifested as the equivalence between the protocol in question, where the same document can be used twice, and the idealised protocol, where a document can participate in one session at most. The idealised situation is depicted on the right part of Fig. 2.1: in the first session, an attacker observes the blue document, and in the second session, the same document appears as green.

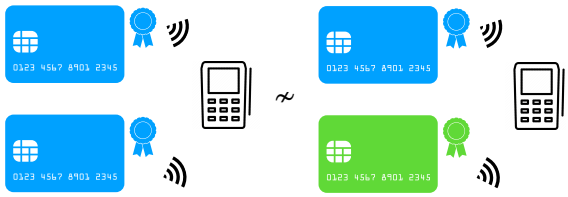


Figure 2.1: If an attacker can detect, that the same document has been used in different sessions, unlinkability fails.

The pattern of a protocol being equivalent to some idealised version is the essence of defining property as an indistinguishability problem, and we can already start introducing the related terminology. We will call the protocol in question *the implementation* or *the system*, while its idealised version, which specifies how the protocol should appear, *the specification*.

Still, we are far from the formal definition of unlinkability. Firstly, we need to model the system and the specification in some formal language; hence the definition is always protocol-specific. Secondly, we need to make the equivalence notion precise. While specifying the protocol defines how honest agents act and how the whole system operates, the equivalence notion captures the capabilities of an attacker trying to distinguish between the real world and the idealised situations making the equivalence part of the threat model.

Imagine that the reader emits the error message whenever it detects that the presented document is "wrong". Hence, an attacker observing that there was no error message during a protocol session may use this observation to infer that the protocol run has been successful and the individual holding the document received some sensitive data demonstrated, for instance, on the reader's display. This simple thought experiment illustrates that it is possible to leak some personal identifiable information without leaking secret data; moreover, this leak is enabled by the observation of a *failure* of an action. The abilities of an attacker as such form the set of attack strategies an attacker can execute to distinguish between the implementation and the specification of the protocol. Several equivalences have already been proposed to define privacy properties like unlinkability. The list includes trace equivalence [BNP01], early bisimilarity [ABF17], failure traces [Hoa85], and failure simulation [vG93], which all capture different attacker's capabilities and correspond to different attack strategies.

The fact that equivalence in Fig. 2.1 fails reflects that an attacker can link sessions by determining that a document has been used twice and, hence, follow the movements of the document holder as the same document can be observed in different locations. We require that this non-equivalence between the implementation and the specification have a precise meaning. There should be a strategy for distinguishing the protocol from its idealised specification. The existence of a strategy whenever a property fails is crucial for the verification outcome and, however, is not always guaranteed. In Sec. 2.2.5, we will encounter the example where the equivalence failure does not correspond to an attack; hence the interpretation of the verification outcome is unclear.

A distinguishing strategy can be described by a formula ϕ in a suitable modal logic, where the formula holds for the implementation but does not hold for the

specification, as we illustrate in Fig. 2.2. Hennessy and Milner [HM85] were the first who proposed the design pattern of using a modal logic to describe distinguishing strategies. Since then, it has been applied to characterise many different equivalences for various process models [MPW93, HALT18, AHT21, PBE+21, VG01, Sim04, DNV95, Dob05, BC14, DP03].

Figure 2.2: Whenever a property fails, a formula ϕ describes an attack.

In the context of privacy problems expressed as an equivalence, we require that all distinguishing formulas are attacks on the property, and conversely, when the property fails, the modal logic formula can always describe an attack. We can now concisely formulate the first demand for our target equivalence notion. *The equivalence used for modelling indistinguishability properties should have a sound and complete modal logic characterisation*.

Let us continue with the unlinkability example in Fig. 2.1. It could be the case that the electronic document participating in the protocol might be forced into a session by an untrusted device, for instance, when the document is a proximity card that can be powered up from a distance. We require that even in this case, the unlinkability of the protocol should be maintained, and it should still be impossible to track the document owner. Such an untrusted device to the document is simply the environment that is able to communicate with it; hence the reduced system and the corresponding reduced unlinkability scheme would require considering a protocol without readers as presented schematically in the left part of Fig. 2.3. It is also natural to require that bringing back honest readers to a picture does not affect the unlinkability of the protocol. The relation between smaller and larger systems is established by what is called context, a term we explain in Section 2.1.2. This relation is protocol-specific, and its existence is not always guaranteed; however, if it would be possible to reduce the definition to a smaller system, it can reduce the amount of work needed for the verification. Thus our second demand for the target equivalence notion is formulated as follows. *The equivalence used for modelling indistinguishability properties should also be a congruence relation*, meaning that the equivalence is preserved by any context.

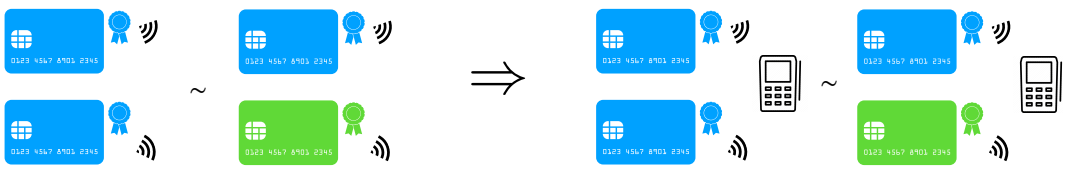


Figure 2.3: Compositionality: a property automatically extends to a larger system.

Finally, we should consider where approximately the target equivalence sits on the finer-coarser scale. The unlinkability analysis of the BAC protocol used in ePassports [FHMS19] demonstrates that such equivalence must allow an attacker to make decisions dynamically during the execution of the protocol, making the

relation some form of bisimilarity, which is finer than trace equivalence[1]. Hence, our third demand for the suitable equivalence is simple *it must be some form of bisimilarity*. Notice that verifying a property against a finer equivalence is safe, as a property formulated via coarser equivalence would automatically hold. The problem arises when we get too fine, such that the equivalence failure happens for technical reasons while there is no practical attack behind this failure.

With this preliminary discussion, we can identify the area in Fig. 2.4 where the target equivalence lives. In this picture blue area contains early labelled bisimilarity, which is sufficiently fine to accommodate the dynamic decisions of an attacker, but it is not a congruence. The red area contains early bisimilarity, which is a congruence, but is too fine to reflect a realistic attacker's capabilities, hence our ideal equivalence should live in the intersection of these two regions.

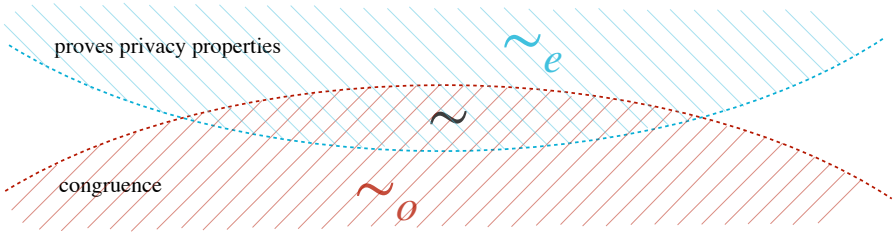


Figure 2.4: Bisimilarities for the applied π-calculus: $\sim_e$ is (early) labelled bisimilarity and $\sim_o$ is open bisimilarity.

In what follows, we pick and explain a single point in the identified region, quasi-open bisimilarity, which is the coarsest bisimilarity congruence for the applied π-calculus. This bisimilarity covers a wide range of attack strategies and admits desirable compositional approach, thereby being suitable to reason about privacy properties of security protocols effectively. In Sec. 2.1, we introduce the applied π-calculus, a formalism used to define and reason about labelled transition systems, and then, in Sec. 2.2, we explain quasi-open bisimilarity, an equivalence notion for the applied π-calculus, that satisfies the requirements we have identified above.

2.1 Applied π-calculus

In this dissertation, we focus on the applied π-calculus [AF01, ABF17] due to its prominent status as a model for cryptographic protocols. The variety of tools using dialects of the applied π-calculus includes DEEPSEC [CKR18], Akiss [CCCK16], Sapic [KK16], SAT-Equiv [CDD17], SPEC [TNH16], and ProVerif [Bla16], one of the most popular verification tools used to analyse, e.g. commitment protocols [CSS15], voting protocols [KR05], distance-bounding protocols [CdRS18], etc.

The central concept of the applied π-calculus is a process. Processes are used to capture the behaviour of honest parties executing the protocol. Processes interact by outputting or receiving messages using channels. Both messages and channels can be message terms subject to some message theory, hence below we first describe

[1] A parallel approach of proving that a particular class of protocols indeed allows employing trace equivalence was recently taken by Baelde, Delaune, and Moreau to analyse stateful protocols [BDM20].

a sample message theory, then we explain process syntax, and, finally, introduce operational semantics which defines how the system moves from state to state.

2.1.1 Message theory

A message theory defines how the messages are formed and axiomatises cryptographic operations. To provide meaningful examples we use a particular message theory local to this chapter defined in Fig. 2.5 which includes public key cryptography, one-way hash function, and tuples. The equational theory E at the bottom of Fig. 2.5 defines the first and second projection functions, and the decryption of an encrypted message using the right secret key. The last equation is a test of whether the message M is the encryption with $\mathtt{pk}(M)$ which we require to hold for any M, making it impossible to detect failed or successful decryption. However, in this theory, it is possible to detect whether the message is a pair by checking if $\langle \mathtt{fst}(M), \mathtt{snd}(M) \rangle =_E M$ holds.

$$
\begin{array}{rll}
M, N, K ::= & x & \text{variable} \\
\mid & \mathtt{pk}(M) & \text{public key} \\
\mid & \mathtt{h}(M) & \text{hash} \\
\mid & \langle M, N \rangle & \text{tuple} \\
\mid & \mathtt{aenc}(M, N) & \text{asymmetric encryption} \\
\mid & \mathtt{adec}(M, N) & \text{asymmetric decryption} \\
\mid & \mathtt{fst}(M) & \text{left} \\
\mid & \mathtt{snd}(M) & \text{right}
\end{array}
$$

$$
\begin{aligned}
\mathtt{fst}(\langle M, N \rangle) &=_E M \\
\mathtt{snd}(\langle M, N \rangle) &=_E N \\
\mathtt{adec}(\mathtt{aenc}(M, \mathtt{pk}(K)), K) &=_E M \\
\mathtt{aenc}(\mathtt{adec}(M, K), \mathtt{pk}(K)) &=_E M
\end{aligned}
$$

Figure 2.5: The applied π-calculus can be instantiated with any message theory.

Further examples of message theories can also be used to instantiate the applied π-calculus and include subterm convergent theories [AC06], blind signatures and homomorphic encryption [BCD14], locally stable theories with inverses [AFN17]. In further chapters we will consider theories that accommodate Diffie-Hellman handshake.

2.1.2 Process syntax in the applied π-calculus

The applied π-calculus can be instantiated with any message theory like the one given in Fig. 2.5 and generalises π-calculus [MPW92] in a way that channels and messages can be any message terms instead of pure variables, hence the syntax is similar. There are two main constructions: processes, which represent actions that could be taken, and extended processes, which represent states. In this subsection we focus on processes; extended processes are explained in the next subsection.

We start with processes and their syntax presented in Fig. 2.6. A process can either contain no actions, i.e. being deadlocked, or send and receive messages on a

$$
\begin{array}{rll}
P, Q ::= & 0 & \text{deadlock} \\
| & \overline{M}\langle N\rangle.P & \text{send} \\
| & M(y).P & \text{receive} \\
| & [M = N]P & \text{match} \\
| & [M \neq N]P & \text{mismatch} \\
| & \nu x.P & \text{new} \\
| & P \mid Q & \text{parallel} \\
| & P + Q & \text{choice} \\
| & !P & \text{replication}
\end{array}
$$

Figure 2.6: A syntax for processes.

dedicated channel. Notice, since the receive action has not yet been taken, the process $M(y).P$ contains variable y that can be a parameter in the process P defining the next actions. The processes we consider admit conditional branching reflecting `if–then–else` construct, common for many programming languages. In the syntax `if` branch corresponds to the match guard and the `else` branch to the mismatch guard. During the execution, processes can generate new or fresh variables using the ν operator and use them in future actions. These variables model private information such as nonces, keys, secret channels, etc. Finally, processes can run in parallel, contain unconditional branching, and be replicated, i.e. to run in parallel with itself infinitely many times. Later in the text we will encounter process schemes which is simply a regular process, parametrised by the finite number of parameters.

Throughout the paper, we use several conventions. We write $P\pi$, when some transition π is available after execution of all actions in P. We do not make a distinction between $\nu x_1.\nu x_2.P$ and $\nu x_1.(\nu x_2.P)$ and typically write $\nu x_1, x_2.P$. We use the symbol $\triangleq$ to define a process and to improve readability we introduce the following abbreviations.

$$
\begin{aligned}
&\texttt{let}\, x := M \,\texttt{in}\, P \;\triangleq\; P\{^{M}/_{x}\} \\
&\texttt{if}\, M = N \,\texttt{then}\, P \,\texttt{else}\, Q \;\triangleq\; [M = N]P + [M \neq N]P
\end{aligned}
$$

We also use pattern matching convention which is clear from the context. For instance we have the following.

$$
\texttt{let}\, \langle x_1, x_2\rangle = M \,\texttt{in}\, P \triangleq P\left\{^{\texttt{fst}(M),\texttt{snd}(M)}/_{x_1,x_2}\right\}
$$

In the process syntax, we have two examples of binding of a variable: when the variable is fresh $\nu x.P$ or when it represents the input $M(y).P$, hence variables x and y are called bound variables. If the variable is not bound, we call it free or open. We denote as $\mathrm{fv}(T)$ the set of free variables in a term T and call a term T ground whenever $\mathrm{fv}(T) = \emptyset$. Since often we talk about more than several messages, we use vector notation to denote respectively the list of variables $\vec{x}$ of messages $\vec{M}$. Whenever such lists are involved in set-theoretic operations, we treat $\vec{x}$ and $\vec{M}$ as unordered sets of elements.

Functions from some finite set of variables to message terms called substitutions, or, historically, active substitutions, play important role in the applied π-calculus. Firstly, substitutions are applied to processes when a receive action has been taken to replace the bound variable with the actual message. Secondly, a substitution represents the messages on the network available to an attacker as we explain below when we discuss extended processes. We reserve Greek letters σ, ρ, and θ for substitutions and write $x\sigma$ when the substitution σ is applied to the variable x. When we want to give a substitution explicitly we write $\sigma := \left\{ {}^{\vec{M}}/_{\vec{x}} \right\}$ and call $\vec{x}$ the domain and $\vec{M}$ the range of σ. A substitution σ being applied to a process P, written as $P\sigma$ results in the replacement of any free occurrence of the variables from the domain of σ with the messages from its range. The word free is crucial here as substitutions must be capture-avoidant: if a bound variable x appears in the range of the substitution, it must be preliminarily renamed to avoid the name clash. Such renaming of bound variables is known as α-conversion [SW01b] and is straightforward to perform as we explain in the following example.

Let us compute $a(x).\overline{a}\langle \mathtt{aenc}(\langle x, y\rangle, k)\rangle \left\{ {}^{\mathtt{h}(x)}/_{y} \right\}$. Since x is a bound variable that appears in the range of the substitution we should rename it using α-conversion before we apply the substitution. Let us choose a "safe" name, s.t. it does not belong to the set of free variables in the range of σ, i.e., is not x, and, does not belong to the set of free variables of $\overline{a}\langle \mathtt{aenc}(\langle x, y\rangle, k)\rangle$, i.e., is not among $\{a, x, y, k\}$. For instance, z meets these conditions, hence we replace x with z in the process and we are left with $a(z).\overline{a}\langle \mathtt{aenc}(\langle z, y\rangle, k)\rangle \left\{ {}^{\mathtt{h}(x)}/_{y} \right\}$ evaluated as $a(z).\overline{a}\langle \mathtt{aenc}(\langle z, \mathtt{h}(x)\rangle, k)\rangle$.

Below we will often encounter a situation when a variable should not belong to a set of variables. We generalise this concept in the following.

Definition 1 (fresh)**.** *A set of variables $\vec{x}$ is fresh for a set of variables $\vec{y}$ whenever $\vec{x} \cap \vec{y} = \emptyset$; $\vec{x}$ is fresh for a given term P, whenever $\vec{x}$ is fresh for* $\mathrm{fv}(P)$*; $\vec{x}$ is fresh for a given substitution σ whenever $\vec{x}$ is fresh for* $\mathrm{dom}(\sigma)$*, and, for all $y \notin \vec{x}$, we have $\vec{x}$ is fresh for $y\sigma$ (that is σ contains no variable from $\vec{x}$ in the domain and in the range). Notation: $\vec{x} \# \vec{y}$, $\vec{x} \# P$, $\vec{x} \# \sigma$.*

Finally, we introduce the notion of a context, which is a process with a dedicated placeholder for another process. We denote such placeholder with a dot, and the context is denoted as $\mathcal{C}\{\cdot\}$. E.g., if $\mathcal{C}\{\cdot\} \triangleq \nu k.\,(a(x).\overline{a}\langle \mathtt{aenc}(\langle x, y\rangle, k)\rangle \mid !\{\cdot\})$, we have $\mathcal{C}\{P\} \triangleq \nu k.\,(a(x).\overline{a}\langle \mathtt{aenc}(\langle x, y\rangle, k)\rangle \mid !P)$ for some process P.

2.1.3 The semantics of the applied π-calculus

In order to describe how the system can evolve from one state into another we first introduce how the states are presented. To define the state of a process we use the so-called extended process $\nu\vec{x}.(\sigma \mid P)$. The syntax is given in Fig. 2.7.

An extended process $\nu\vec{x}.(\sigma \mid P)$ comprises the private values $\vec{x}$ generated during the execution of the protocol, a substitution σ used to record messages already output on the network, and the process P that defines future actions the system could take. Consider the extended process $\nu s.\left(\left\{ {}^{\mathtt{pk}(s), M}/_{u_1, u_2} \right\} \mid a(z) \right)$ as an example. It consists of the fresh secret key s, the messages $\mathtt{pk}(s)$ and M which has been output in the past (the domain of the substitution, u_1 and u_2, records labels on

Extended processes:

$$A ::= \sigma \mid P \mid \nu x.A$$

Actions on labels:

$$\pi ::= \tau \mid \overline{M}(z) \mid M\,N$$

Figure 2.7: A syntax for extended processes and transition labels.

these messages; an attacker can only access messages on the network using labels), and the input action $a(z)$, that is not yet taken. The part of the extended process $\nu\vec{x}.(\sigma \mid \_)$ consisting of the private values $\vec{x}$ and the substitution σ is traditionally referred as a frame and it represents the static snapshot of the system at some stage of the execution of the protocol. Whenever we talk about the domain of the frame we mean $\mathrm{dom}(\sigma)$.

When describing the state of the system we require it to be presented in normal form. We say that an extended process $\nu\vec{x}.(\sigma \mid P)$ is in normal form if $\mathrm{dom}(\sigma)$ is fresh for $\vec{x}$, $\mathrm{fv}(P)$, $\mathrm{fv}(y\sigma)$ for any variable y, meaning that σ is idempotent ($\sigma \circ \sigma = \sigma$) and σ are fully applied to P. For instance, $\nu n.\left(\left\{{}^{n,\mathtt{aenc}(u,\mathtt{pk}(k))}/_{u,v}\right\} \mid \nu k.\overline{a}\langle\langle k, u\rangle\rangle\right)$ is not in the normal form, since the substitution is not idempotent and the process refers the variable u appearing in the domain of the substitution; while the process $\nu n.\left(\left\{{}^{n,\mathtt{aenc}(n,\mathtt{pk}(k))}/_{u,v}\right\} \mid \nu k.\overline{a}\langle\langle k, n\rangle\rangle\right)$ satisfies conditions of the normal form.

Finally, we present transition rules in Fig. 2.9 to complete the description of the labelled transition system associated with an extended process representing the initial state. This transition system is early due to the way inputs are treated, i.e. variables representing input should be substituted immediately; and this transition system is open, as it does not assume that free variables are ground names unless stated so explicitly in the name environment $\vec{z}$ to the left of the transition relation. Rules are defined on extended processes in the normal form, transitions are presented as labelled arrows, where the label π defines the type of the transition: input $M\,N$ or output $\overline{M}(x)$ on the channel M, or a silent internal action τ. The set of possible transitions that a system can make from the state $\nu\vec{x}.(\sigma \mid P)$ is defined by actions available in P. To avoid potential name clash and to keep track of the private information we introduce the set of names of the label $\mathrm{n}(\pi)$, and the set of bound names of the label $\mathrm{bn}(\pi)$ which we also define in Fig. 2.9.

$$\vec{z}:\ \nu\vec{x}.(\sigma \mid P) \xrightarrow{\overline{M}(u)} \nu\vec{x}, \vec{w}.\left(\left\{{}^{K}/_{u}\right\} \circ \sigma \mid Q\right)$$

$$\vec{z}:\ \nu\vec{x}.(\sigma \mid P) \xrightarrow{M\,N} \nu\vec{x}, \vec{w}.\left(\sigma \mid Q\left\{{}^{N\sigma}/_{y}\right\}\right)$$

Figure 2.8: The anatomy of output and input transitions.

We illustrate the general form of input and output transitions in Fig. 2.8, where the annotations highlight key elements and their role.

(A) A set of variables which should be treated as ground names that are used to enable inequalities.

(B) Bound variables of the state (extended process) that cannot be observed directly by attackers.

(C) A representation of message previously output and intercepted by an attacker that may be used to construct inputs.

(D) A process defining actions to be performed in the future, including the current transition.

(E) A channel on the label that cannot refer to any bound private variables, yet *can* use variables from $\mathrm{dom}(\sigma)$.

(F) Variables representing names that are extruded by the transition and that may appear in K.

(G) An input message that *can* refer to the aliases of messages recorded in $\mathrm{dom}(\sigma)$.

(H) A fresh alias for an output message that is recorded in the domain of the active substitution.

(I) The actual message being output that can be accessed indirectly using the alias above.

(J) The messages already output in the past are updated with a new message output in this transition.

(K) A process defining actions to be taken after the transition has been taken.

(L) The message that is being input in which aliases are replaced by messages according to σ.

(M) A bound variable representing the input is replaced by the message input.

We can summarise the detailed description of input and output transitions above as follows. The input action requires the substitution of a bound variable representing input with a message term; the output action requires recording an alias for a message appearing on the network in the active substitution part of the extended process. Several examples of proof trees justifying transitions a particular (extended) process can make and demonstrating how the rules from Fig. 2.9 work can be found in Sec. 2.2.6 of the current chapter and in the proof of Theorem 7 of the Chapter 3.

There are situations where a name clash is handled by the conventions we have described above. Recall that substitutions are always capture-avoidant, therefore before taking a transition it might be necessary to apply α-conversion first. For example the state $\nu n, k.\ \left(\left\{{}^{\{n\}_k}/_{u}\right\} \mid c(x).P\right)$ can make a transition $c\,n$ since the next available action is the input on the channel c. However, because n is bound by the

new name binder ν, there is a name clash when the substitution is applied to P in the resulting state. Hence, we should rename n to a safe name, e.g. z in the initial extended process, and only then execute the transition which results in the extended process $\nu z, k \left(\left\{ {}^{\{z\}_k}/_{u} \right\} \mid P\{{}^{z,n}/_{n,x}\} \right)$.

Mismatch is intuitionistic. The situation where the next available action is guarded by mismatch (inequality) requires special care. The main insight from the work of Horne et al. [HALT18] where they study a congruence relation in the version of the π-calculus extended with mismatch is that mismatch should be preserved under substitution. Consider the process $\mathtt{if}\, x \neq \mathtt{h}(y)\ \mathtt{then}\, a(r)$ which can execute the input action $a(r)$ if $\{{}^{y}/_{x}\}$ is applied, but cannot if we apply $\left\{ {}^{\mathtt{h}(y)}/_{x} \right\}$, hence neither $x = \mathtt{h}(y)$ nor $x \neq \mathtt{h}(y)$ holds until there is enough information about the environment s.t. we can once and forever decide between two options. In the next section, where we define our equivalence notion, the reader will see, how an attacker can systematically consider all possible universes by clarifying the environment and, in turn, evaluating the process unambiguously. To define the negation of the equation between two messages that enjoys the property of being preserved under substitution we introduce the following definition.

Definition 2 (entails)**.** *The entailment $\vec{z} \models M \neq N$ holds if there is no substitution σ s.t. $\vec{z} \# \sigma$ and $M\sigma =_E N\sigma$.*

This definition precisely captures that two message terms are distinct if there is no universe where they are equal. To give an example, the entailment $\emptyset \models x \neq \mathtt{h}(x)$ holds since there is no substitution unifying both sides of the inequality, hence there is enough information to resolve such guard. On the other hand, the entailment $\emptyset \models x \neq h(y)$ does not hold, as the substitution $\left\{ {}^{\mathtt{h}(y))}/_{x} \right\}$ unifies both sides, meaning that there is a universe where both sides are equal and there is insufficient information to decide they are not equal. Let us clarify, or, extend, the environment by declaring y private. In that case the entailment $y \models x \neq \mathtt{h}(y)$ holds since the most general unifier $\left\{ {}^{h(y)}/_{x} \right\}$ has y in the range, hence is not fresh for y – anyone who has the power to influence variable x, cannot make x equal to $h(y)$ without access to y.

2.2 Quasi-open bisimilarity

This section is dedicated to the notion of quasi-open bisimilarity, an equivalence for the applied π-calculus. This equivalence generalises a similar notion for the core π-calculus [SW01a, HALT18] and satisfies the requirements we have identified in the introduction to this chapter. Processes in this dissertation are used primarily to describe the protocol's behaviour in the real-world and the idealised situation. As we have described above, a process defines labelled transition system. Abstractly, two processes are equivalent if the associated labelled transition systems are equivalent, i.e. there is a pairing between the states (extended processes) of the labelled transition systems that an attacker cannot leave using their capabilities. We start this section with an informal description of what an attacker can do in order to

$$\frac{M\sigma =_E K}{\vec{z}\colon \sigma \mid K(x).P \xrightarrow{M\,N} \sigma \mid P\left\{{}^{N\sigma}/_{x}\right\}}\ \textsc{Inp} \qquad \frac{x \# M, N, P, \sigma, \vec{z} \qquad M\sigma =_E K}{\vec{z}\colon \sigma \mid \overline{K}\langle N\rangle.P \xrightarrow{\overline{M}(x)} \left\{{}^{N}/_{x}\right\} \circ \sigma \mid P}\ \textsc{Out}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\pi} A}{\vec{z}\colon \sigma \mid P + Q \xrightarrow{\pi} A}\ \textsc{Sum-l} \qquad \frac{\vec{z}\colon \sigma \mid Q \xrightarrow{\pi} A}{\vec{z}\colon \sigma \mid P + Q \xrightarrow{\pi} A}\ \textsc{Sum-r}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\pi} A \qquad M =_E N}{\vec{z}\colon \sigma \mid [M = N]P \xrightarrow{\pi} A}\ \textsc{Mat} \qquad \frac{\vec{z}\colon \sigma \mid P \xrightarrow{\pi} A \qquad \vec{z} \models M \neq N}{\vec{z}\colon \sigma \mid [M \neq N]P \xrightarrow{\pi} A}\ \textsc{Mismatch}$$

$$\frac{\vec{z}, x\colon \sigma \mid P \xrightarrow{\pi} B \qquad x \# \vec{z}, \sigma, \mathrm{n}(\pi)}{\vec{z}\colon \sigma \mid \nu x.P \xrightarrow{\pi} \nu x.B}\ \textsc{Extrude} \qquad \frac{\vec{z}, x\colon A \xrightarrow{\pi} B \qquad x \# \vec{z}, \mathrm{n}(\pi)}{\vec{z}\colon \nu x.A \xrightarrow{\pi} \nu x.B}\ \textsc{Res}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\pi} \nu\vec{x}.(\theta \mid R) \quad \vec{x}, \mathrm{bn}(\pi) \ \# Q}{\vec{z}\colon \sigma \mid P \mid Q \xrightarrow{\pi} \nu\vec{x}.(\theta \mid R \mid Q)}\ \textsc{Par-l} \qquad \frac{\vec{z}\colon \sigma \mid Q \xrightarrow{\pi} \nu\vec{x}.(\theta \mid R) \quad \vec{x}, \mathrm{bn}(\pi) \ \# P}{\vec{z}\colon \sigma \mid P \mid Q \xrightarrow{\pi} \nu\vec{x}.(\theta \mid P \mid R)}\ \textsc{Par-r}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\overline{M}(x)} \nu\vec{y}.\left(\left\{{}^{N}/_{x}\right\} \circ \sigma \mid P'\right) \quad \vec{z}\colon \sigma \mid Q \xrightarrow{M\,x} \nu\vec{w}.(\sigma \mid Q') \quad x, \vec{y} \# Q \quad \vec{w} \# P, \vec{y}}{\vec{z}\colon \sigma \mid P \mid Q \xrightarrow{\tau} \nu\vec{y}, \vec{w}.\left(\sigma \mid P' \mid Q'\left\{{}^{N\sigma}/_{x}\right\}\right)}\ \textsc{Close-l}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{M\,x} \nu\vec{y}.(\sigma \mid P') \quad \vec{z}\colon \sigma \mid Q \xrightarrow{\overline{M}(x)} \nu\vec{w}.\left(\left\{{}^{N}/_{x}\right\} \circ \sigma \mid Q'\right) \quad x, \vec{w} \# P \quad \vec{y} \# Q, \vec{w}}{\vec{z}\colon \sigma \mid P \mid Q \xrightarrow{\tau} \nu\vec{y}, \vec{w}.\left(\sigma \mid P'\left\{{}^{N\sigma}/_{x}\right\} \mid Q'\right)}\ \textsc{Close-r}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\pi} \nu\vec{x}.(\theta \mid Q) \quad \vec{x}, \mathrm{bn}(\pi) \ \# P}{\vec{z}\colon \sigma \mid !P \xrightarrow{\pi} \nu\vec{x}.(\theta \mid Q \mid !P)}\ \textsc{Rep-act}$$

$$\frac{\vec{z}\colon \sigma \mid P \xrightarrow{\overline{M}(x)} \nu\vec{y}.\left(\left\{{}^{N}/_{x}\right\} \circ \sigma \mid Q\right) \quad \vec{z}\colon \sigma \mid P \xrightarrow{M\,x} \nu\vec{w}.(\sigma \mid R) \quad \vec{y} \# P, \vec{w} \quad \vec{w} \# P}{\vec{z}\colon \sigma \mid !P \xrightarrow{\tau} \nu\vec{y}, \vec{w}.\left(\sigma \mid Q \mid R\left\{{}^{N\sigma}/_{x}\right\} \mid !P\right)}\ \textsc{Rep-close}$$

BOOKKEEPING

$$\mathrm{n}(\pi) = \begin{cases} \mathrm{fv}(M) \cup \{x\} & \text{if } \pi = \overline{M}(x) \\ \mathrm{fv}(M) \cup \mathrm{fv}(N) & \text{if } \pi = M\,N \\ \emptyset & \text{otherwise} \end{cases} \qquad \mathrm{bn}(\pi) = \begin{cases} \{x\} & \text{if } \pi = \overline{M}(x) \\ \emptyset & \text{otherwise} \end{cases}$$

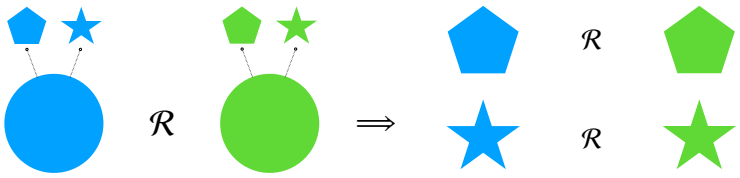
Figure 2.9: An open early labelled transition system.

distinguish between two given processes, then we give the formal definition of quasi-open bisimilarity with a self-contained background, continue with the description of the modal logic that characterises quasi-open bisimilarity, and present main results about quasi-open bisimilarity. Finally, we apply quasi-open bisimilarity to show that in an unlinkable authentication protocol scheme presented in the introduction, readers can be withdrawn from the picture.

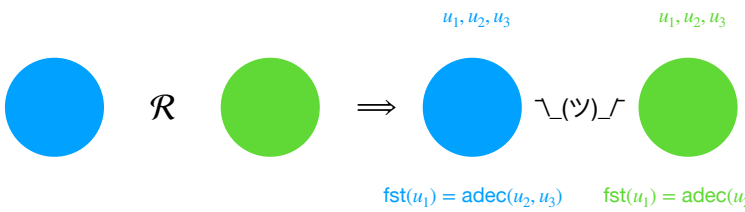
2.2.1 Attacker's capabilities

Let us return to mismatch discussion above. We have seen that there are situations when declaring some open variables private can enable mismatch guard, i.e. such manipulation with open variables can disable accessing any "universe" where two terms are equal. Such mechanism of accessing and considering different "universes" where the execution of the protocol might take place is the essence of our equivalence.

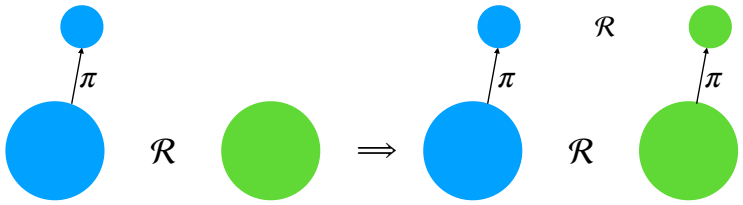
Being able to use the mechanism described above is the first capability of our attacker. Consider two equivalent states, depicted by relation $\mathfrak{R}$ below, then they must also be equivalent when we allow open variables to be instantiated with or fresh names, thereby declaring them private, or, capture avoiding substitutions, thereby declaring there is a particular message term behind the variable. We say that these "clarified" states (the pentagon and the star) are *accessible* (the dotted line) from the initial state (the circle). Returning back to the example in Fig. 2.1, if an attacker can instantiate open variables accessing a pair of states not belonging to the relation $\mathfrak{R}$, the protocol in question deviates from its ideal behaviour in this universe, and sessions with the same electronic document can be identified.



Secondly, our attacker is also able to compare two static snapshots (frames) of the system, known as *static equivalence*. Static equivalence is performed by checking whether all possible equality tests that may be constructed using a snapshot of an attacker's knowledge of the system, resulting in the same outcome for all tests. In the illustration below we have three message terms, labelled as u_1, u_2, and u_3, exposed to the environment and the sample testing equality $\mathtt{fst}(u_1) = \mathtt{adec}(u_2, u_3)$ that holds for both, blue and green, states. In the case of the example from Fig. 2.1, the equation that holds for the actual protocol might reflect that an electronic document presented at different sessions is the same document. This would never hold for the idealised protocol since no document is allowed to participate in a protocol more than once.



Finally, our attacker can make decisions dynamically. In any stage of execution, it can choose the perspective of one of the two processes and perform an input or output action (or a silent action τ). The attacker has a winning strategy if the other process cannot match the action at all, or can only do so by reaching an inequivalent state. Hence, if there is no attack there is always a way to match every action of a process with a corresponding action of the other process while staying within a relation $\mathfrak{R}$, as illustrated below. For our example from Fig. 2.1, an attacker capable of finding a pair of states and an action that cannot be matched by the idealised specification of the protocol would mean that there is a way of exploiting the fact that the same electronic document was used across several sessions.



The last two capabilities correspond to a standard notion of early bisimilarity[2].Therefore the key novelty of quasi-open bisimilarity is the first point above. In what follows we formally explain all three points starting from the easiest, which is static equivalence.

2.2.2 The definition of quasi-open bisimilarity

We start from the concept of static equivalence, which is no different from original work on the applied π-calculus [ABF17]. Static equivalence is defined over the static information in an extended process – the frame consisting of the active substitutions and new name binders.

Definition 3 (static equivalence)**.** *Two extended processes $\nu\vec{x}.(\sigma \mid \_)$ and $\nu\vec{y}.(\theta \mid \_)$, where $\_$ may be any two processes, are statically equivalent whenever the following holds. For all messages M and N such that $\vec{x},\vec{y} \# M, N$, we have $M\sigma =_E N\sigma$ if and only if $M\theta =_E N\theta$. Notation: $\nu\vec{x}.(\sigma \mid \_) \sim_{seq} \nu\vec{y}.(\theta \mid \_)$.*

A recipe is a message term that cannot refer to private information recorded in the extended process, i.e. variables bounded by the ν binder, but can refer to message aliases from the domain of the active substitution and free variables. Thus, in the above definition, messages M and N represent two different recipes for producing messages, and two extended processes are distinguished by static equivalence only when the two recipes produce equivalent messages under one substitution, but distinct messages under the other substitution.

For a self-contained presentation, we provide several examples. The following extended processes are not statically equivalent.

$$\nu k.\left(\left\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(x,\mathtt{pk}(k))}/_{v,w}\right\} \mid 0\right) \quad \sim_{seq} \quad \nu k.\left(\left\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(z,\mathtt{pk}(k))}/_{v,w}\right\} \mid 0\right)$$

[2]We purposefully use the term early bisimilarity used in the (core) π-calculus literature to be more precise. While in the applied π-calculus the term labelled bisimilarity is more common, we find the word "labelled" ambiguous since there are many distinct notions of bisimilarity defined in terms of a labelled transition system.

There are two distinguishing recipes v and w for which we have $v\sigma =_E w\sigma$, but $v\theta \neq_E w\theta$, where $\sigma = \left\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(x,\mathtt{pk}(k))}/_{v,w}\right\}$, $\theta = \left\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(z,\mathtt{pk}(k))}/_{v,w}\right\}$.

For a positive example, consider the following pair of extended processes.

$$\nu m, k, n.\left(\left\{{}^{\mathtt{aenc}(m,\mathtt{pk}(k)),n}/_{x_1,x_2}\right\} \mid 0\right) \quad \sim_{seq} \quad \nu m, k.\left(\left\{{}^{\mathtt{aenc}(m,\mathtt{pk}(k)),k}/_{x_1,x_2}\right\} \mid 0\right)$$

In this case, the static equivalence relies on the fact that our sample message theory, in Fig. 2.5, does not allow successful decryption to be detected. Thus, for example, recipe $\mathtt{adec}(x_1, x_2)$ produces what looks like a random number for both processes.

If a protocol requires successful decryption to be detected, we can introduce a tag when a nonce is encrypted. Then the presence of a tag after decryption implies that the decryption was successful. For example, consider the following extended processes, where the nonce m is tagged with t before being encrypted.

$$\nu m, k, n.\left(\left\{{}^{\mathtt{aenc}(\langle t,m\rangle,\mathtt{pk}(k)),n}/_{x_3,x_4}\right\} \mid 0\right) \quad \not\sim_{seq} \quad \nu m, k.\left(\left\{{}^{\mathtt{aenc}(\langle t,m\rangle,\mathtt{pk}(k)),k}/_{x_3,x_4}\right\} \mid 0\right)$$

In contrast to the previous example, the above are not statically equivalent, since they can be distinguished by recipes $\mathtt{fst}(\mathtt{adec}(x_3, x_4))$ and t, which only produce equal messages according to the extended process above right.

Next, we introduce the attacker's capability of manipulating free variables to consider different universes, which is reflected in the notion of openness of the relation between extended processes – if two states are related, they are related in the accessible universe.

Definition 4 (open)**.** *A relation over extended processes $\mathfrak{R}$ is open whenever we have that if $\nu\vec{x}.(\sigma \mid P)\ \mathfrak{R}\ \nu\vec{y}.(\theta \mid Q)$ and there exist variables $\vec{z}$ and idempotent substitution ρ such that: $\vec{z} \# \sigma, P, \theta, Q$ and $\rho \# \vec{x}, \vec{y}$, $\mathrm{dom}(\sigma)$, $\mathrm{dom}(\theta)$, we have*

$$\nu\vec{z}, \vec{x}.(\sigma \circ \rho \mid P\rho)\ \mathfrak{R}\ \nu\vec{z}, \vec{y}.(\theta \circ \rho \mid Q\rho)$$

In the context of the definition above, we say that the extended process $A \triangleq \nu\vec{x}.(\sigma \mid P)$ *can access* the extended process $A' \triangleq \nu\vec{z}, \vec{x}.(\sigma \circ \rho \mid P\rho)$ by the environment extension $\nu\vec{z}.\rho$, written as $A \sqsubseteq_{\nu\vec{z}.\rho} A'$ via $\nu\vec{z}.\rho$ if $\vec{z} \# \sigma, P$ and $\rho \# \vec{x}, \mathrm{dom}(\sigma)$. In what follows we omit subscript when is it not important how we access A' from A. Using accessibility mechanism an attacker can declare free variables private, and instantiate them with message terms without accessing private information and available messages.

An important property of accessibility is that it is monotonous as proven in the related work [HMY23], i.e. if a transition π available from the extended process A, it is always available in any accessible state A', however accessibility may enable new transitions, not available in the original state. For instance, we have $\nu x.\overline{a}\langle x\rangle \sqsubseteq_{\nu a.\{{}^{z}/_{a}\}} \nu z, x.\,(\{{}^{z}/_{a}\} \mid \overline{z}\langle x\rangle)$ and for both states the transition $\overline{a}v$ is available. For the example where a new transition is enabled consider the process $[x \neq z]a(y).[x = y]\pi$ that cannot yet act since there is no evidence x and z are distinct, or, similarly, there is a universe where they are equal, as in the remark

about mismatch in Sec. 2.1.3. However, by extending the frame with a fresh name, we have the following transition.

$$[x \neq z]a(y).[x = y]\pi \sqsubseteq_{\nu n.\{^n/_x\}} \nu n.(\{^n/_x\} \mid [n \neq z]a(y).[n = y]\pi) \xrightarrow{a\,x} \nu n.(\{^n/_x\} \mid [n = n]\pi)$$

Notice, how declaring the free variable private – a in the first example and x in the second example, makes these names available to be used as a the message alias in the domains of $\{^z/_a\}$ and $\{^n/_x\}$ respectively.

Finally, we incorporate the attacker's capability of making decisions dynamically directly in the definition of quasi-open bisimilarity for the applied π-calculus.

Definition 5 (quasi-open bisimilarity)**.** *A symmetric relation between extended processes $\mathfrak{R}$ is a quasi-open bisimulation if*

- *$\mathfrak{R}$ is open.*

And whenever $A\ \mathfrak{R}\ B$ we have the following:

- *A and B are statically equivalent.*
- *If $A \xrightarrow{\pi} A'$ there exists B' such that $B \xrightarrow{\pi} B'$ and $A'\ \mathfrak{R}\ B'$.*

Processes P and Q are quasi-open bisimilar whenever $P\ \mathfrak{R}\ Q$ for some quasi-open bisimulation $\mathfrak{R}$. Notanion: $P \sim Q$.

The keyword in the definition above is "open" in the sense of Def. 4. Without ensuring that properties are preserved under accessibility, the above definition would be a strong version of the classical *labelled bisimilarity* for the applied π-calculus [ABF17], i.e. when the number of internal τ transitions is observable (in contrast to weak, where the number of τ transactions is hidden).

By insisting that a quasi-open bisimulation is an open relation as in Def. 4, static equivalence must also be preserved by all fresh substitutions. This has an impact on examples such as the following.

Processes $\nu x.\overline{a}\langle \mathtt{aenc}(x,z)\rangle$ and $\nu x.\overline{a}\langle \mathtt{aenc}(\langle x,y\rangle,z)\rangle$ are labelled bisimilar but not quasi-open bisimilar. Indeed, let v be the alias for the message output on channel a, then both processes can perform a $\overline{a}(v)$-transition, to the respective extended processes $\nu x.\left(\left\{^{\mathtt{aenc}(x,z)}/_v\right\} \mid 0\right)$ and $\nu x.\left(\left\{^{\mathtt{aenc}(\langle x,y\rangle,z)}/_v\right\} \mid 0\right)$. These extended processes are statically equivalent (recall z cannot be used to decrypt these cyphertexts in asymmetric cryptography). However, since a quasi-open bisimulation must be preserved under substitution $\left\{^{\mathtt{pk}(w)}/_z\right\}$ # x, v, we should check static equivalence of $\nu x.\left(\left\{^{\mathtt{aenc}(x,z)}/_v\right\} \mid 0\right)\left\{^{\mathtt{pk}(w)}/_z\right\}$ and $\nu x.\left(\left\{^{\mathtt{aenc}(\langle x,y\rangle,z)}/_v\right\} \mid 0\right)\left\{^{\mathtt{pk}(w)}/_z\right\}$. After applying the substitution, the extended processes are no longer statically equivalent, witnessed by distinguishing recipes $\mathtt{snd}(\mathtt{adec}(v,w))$ and y. Thus the processes are not quasi-open bisimilar. The fact that such an attack succeeds is directly caused by the attacker's capability to influence the message bound to z in order to stage an attack, thereby accessing the universe where it is possible to distinguish between two states. In that universe, an attacker chooses that message to be a public key $\mathtt{pk}(w)$ for which he knows the secret key w.

For another example, consider processes $\nu k.\overline{a}\langle \mathtt{aenc}(x, \mathtt{pk}(k))\rangle.\overline{a}\langle \mathtt{aenc}(y, \mathtt{pk}(k))\rangle$ and $\nu k.\overline{a}\langle \mathtt{aenc}(x, \mathtt{pk}(k))\rangle.\overline{a}\langle \mathtt{aenc}(z, \mathtt{pk}(k))\rangle$ which are labelled bisimilar but not quasi-open bisimilar. To see why, firstly observe the above processes move to the states $\nu k.\Big(\Big\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(y,\mathtt{pk}(k))}/_{v,w}\Big\} \mid 0\Big)$ and $\nu k.\Big(\Big\{{}^{\mathtt{aenc}(x,\mathtt{pk}(k)),\mathtt{aenc}(z,\mathtt{pk}(k))}/_{v,w}\Big\} \mid 0\Big)$ respectively, by executing both output actions. Secondly, we apply the substitution $\{{}^{y}/_{x}\}$ satisfying the conditions from Def. 4, setting x to y and reaching a scenario explained after Def. 3, where the attacker can observe the same message is output twice for the process on the left but not for the process on the right.

This feature of quasi-open bisimilarity is related to the security property of *strong secrecy* [Bla04], meaning that an adversary cannot detect when the value of the secret changes. The open nature of secrets when modelling such property represents that the attacker may interfere with messages at runtime.

Lastly, we provide an example for which it matters that channels can also be message terms. Consider the following pair of processes $\nu z.\overline{x}\langle z, y\rangle.z(w)$ and $\nu z.\overline{x}\langle z, y\rangle$. For both processes we firstly execute the output action $\overline{x}(v)$, where v is the alias for the message a process outputs.

$$\nu z.\overline{x}\langle z, y\rangle.z(w) \xrightarrow{\overline{x}(v)} \nu z.\Big(\Big\{{}^{\langle z,y\rangle}/_{v}\Big\} \mid z(w)\Big) \quad \nu z.\overline{x}\langle z, y\rangle \xrightarrow{\overline{x}(v)} \nu z.\Big(\Big\{{}^{\langle z,y\rangle}/_{v}\Big\} \mid 0\Big)$$

Then an attacker can use $\mathtt{fst}(v)$ to refer to channel z when executing the next available action $\mathtt{fst}(v)\, w$ for the process on the left $\nu z.\Big(\Big\{{}^{\langle z,y\rangle}/_{v}\Big\} \mid z(w)\Big) \xrightarrow{\mathtt{fst}(v)\, w} \nu z.\Big(\Big\{{}^{\langle z,y\rangle}/_{v}\Big\} \mid 0\Big)$. The process $\nu z.\Big(\Big\{{}^{\langle z,y\rangle}/_{v}\Big\} \mid 0\Big)$ on the right cannot match this transition, hence the initial processes are not quasi-open bisimilar.

2.2.3 Whenever quasi-open bisimilarity fails, a modal logic formula describes an attack

Our first demand for an equivalence suitable to model privacy properties of cryptographic protocols is to have sound and complete modal logic characterisation. In what follows we describe the modal logic intuitionistic $\mathcal{FM}$ that characterises quasi-open bisimilarity, hence quasi-open bisimilarity meets this demand. The modal logic intuitionistic $\mathcal{FM}$ presented below extends the restricted version communicated at LICS'18 [HALT18] used to characterise quasi-open bisimilarity for the core π-calculus, where messages can be variables only. Since we consider arbitrary messages, this logic for the applied π-calculus can describe attacks on privacy properties expressed as process equivalence problems. For the sake of completeness, we formulate the corresponding results about this logic, but we stress that these results are not among the contributions of the current dissertation, though the author has contributed to the paper [HMY23] this subsection is based on.

The syntax of intuitionistic $\mathcal{FM}$ is presented below.

$$
\begin{array}{rlr}
\phi ::= & \mathtt{tt} & \text{top} \\
\mid & \mathtt{ff} & \text{bottom} \\
\mid & M = N & \text{equality} \\
\mid & \phi \wedge \phi & \text{conjunction} \\
\mid & \phi \vee \phi & \text{disjunction} \\
\mid & \phi \supset \phi & \text{implication} \\
\mid & \langle \pi \rangle \phi & \text{diamond} \\
\mid & [\pi] \phi & \text{box}
\end{array}
\quad \left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \\ \\ \end{array}\right\} \begin{array}{l} \text{intuitionistic logic} \\ \\ \\ \\ \\ \text{modalities} \\ \\ \end{array}
$$

In the syntax above, observe that connectives cover the standard conjunction, disjunction, implication, top, and bottom of intuitionistic logic with equality predicates. Negation $\neg\phi$ is an abbreviation for $\phi \supset \mathtt{ff}$ and hence inequality $M \neq N$ is defined as $(M = N) \supset \mathtt{ff}$. The two modalities box and diamond range over all observable actions.

Observable actions π, as defined in Section 2.1.3, range over bound outputs $\overline{M}(u)$, free inputs $M\,N$, and τ. The symbol $\mathcal{F}$ indicates the use of free inputs, where the message input appears as a message, reflecting the fact that we use *early* labelled transition system. The symbol $\mathcal{M}$ indicates the presence of "match", i.e., the equality predicates, which are a way to reflect the role of static equivalence in the logic. Finally, the semantics of intuitionistic $\mathcal{FM}$ is given in Fig. 2.10.

$A \models \mathtt{tt}$		always holds.
$\nu\vec{x}.(\sigma \mid P) \models M = N$	iff	$M\sigma =_E N\sigma$ and $\vec{x} \# M, N$
$A \models \phi_1 \wedge \phi_2$	iff	$A \models \phi_1$ and $A \models \phi_2$.
$A \models \phi_1 \vee \phi_2$	iff	$A \models \phi_1$ or $A \models \phi_2$.
$A \models \phi_1 \supset \phi_2$	iff	whenever $A \sqsubseteq B$, and $B \models \phi_1$, we have $B \models \phi_2$.
$A \models \langle \pi \rangle \phi$	iff	there exists B such that $A \xrightarrow{\pi} B$ and $B \models \phi$.
$A \models [\pi] \phi$	iff	whenever $A \sqsubseteq B$, and $B \xrightarrow{\pi} C$, we have $C \models \phi$.

Figure 2.10: The semantics of intuitionistic modal logic $\mathcal{FM}$ for the applied π-calculus. Satisfaction $\models$ is the least relation satisfying the rules above; hence $\mathtt{ff}$ is defined implicitly, as there are no rules that force $A \models \mathtt{ff}$ for any A.

Now we state that quasi-open bisimilarity is sound with respect to intuitionistic $\mathcal{FM}$.

Theorem 1 (soundness)**.** *If $P \sim Q$, then for all ϕ, $P \models \phi$ if and only if $Q \models \phi$.*

The proof of the theorem above is conducted by induction on the structure of formulae and it shows, that any attack strategy described by a formula that is valid for one process, is valid for any quasi-open bisimilar process. The completeness result, as explained in [HMY23], holds whenever the message theory is finitary [BN99]. For the sake of completeness, we provide the reader with the definition of finitary message theory.

Definition 6. *An equational theory E is finitary whenever, for all messages M and N, there is a finite set of substitutions $\{\sigma_i\}_{i\in I}$ such that, for all $i \in I$, we have $M\sigma_i =_E N\sigma_i$ and, for all θ such that $M\theta =_E N\theta$, there exists $j \in I$ such that σ_j is more general than θ, i.e. there exists ς s.t. $\sigma_j\varsigma = \theta$.*

Examples of finitary message theories include standard symmetric encryption and the equational theory in Fig. 2.5. Message theories with an associative operator, such as string concatenation, are not finitary in general [Plo72]. However, an associative-commutative operator, such as xor [AFN17], is finitary [Sti81, Fag87]. The restriction to finitary message theories allows us to write down a finite set of equations representing all substitutions that equate two messages. The completeness result, then, is formulated as follows.

Theorem 2 (completeness)**.** *For a finitary message theory the following holds. If for all ϕ, $P \models \phi$ iff $Q \models \phi$, then $P \sim Q$.*

The proof of the above theorem requires a notion of distinguishing strategy, which is, roughly speaking, a series of attacker's actions (that include accessing different worlds as in Def. 4) leading to either a pair of states where one state can make a transition which the other state cannot match, or, a pair of states that are not statically equivalent. The first step of the proof ensures that two processes are not quasi-open bisimilar whenever there is a finite distinguishing strategy and the second step describes how to construct a distinguishing formula from that distinguishing strategy.

Finally, we present several examples demonstrating how intuitionistic $\mathcal{FM}$ is employed to represent distinguishing strategies whenever quasi-open bisimilarity fails.

For the first example consider the processes $\nu m, n.\overline{a}\langle m\rangle.\overline{a}\langle n\rangle \not\sim \nu n.\overline{a}\langle n\rangle.\overline{a}\langle \mathtt{h}(n)\rangle$, which are not quasi-open bisimilar. A distinguishing strategy is that both processes can perform output transitions $\overline{a}(u)$ and $\overline{a}(v)$, where u and v are respective message aliases, reaching the pair of processes $\nu m, n.(\{^{m,n}/_{u,v}\} \mid 0) \not\sim \nu n.\left(\left\{^{n,\mathtt{h}(n)}/_{u,v}\right\} \mid 0\right)$. These processes are not statically equivalent with distinguishing messages v and $\mathtt{h}(u)$. Thereby we can construct the following distinguishing formulae representing the quasi-open bisimilarity failure, biased to the left or the right process.

$$\begin{array}{cc} \phi = \langle\overline{a}(u)\rangle\langle\overline{a}(v)\rangle(v \neq \mathtt{h}(u)) & \psi = \left[\overline{a}(u)\right]\left[\overline{a}(v)\right](v = \mathtt{h}(u)) \\ \nu m, n.\overline{a}\langle m\rangle.\overline{a}\langle n\rangle \models \phi & \nu m, n.\overline{a}\langle m\rangle.\overline{a}\langle n\rangle \not\models \phi \\ \nu n.\overline{a}\langle n\rangle.\overline{a}\langle h(n)\rangle \not\models \phi & \nu n.\overline{a}\langle n\rangle.\overline{a}\langle h(n)\rangle \models \psi \end{array}$$

The duality between ϕ and ψ is rooted in the proof of Theorem 2 [HMY23] where it is explained how the distinguishing formula is being constructed from a distinguishing strategy. Two formulas constructed from the strategy described above are as follows. The formula ϕ essentially tells that there is no world accessible from the resulting state of the left process (after two output action has been taken) where $v = \mathtt{h}(u)$ holds, while the formula ϕ tells that in any accessible resulting that a right process can reach $v = \mathtt{h}(u)$ always holds.

For the second example recall $\nu x.\overline{a}\langle \mathtt{aenc}(x,z)\rangle \not\sim \nu x.\overline{a}\langle \mathtt{aenc}(\langle x,y\rangle,z)\rangle$ from previous section, after the Def. 5 of quasi-open bisimilarity. The distinguishing strategy, described previously, the involved substitution $\left\{{}^{\mathtt{pk}(w)}/_{z}\right\}$, and distinguishing recipes $\mathtt{snd}(\mathtt{adec}(u,w))$ and y, both of which appear in the following distinguishing formula biased to the right, which tells that in any world accessible from the resulting state where $z = \mathtt{pk}(w)$ holds, we also have $\mathtt{snd}(\mathtt{adec}(u,w)) = y$.

$$\nu x.\overline{a}\langle \mathtt{aenc}(\langle x,y\rangle,z)\rangle \models \langle \overline{a}(u)\rangle (z = \mathtt{pk}(w) \supset \mathtt{snd}(\mathtt{adec}(u,w)) = y)$$

From the same strategy, we can construct the distinguishing formula biased to the left, telling that for any accessible state, the left process can reach, there is no accessible world where $\mathtt{snd}(\mathtt{adec}(u,w)) = y$ holds.

$$\nu x.\overline{a}\langle \mathtt{aenc}(x,z)\rangle \models [\overline{a}(u)](\mathtt{snd}(\mathtt{adec}(u,w)) \neq y)$$

Our third example illustrates the situation when a transition cannot be matched by another process. Recall, from the end of Sec. 2.2.2 that $\nu z.\overline{x}\langle\langle z,y\rangle\rangle.z(w) \not\sim \nu z.\overline{x}\langle\langle z,y\rangle\rangle$. The distinguishing strategy presented there yields the following formulae biased to each process. Notice, yet again, how the message $\mathtt{fst}(u)$ is used as a channel name for the second output action.

$$\nu z.\overline{x}\langle\langle z,y\rangle\rangle.z(w) \models \langle \overline{a}(u)\rangle\langle \mathtt{fst}(u)\, w\rangle \mathtt{tt} \quad \text{and} \quad \nu z.\overline{x}\langle\langle z,y\rangle\rangle \models [\overline{a}(u)]\,[\mathtt{fst}(u)\, w]\,\mathtt{ff}$$

Finally, we develop further the example from Sec. 2.1.3 demonstrating that mismatch is intuitionistic and present a formula requiring the absence of the law of excluded middle. The following two processes are not quasi-open bisimilar.

$$\overline{a}\langle r\rangle \quad \not\sim \quad \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a}\langle \mathtt{aenc}(m,\mathtt{pk}(k))\rangle \mathtt{else}\ \overline{a}\langle r\rangle$$

The strategy exploiting the law of excluded middle is as follows. The $\overline{a}\langle r\rangle \xrightarrow{\overline{a}(u)} \{{}^{r}/_{u}\} \mid 0$, cannot be matched by the process on the right without additional assumptions about x and k. Thus for a distinguishing formula biased to the left, we have the following.

$$\overline{a}\langle r\rangle \models \langle a(u)\rangle \mathtt{tt} \qquad \text{and} \qquad \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a}\langle \mathtt{aenc}(m,\mathtt{pk}(k))\rangle \mathtt{else}\ \overline{a}\langle r\rangle \not\models \langle a(u)\rangle \mathtt{tt}$$

The process on the right can only perform an output transition either: under substitutions σ such that $x\sigma =_E \mathtt{pk}(k)\sigma$; or, under substitutions ρ and environments $\vec{n}$ such that $\vec{n} \models x\rho \neq \mathtt{pk}(k)\rho$. Hence, in a distinguishing formula biased to the right below, we have a postcondition indicating that either equality or inequality must be decided in advance.

$$\psi = [a(u)](x = \mathtt{pk}(k) \vee x \neq \mathtt{pk}(k))$$
$$\overline{a}\langle r\rangle \not\models \psi \qquad \text{and} \qquad \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a}\langle \mathtt{aenc}(m,\mathtt{pk}(k))\rangle \models \psi$$

Observe that in a classical setting neither of the above formulae would be distinguishing, and $\big[a(u)\big](x = \mathtt{pk}(k) \vee x \neq \mathtt{pk}(k))$ would be a tautology, due to the law of excluded middle. Thus the absence of the law of excluded middle for intuitionistic $\mathcal{FM}$ provides additional distinguishing power.

2.2.4 Quasi-open bisimilarity enables compositional reasoning

Our second demand for the target equivalence is that it should also be a congruence relation. The quasi-open bisimilarity, as defined in Def. 5 enjoys this property, since we have the following theorem reported in [HMY21].

Theorem 3 (contexts)**.** *If $P \sim Q$ then for all contexts $\mathcal{C}\{\cdot\}$, we have $\mathcal{C}\{P\} \sim \mathcal{C}\{Q\}$.*

The proof of the theorem above reported in [HMY21, HMY23] is of interest in itself and comprises a systematic check that quasi-open bisimilarity is preserved under each process construct, i.e. it is always possible to construct a quasi-open bisimulation containing two initial processes. Of particular interest is, e.g. the case for parallel composition – whenever $P \sim Q$, then for any R we have $P \mid R \sim Q \mid R$: despite the parallel processes R may diverge during executions as different data may be transmitted by both processes, it is shown that the related states belong to the constructed relation.

We motivate the importance of being congruence by developing the authentication to an untrusted device example from Fig. 2.3. Also, the introduced formalism allows us now to formalise the unlinkability scheme expressed in the introduction informally. Consider two roles, *C*, and *T*. The agent playing role *C* holds credentials signed by the secret key s of the certification authority *CA* and wants to be able to present the same identity multiple times without the risk of being reidentified. The goal of the agent playing *T* is to verify these credentials using the public key $\mathtt{pk}(s)$ of the *CA* and authenticate *C*. The real-world behaviour of the system can be modelled as follows.

$$Sys \triangleq \nu s.\Big(!\nu c.!\nu ch_c.\overline{doc}\langle ch_c\rangle.C(s, ch_c, c) \mid \overline{out}\langle \mathtt{pk}(s)\rangle.!\nu ch_t.\overline{reader}\langle ch_t\rangle.T(\mathtt{pk}(s), ch_t)\Big)$$

Initially, the *CA*'s secret key s is created. The first parallel component above defines agents with identity c that can participate in an **arbitrary number of sessions** of the protocol. Each session begins with advertising a fresh session channel ch_c on the public channel *doc*, modelling a new connection to a new session. The leftmost replication models that any number of agents can exist in the system, while the subsequent replication is what allows an agent to appear with the same identity across multiple sessions. The second parallel component above makes the public key $\mathtt{pk}(s)$ of the *CA* available to the environment via the output on the public channel *out*. After that, the role *T* is specified with a goal to authenticate a genuine agent in role *C* making use of $\mathtt{pk}(s)$. Such sessions in role *T* also begin by advertising a fresh session channel, in this case on the public channel *reader*. The process schemes $C(s, ch_c, c)$ and $T(\mathtt{pk}(s), ch_t)$ can be instantiated to model various protocols.

On the other hand, the idealised system is obtained from *Sys* by removing the second replication, which means that the agent with the identity c can participate in one protocol run only.

$$Spec \triangleq \nu s.\Big(!\nu c.\nu ch_c.\overline{doc}\langle ch_c\rangle.C(s,ch_c,c) \mid \overline{out}\langle \mathtt{pk}(s)\rangle.!\nu ch_t.\overline{reader}\langle ch_t\rangle.T(\mathtt{pk}(s),ch_t)\Big)$$

Since the definition of unlinkability is always protocol-specific, let us call the protocol scheme described above AU, standing for authentication to untrusted (device). The definition of unlinkability for AU below is in line with the original unlinkability definition [ACRR10][3].

Definition 7 (AU-unlinkability)**.** *The AU system satisfies unlinkability if Sys* $\approx$ *Spec holds, where* $\approx$ *is weak early bisimilarity.*

The modelling decision of using session channels announced on special dedicated public channels is not arbitrary. In the original definition of (strong) unlinkability proposed by Arapinis et al. [ACRR10], the underlying bisimilarity notion uses a weak transition system, which does not support image finiteness, i.e. for a given process A and the transition label π there could be infinitely many states B s.t. $A \xrightarrow{\pi} B$ which can make verification a difficult task. To overcome this obstacle, we follow a method developed by Horne and Mauw [HM21], allowing the reduction of weak to strong bisimilarity that supports image finiteness. The insight of their work is that expressing a protocol using such session channels makes verification easier without compromising unlinkability in the original sense. Not only session channels endows an attacker with the explicit ability to observe whether output/input events are performed within the same session and inject message in a particular session, they are among the requirements in the transport protocol ISO/IEC 14443 [iso18] used in electronic documents [pas15], payment cards [emv11], electronic ticketing system [cal21] etc.

The fact that quasi-open bisimilarity is a congruence allows us to verify an equivalence property for a smaller system looking exactly like the one in Fig. 2.1, but without the reader role T, and extend the proof to a larger system. Consider a smaller system comprising only agents playing the role C.

$$Small\_Sys \triangleq \nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.!\nu ch_c.\overline{doc}\langle ch_c\rangle.C(s,ch_c,c)$$

The corresponding, smaller version of the idealised specification where there is one session per identity is as follows.

$$Small\_Spec \triangleq \nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.\nu ch_c.\overline{doc}\langle ch_c\rangle.C(s,ch_c,c)$$

[3]Potential terminology clash. In their paper Arapinis et al. explain two flavours of unlinkability, weak and strong. In this dissertation we consider only strong unlinkability (hence we call it just unlinkability) in a sense explained in the introduction, i.e. when the world where the protocol can run multiple times with the same identity is indistinguishable from the world where the protocol can run with an identity no more than once. However, to define strong unlinkability, they use a weak labelled transition system. Hence, the reader here should not confuse between strong/weak unlinkability and strong/weak labelled transition systems.

We are ready to prove that if we prove unlinkability using the smaller specification with one role, then it holds in the more traditional specification with two roles.

Theorem 4. *If Small_Sys* $\sim$ *Small_Spec, then Sys* $\approx$ *Spec.*

Proof. Consider the following context, where out' is a free variable for any of the processes above.

$$\mathcal{C}\{\ \cdot\ \} \triangleq\ \nu out.\Big(\ \{\ \cdot\ \} \mid out(pks).\overline{out'}\langle pks\rangle.!\nu ch_t.\overline{reader}\langle ch_t\rangle.T(pks, ch_t)\Big)$$

Firstly, we have $\mathcal{C}\{Small\_Sys\}\{^{out}/_{out'}\} \sim \tau.Sys$ and $\mathcal{C}\{Small\_Spec\}\{^{out}/_{out'}\} \sim \tau.Spec$ hold. By the assumption *Small_Sys* $\sim$ *Small_Spec* and the fact that since quasi-open bisimilarity is a congruence (Theorem 3), the following holds.

$$\mathcal{C}\{Small\_Sys\} \sim \mathcal{C}\{Small\_Spec\}$$

Furthermore, since quasi-open bisimilarity is closed under substitutions involving free variables (by definition) we have that the following holds.

$$\mathcal{C}\{Small\_Sys\}\{^{out}/_{out'}\} \sim \mathcal{C}\{Small\_Spec\}\{^{out}/_{out'}\}$$

Hence, since quasi-open bisimilarity is an equivalence relation, we have the following.

$$\tau.Spec \sim \tau.Sys$$

Thus there exists a quasi-open bisimulation $\mathfrak{R}$ such that $\tau.Spec\ \mathfrak{R}\ \tau.Sys$. Hence, since $\tau.Spec \xrightarrow{\tau} Spec$ it must be the case that $\tau.Sys \xrightarrow{\tau} Sys$ and $Spec\ \mathfrak{R}\ Sys$. Therefore $Spec \sim Sys$. Finally, since $\sim \subseteq \approx$ we have $Spec \approx Sys$. □

Hence, in this situation, we are able to reduce significantly the amount of work needed for unlinkability verification by studying a smaller system and proving that *Small_Sys* $\sim$ *Small_Spec*. This approach to unlinkability allows us to revise the Def. 7 and consider only one party right in the definition. We will apply this approach in the next chapter where we will study the key agreement for contactless payments and consider only honest cards in the unlinkability definition.

This example also demonstrates the limitations of the method. Here we *can* drop one party, the verifier, from consideration because documents and verifiers share no secret, while otherwise, an observed reaction of an honest participant may break unlinkability.

2.2.5 Quasi-open bisimilarity is the coarsest bisimilarity congruence

Since quasi-open bisimilarity is a bisimilarity, it is by definition meets our third demand for an equivalence suitable for modelling privacy properties. In this subsection we motivate the choice of quasi-open bisimilarity over a finer bisimilarity congruence called open bisimilarity [San96, HM21] by providing an example of a protocol for which a privacy property fails if we employ open bisimilarity, but holds

if quasi-open bisimilarity is employed. The failure we describe below, however, has no practical interpretation, demonstrating that open bisimilarity is too fine to accurately model real-world protocols.

The example we use as illustration is a variant of a private server example [AF04, CCLD17]. Similarly to previous illustrations, we express the privacy property as an equivalence between the "real" and the "ideal" behaviours. Consider a server *Server A* that responds with an encrypted message only when it receives a particular public key. Otherwise, it responds with a nonce, indistinguishable from a ciphertext. We assume an attacker knows public key $\mathtt{pk}(k)$ but does not know private key k or nonce r.

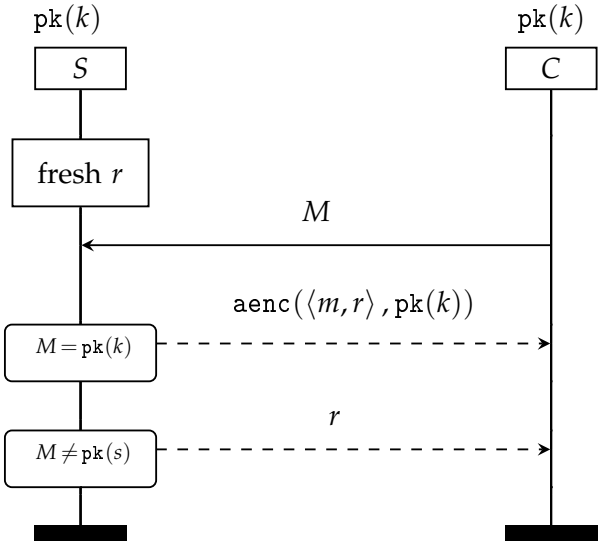


Figure 2.11: Specification of a private server *Server A*.

Informally, a private server can be described by the message sequence chart in Fig. 2.11 and, formally, *Server A* can be modelled in the applied π-calculus as follows.

$$\textit{Server A}: \quad \nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.!\,\nu a.\overline{c}\langle a\rangle.a(x).\nu r. \\ \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a}\langle \mathtt{aenc}(\langle m, r\rangle, \mathtt{pk}(k))\rangle\ \mathtt{else}\ \overline{a}\langle r\rangle$$

In the specification for *Server A*, the prefix $\nu k.\overline{s}\langle \mathtt{pk}(k)\rangle$ stands for announcing a public key on a public channel s. The prefix $!\nu a.\overline{c}\langle a\rangle.a(x).\nu r$ represents the start of an unbounded number of sessions on a fresh channel a where, in each session, an input is received and a nonce r is freshly generated. In each session, one of the following decisions is made, based on the input received. If an input is a public key output previously, *Server A* responds with a message-nonce pair encrypted with the public key $\overline{a}\langle \mathtt{aenc}(\langle m, r\rangle, \mathtt{pk}(k))\rangle$. Otherwise, *Server A* sends a random dummy message r indistinguishable from a random ciphertext. In this minimal formulation of the problem, we refrain from modelling the clients (possibly knowing key k). Of course, the fact that clients transmit their public keys in plaintext may introduce further privacy concerns, which we do not model in this minimal illustration.

We approach the problem of proving that the privacy of the owner of secret key k is preserved by specifying idealised behaviour. The private server from the perspective of an attacker should ideally behave as *Server B* below, which differs from *Server A* in that it transmits a nonce regardless of the message received.

$$\textit{Server B}: \quad \nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.!\nu a.\overline{c}\langle a\rangle.a(x).\nu r.\overline{a}\langle r\rangle$$

Server B and *Server A* are indistinguishable to an external observer – the attacker. An attacker cannot learn that *Server A* responds in a special way to input $\mathtt{pk}(k)$. The idea is that an attacker without private key k cannot learn that *Server A* serves some data m to the owner of k. Thus the privacy of the intended recipient of the data is preserved.

The point we make here is that if we employ open bisimilarity, which is also a congruence, to verify the privacy of the owner of secret key k, the processes above are not equivalent. Using a suitable labelled transition system [HM21], *Server A* can reach the following state, at which point open bisimilarity still allows x, a free variable representing an input, to be instantiated with the message bound to u, representing $\mathtt{pk}(k)$, output previously.

$$\begin{aligned}\nu k, a_1, r_1.\Big(\Big\{{}^{\mathtt{pk}(k),a_1}/_{u,v}\Big\} \mid \quad & \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a_1}\langle \mathtt{aenc}(\langle m, r_1\rangle, \mathtt{pk}(k))\rangle\ \mathtt{else}\ \overline{a_1}\langle r_1\rangle \\ \mid \quad & !\ \ \nu a.\overline{c}\langle a\rangle.a(x).\nu r. \\ & \quad \mathtt{if}\ x = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a}\langle \mathtt{aenc}(\langle m, r\rangle, \mathtt{pk}(k))\rangle\ \mathtt{else}\ \overline{a}\langle r\rangle\ \Big)\end{aligned}$$

Thus, we have not yet committed to $x = \mathtt{pk}(k)$ or $x \neq \mathtt{pk}(k)$, and hence we cannot proceed until we provide more information about x. Therefore the guard in the if-then-else statement above cannot yet be resolved. To the contrary, *Server B* cannot reach an equivalent state, since it can only reach a state which is immediately ready to perform an action regardless of whether $x = \mathtt{pk}(k)$ or $x \neq \mathtt{pk}(k)$. Note, we are still in an intuitionistic setting [AHT17, AHT21] as we have explained in a remark in the end of Sec. 2.1.3, and illustrated by the last example in Sec. 2.2.3; and we do not assume $x = \mathtt{pk}(k) \vee x \neq \mathtt{pk}(k)$ holds, which would be an instance of the law of excluded middle. The presented distinguishing strategy, does not correspond to a real attack on the privacy of *Server A*; hence open bisimilarity is not sufficiently coarse to verify this privacy property.

Quasi-open bisimilarity, on the other hand, verifies the privacy property for the protocol in Fig. 2.11, as we explain below by providing a relation between *Server B* and *Server A* satisfying conditions of quasi-open bisimulation in Def. 5.

For greater clarity, we first consider the case of a single session, i.e., with replication removed. The equivalence of running examples *Server B* and *Server A* for the single session case can be established by taking the least *symmetric open* relation satisfying the constraints in Fig. 2.12.

The critical observation is that message N in Fig. 2.12 ranges over all permitted inputs. Since $N = u$ is permitted, we have the following pair in relation $\mathfrak{S}$.

$$\nu k, a, r.\Big(\Big\{{}^{\mathtt{pk}(k),a}/_{u,v}\Big\} \mid \overline{a}\langle r\rangle\Big)\ \ \mathfrak{S}\ \ \nu k.a, r.\Big(\Big\{{}^{\mathtt{pk}(k),a}/_{u,v}\Big\} \mid \begin{array}{l}\mathtt{if}\ \mathtt{pk}(k) = \mathtt{pk}(k)\ \mathtt{then} \\ \overline{a}\langle \mathtt{aenc}(\langle m, r\rangle, \mathtt{pk}(k))\rangle \\ \mathtt{else}\ \overline{a}\langle r\rangle\Big)\end{array}$$

$$\nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.\nu a.\overline{c}\langle a\rangle.a(x).\nu r.\overline{a}\langle r\rangle \;\mathfrak{S}\; \nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.\nu a.\overline{c}\langle a\rangle.a(x).\nu r.\,\mathtt{if}\,x=\mathtt{pk}(k)\;\mathtt{then}\,\overline{a}\langle \mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))\rangle\;\mathtt{else}\,\overline{a}\langle r\rangle$$

$$\nu k.\left(\left\{{}^{\mathtt{pk}(k)}/_{u}\right\} \mid \nu a.\overline{c}\langle a\rangle.a(x).\nu r.\overline{a}\langle r\rangle\right) \;\mathfrak{S}\; \nu k.\left(\left\{{}^{\mathtt{pk}(k)}/_{u}\right\} \mid \nu a.\overline{c}\langle a\rangle.a(x).\nu r.\,\mathtt{if}\,x=\mathtt{pk}(k)\;\mathtt{then}\,\overline{a}\langle \mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))\rangle\;\mathtt{else}\,\overline{a}\langle r\rangle\right)$$

$$\nu k,a.\left(\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} \mid a(x).\nu r.\overline{a}\langle r\rangle\right) \;\mathfrak{S}\; \nu k,a.\left(\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} \mid a(x).\nu r.\,\mathtt{if}\,x=\mathtt{pk}(k)\;\mathtt{then}\,\overline{a}\langle \mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))\rangle\;\mathtt{else}\,\overline{a}\langle r\rangle\right)$$

$$\nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} \mid \overline{a}\langle r\rangle\right) \;\mathfrak{S}\; \nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} \mid \mathtt{if}\,N\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\}=\mathtt{pk}(k)\;\mathtt{then}\,\overline{a}\langle \mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))\rangle\;\mathtt{else}\,\overline{a}\langle r\rangle\right)$$

$$\nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,r}/_{u,v,w}\right\} \mid 0\right) \;\mathfrak{S}\; \nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,\mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))}/_{u,v,w}\right\} \mid 0\right)$$

$$\nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,r}/_{u,v,w}\right\} \mid 0\right) \;\mathfrak{S}\; \nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,r}/_{u,v,w}\right\} \mid 0\right)$$

where s, c, m and N are messages and u, v and w are variables such that $s,c,m,N \# a,k,r$, and $u \# s,c,m,a,k,r$, and $v \# s,c,a,m,k,r,u$, and $w \# s,c,m,a,k,r,u,v,N$.

Figure 2.12: Relation $\mathfrak{S}$ defining a quasi-open bisimulation verifying the anonymity of *Server A* in the case for a single session, without replication.

In the above, observe the branch sending an encrypted message is enabled. Otherwise, we always have $k,a,r \models N\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} \neq \mathtt{pk}(k)$ since if N were a message term such that $k,a,r \# N$ such that $N\left\{{}^{\mathtt{pk}(k),a}/_{u,v}\right\} = \mathtt{pk}(k)$, then N must be equivalent to u. Thus in all other cases, the `else` branch is enabled.

Notice that the state $\nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,r}/_{u,v,w}\right\} \mid 0\right)$ is statically equivalent to the state $\nu k,a,r.\left(\left\{{}^{\mathtt{pk}(k),a,\mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))}/_{u,v,w}\right\} \mid 0\right)$. To see why, observe that an attacker neither has the key k to decrypt $\mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))$, nor can an attacker reconstruct the message $\langle m,r\rangle$, without knowing r.

For the unbounded case, let us first define the process specifying the server role as $R \triangleq \nu a.\overline{c}\langle a\rangle.a(x).\nu r.\mathtt{if}\,x = \mathtt{pk}(k)\;\mathtt{then}\,\overline{a}\langle \mathtt{aenc}(\langle m,r\rangle,\mathtt{pk}(k))\rangle\;\mathtt{else}\,\overline{a}\langle r\rangle$, and consider the least symmetric open relation $\mathfrak{T}$ satisfying the constraints in Fig. 2.13. This generalises the finite case by defining all scenarios where there are l parallel sessions that are either in the state of having just announced the communication channel a, having just received a message, or having responded already. The relation $\mathfrak{T}$ is closed under all transitions and accessibility, as required to establish that $\mathfrak{T}$ is a quasi-open bisimulation, moreover $\mathfrak{T}$ is a quasi-open bisimulation such that *Server B* $\mathfrak{T}$ *Server A*, hence the desired privacy property of *Server A* being indistinguishable from the idealised *Server B* always responding with a nonce is verified.

$$\nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.!\nu a.\overline{c}\langle a\rangle.a(x).\nu r.\overline{a}\langle r\rangle \;\; \mathfrak{T} \;\; \nu k.\overline{s}\langle \mathtt{pk}(k)\rangle.!R$$

$$\nu k, a_1, \ldots a_l, r_1, \ldots r_l.\Big(\sigma \mid P_1 \mid \ldots P_l \mid\; !\nu a.\overline{c}\langle a\rangle.a(x).\nu r.\overline{a}\langle r\rangle\Big) \;\; \mathfrak{T} \;\; \nu k, a_1, \ldots a_l, r_1, \ldots r_l.\Big(\theta \mid Q_1 \mid \ldots Q_l \mid\; !R\Big)$$

for any I, J, I', J' partitioning $\{1, \ldots l\}$ such that the following hold

$$\begin{array}{ll} u\sigma = \mathtt{pk}(k) & \\ v_i\sigma = a_i & \text{if } i \in \{1,\ldots l\} \\ w_i\sigma = r_i & \text{if } i \in I' \cup J' \end{array} \qquad \begin{array}{ll} u\theta = \mathtt{pk}(k) & \\ v_i\theta = a_i & \text{if } l \in \{1,\ldots l\} \\ w_i\theta = \mathtt{aenc}(\langle m, r_i\rangle, \mathtt{pk}(k)) & \text{if } l \in I' \\ w_i\theta = r_i & \text{if } l \in J' \end{array}$$

$$P_i \triangleq \begin{cases} a_i(x).\nu r.\overline{a_i}\langle r\rangle & \text{if } i \in I \\ \overline{a_i}\langle r_i\rangle & \text{if } i \in J \\ 0 & \text{if } l \in I' \cup J' \end{cases}$$

$$Q_i \triangleq \begin{cases} a_i(x).\nu r.\mathtt{if}\ x\theta = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a_i}\langle \mathtt{aenc}(\langle m, r\rangle, \mathtt{pk}(k))\rangle\ \mathtt{else}\ \overline{a_i}\langle r\rangle & \text{if } i \in I \\ \mathtt{if}\ N_i\theta = \mathtt{pk}(k)\ \mathtt{then}\ \overline{a_i}\langle \mathtt{aenc}(\langle m, r_i\rangle, \mathtt{pk}(k))\rangle\ \mathtt{else}\ \overline{a_i}\langle r_i\rangle & \text{if } i \in J \\ 0 & \text{if } l \in I' \cup J' \end{cases}$$

$$\begin{array}{l} s, c, m, N_i \,\#\, k, a_1, \ldots a_l, r_1, \ldots r_l \\ u, v_1, \ldots v_l, w_1 \ldots w_l \,\#\, s, c, m, a_1, \ldots a_l, k, r_1, \ldots r_l \end{array}$$

Figure 2.13: Relation $\mathfrak{T}$ verifying *Server B* $\sim$ *Server A* in the unbounded case.

In fact, quasi-open bisimilarity is not just coarser than open bisimilarity, allowing us to verify the examples similar to the one in Fig. 2.11, it is *the coarsest (strong) bisimilarity congruence for the applied π-calculus* since it coincides with an equivalence that is the coarsest bisimilarity congruence in the objective sense it is defined. This equivalence is *open barbed bisimilarity* [SW01a] and is by definition a congruence defined independently of the messages sent and received. Below we define open barbed bisimilarity and formulate the result published in the related work [HMY23].

We say process P has *barb* M, written $P{\downarrow}M$, whenever, for some A, $P \xrightarrow{\overline{M}(z)} A$, or $P \xrightarrow{M\,N} A$, that is a barb represents the ability to observe an input or output action on a channel.

Definition 8 (open barbed bisimilarity)**.** *An open barbed bisimulation $\mathfrak{R}$ is a symmetric relation over processes such that whenever $A\ \mathfrak{R}\ B$ holds the following hold:*

- *For all contexts $\mathcal{C}\{\ \cdot\ \}$, $\mathcal{C}\{A\}\ \mathfrak{R}\ \mathcal{C}\{B\}$.*
- *If $A{\downarrow}M$ then $B{\downarrow}M$.*
- *If $A \xrightarrow{\tau} A'$, there exists B' such that $B \xrightarrow{\tau} B'$ and $A'\ \mathfrak{R}\ B'$ holds.*

Processes A and B are open barbed bisimilar whenever there exists an open barbed bisimulation $\mathfrak{R}$ such that $A\ \mathfrak{R}\ B$.

The power of open barbed bisimilarity comes from closing by all contexts at every step, not only at the beginning of execution. Closing by all contexts at every

step ensures the robustness of open barbed bisimilarity even if the environment is extended at runtime; i.e., we stay within a congruence relation at every step of the bisimulation game. Open barbed bisimilarity is concise – the definition requires only the open labelled transition system in Fig. 2.9 and the three clauses in Def. 8. Notice also that due to the independence of the information on the labels, open barbed bisimilarity applies to any language. However, it is unwieldy due to closure of the definition under all contexts, making it unsuitable for verification.

Having objective definition Def. 8 above we formulate the following.

Theorem 5. *Quasi-open bisimilarity coincides with open barbed bisimilarity.*

With the theorem and the definition of open barbed bisimilarity above, the value of quasi-open bisimilarity becomes apparent – it admits effective verification, as this dissertation demonstrates.

2.2.6 Proof certificates vs. formal proofs

To be more precise, a distinguishing modal logic formula and the relation similar to that presented in Fig. 2.13 are proof certificates rather than formal proofs. This means that to obtain a formal proof of non-equivalence or equivalence between two applied π-calculus processes, one should respectively do the following. For a given modal logic formula and two processes, one should check that, indeed, the formula holds for one process and also that it does not hold for the other process. For a given open relation one should check that it satisfies the defining clauses for a quasi-open bisimulation. It would be useful to have a tool for automated checking of such proof certificates which we consider as future work. Such a tool would play an important part in the automated verification of privacy properties of cryptographic protocols, i.e. a verification outcome should always be backed up with a proof certificate that can be checked independently, thereby ensuring that the outcome is correct. We have already checked in Sec. 2.2.3 several "negative" certificates comprising formulas that correspond to the failure of two processes being quasi-open bisimilar. For the sake of completeness of this background chapter below, we provide a reader with a full check of a "positive certificate" comprising a quasi-open bisimulation relation that corresponds to the fact that two processes are quasi-open bisimilar.

Consider the process $A \triangleq \nu x.\overline{a}\langle x\rangle.\,(a(y)+a(y).\tau + a(y).\,[x=y]\,\tau)$ is quasi-open bisimilar to the process $B \triangleq \nu x.\overline{a}\langle x\rangle.\,(a(y)+a(y).\tau)$. Let us check the certificate presented in Fig. 2.14 comprising defining conditions for a quasi-open bisimulation $\mathfrak{R}$ between A and B. The relation $\mathfrak{R}$ is the least symmetric open relation satisfying presented conditions. Notice that to construct such certificate we simply list all possible pairs of states reachable by the same transitions.

Then, to check this certificate, we should verify that $\mathfrak{R}$ relates initial processes A and B and that it is indeed a quasi-open bisimulation. That is, according to Def. 5, we must demonstrate the following.

$$
\begin{array}{lrcll}
(1) & \nu x.\overline{a}\langle x\rangle.\,(a(y){+}a(y).\tau + a(y).\,[x=y]\,\tau) & \mathfrak{R} & \nu x.\overline{a}\langle x\rangle.\,(a(y){+}a(y).\tau) & \\
(2) & \nu x.(\{^{x}/_{u}\} \mid a(y){+}a(y).\tau + a(y).\,[x=y]\,\tau) & \mathfrak{R} & \nu x.(\{^{x}/_{u}\} \mid a(y){+}a(y).\tau) & \\
(3) & \nu x.(\{^{x}/_{u}\} \mid 0) & \mathfrak{R} & \nu x.(\{^{x}/_{u}\} \mid 0) & \\
(4) & \nu x.(\{^{x}/_{u}\} \mid \tau) & \mathfrak{R} & \nu x.(\{^{x}/_{u}\} \mid \tau) & \\
(5) & \nu x.(\{^{x}/_{u}\} \mid [x=N\{^{x}/_{u}\}]\,\tau) & \mathfrak{R} & \nu x.(\{^{x}/_{u}\} \mid \tau) & \\
(6) & \nu x.(\{^{x}/_{u}\} \mid [x=N\{^{x}/_{u}\}]\,\tau) & \mathfrak{R} & \nu x.(\{^{x}/_{u}\} \mid 0) & \text{if } N \neq_E u
\end{array}
$$

Figure 2.14: Defining conditions for the relation $\mathfrak{R}$ certifying $A \sim B$.

1. *(bisimulation)* Whenever $P\ \mathfrak{R}\ Q$, and $P \xrightarrow{\pi} P'$, there exists Q' such that $Q \xrightarrow{\pi} Q'$ and $P'\ \mathfrak{R}\ Q'$.
2. *(openness)* $\mathfrak{R}$ is closed under the application of a substitution fresh for the domain of the frame of any of the related states.
3. *(static equivalence)* Whenever $P\ \mathfrak{R}\ Q$, P is statically equivalent to Q.

We start by the obvious observation that $A\ \mathfrak{R}\ B$, given (1) in Fig. 2.14. Then we systematically check that the above condition holds.

Bisimulation. Since $\mathfrak{R}$ is by definition a symmetric relation, we provide proof only for the cases when the left-hand side process starts first. Below we present the exhaustive list of cases for the defining conditions of the relation $\mathfrak{R}$ in Fig. 2.14. Notice that in particular we have to accommodate the openness of $\mathfrak{R}$: if $P = \sigma\hat{A}$ can make a transition π to the state $P' = \sigma\hat{A}'$, there exists a state $\sigma\hat{B}'$ to which the process $Q = \sigma\hat{B}$ can make the transition π and $\sigma\hat{A}'\ \mathfrak{R}\ \sigma\hat{B}'$ where σ is fresh for the domain of the frame of P and $\hat{A}$, $\hat{A}'$, $\hat{B}$, $\hat{B}'$ are states listed in Fig. 2.14.

Let us make the following observation. The only free variable in A and B that can be affected by σ is a, i.e. σ is of the form $\left\{{}^{M,\vec{\mathcal{M}}}/_{a,\vec{\mathcal{A}}}\right\}$. Therefore the application of σ to any of the base states listed in Fig. 2.14 leads to the replacement of a with M, s.t. $M\ \#\ x$. To indicate the openness of $\mathfrak{R}$ we use M in the proof trees everywhere below. To obtain proof trees for base states it is enough to set $\sigma = id$ and $M = a$ below.

Case 1. $A \triangleq \nu x.\overline{a}\langle x\rangle.\,(a(y){+}a(y).\tau + a(y).\,[x=y]\,\tau)\ \mathfrak{R}\ \nu x.\overline{a}\langle x\rangle.\,(a(y){+}a(y).\tau) \triangleq B$

By definition $\mathfrak{R}$ is open, hence for an arbitrary substitution σ we have $A\sigma\ \mathfrak{R}\ B\sigma$.

Case 1.1. Let $\sigma\ \#\ u$. The process $A\sigma \triangleq \nu x.\overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau)$ can do the transition to $\overline{M}(u)$ to $A_y\sigma \triangleq \nu x.\,(\{^{x}/_{u}\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau)$ justified by the following proof tree.

$$
\dfrac{\dfrac{u\ \#\ M,\ x,\ M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau, id, x \quad M\,id =_E M}{x\colon \overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau) \xrightarrow{\overline{M}(u)} \{^{x}/_{u}\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau}\ \textsf{Out} \qquad x\ \#\ \emptyset, M, u}{\emptyset\colon \nu x.\overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau) \xrightarrow{\overline{M}(u)} \nu x.\,(\{^{x}/_{u}\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau)}\ \textsf{Res}
$$

There is a state $B_y\sigma \triangleq \nu x.\,(\{^{x}/_{u}\} \mid M(y){+}M(y).\tau)$ to which the process $B\sigma \triangleq \nu x.\overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau)$ can make the transition $\overline{M}(u)$ justified by the following proof tree.

$$\dfrac{\dfrac{u \# M,\, x,\, M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau, id, x \quad M\,id =_E M}{x\colon \overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau) \xrightarrow{\overline{M}(u)} \{^x/_u\} \mid M(y){+}M(y).\tau}\,\textsc{Out} \qquad x \# \emptyset, M, u}{\emptyset\colon \nu x.\overline{M}\langle x\rangle.\,(M(y){+}M(y).\tau{+}) \xrightarrow{\overline{M}(u)} \nu x.\,(\{^x/_u\} \mid M(y){+}M(y).\tau)}\,\textsc{Res}$$

By (2) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A_y\sigma\ \mathfrak{R}\ B_y\sigma$.

Case 1.2 Let σ is s.t. it is not fresh for u. Then, to satisfy the conditions of the Out rule, we record the output using another alias, e.g. $u' \# M$ and the respective transition label is $\overline{M}(u')$. The rest is identical to case 1.1.1 – it is enough to replace u with u'. Notice that by the definition of $\mathfrak{R}$ we have the following.

$$\nu x.(\{^x/_{u'}\} \mid a(y){+}a(y).\tau + a(y).\,[x=y]\,\tau)\ \ \mathfrak{R}\ \nu x.(\{^x/_{u'}\} \mid a(y){+}a(y).\tau)$$

Case 2. $\nu x.(\{^x/_u\} \mid a(y){+}a(y).\tau + a(y).\,[x=y]\,\tau)\ \mathfrak{R}\ \nu x.(\{^x/_u\} \mid a(y){+}a(y).\tau)$.

Let us call the left and the right parts related by $\mathfrak{R}$ as A_y and B_y respectively. By definition $\mathfrak{R}$ is open, hence for the substitution $\sigma \# u$ we have $A_y\sigma\ \mathfrak{R}\ B_y\sigma$.

Since the process $A_y\sigma \triangleq \nu x.(\{^x/_u\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau)$ contains the choice, it can make the transition $a\,N$, where N is any permitted input to several different states. We consider each possible case below.

Case 2.1. The process $A_y\sigma$ can make the transition $M\,N\sigma$, where N is any permitted input to the state $A_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{M =_E M}{x\colon \{^x/_u\} \mid M(y) \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\,\textsc{Inp}}{x\colon \{^x/_u\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\,\textsc{Sum-L} \qquad x \# \emptyset, M, N\sigma}{\emptyset\colon \nu x.\,(\{^x/_u\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau) \xrightarrow{MN\sigma} \nu x.(\{^x/_u\} \mid 0)}\,\textsc{Res}$$

There exists a state $B_0\sigma \triangleq \nu x.(\{^x/_u\})$ to which the extended process $B_y\sigma \triangleq \nu x.\,(\{^x/_u\} \mid M(y){+}M(y).\tau)$ can make the transition $M\,N\sigma$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{M =_E M}{x\colon \{^x/_u\} \mid M(y) \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\,\textsc{Inp}}{x\colon \{^x/_u\} \mid M(y){+}M(y).\tau \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\,\textsc{Sum-L} \qquad x \# \emptyset, M, N\sigma}{\emptyset\colon \nu x.\,(\{^x/_u\} \mid M(y){+}M(y).\tau) \xrightarrow{MN\sigma} \nu x.(\{^x/_u\} \mid 0)}\,\textsc{Res}$$

By (3) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A_0\sigma\ \mathfrak{R}\ B_0\sigma$.

Case 2.2. The process $A_y\sigma$ can make a transition $M\,N\sigma$, where N is any permitted input, to the state $A_\tau\sigma \triangleq \nu x.(\{^x/_u\} \mid \tau)$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{M =_E M}{x\colon \{^x/_u\} \mid M(y).\tau \xrightarrow{MN\sigma} \{^x/_u\} \mid \tau}\,\textsc{Inp}}{x\colon \{^x/_u\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau \xrightarrow{MN\sigma} \{^x/_u\} \mid \tau}\,\textsc{Sum-L} \qquad x \# \emptyset, M, N\sigma}{\emptyset\colon \nu x.\,(\{^x/_u\} \mid M(y){+}M(y).\tau + M(y).\,[x=y]\,\tau) \xrightarrow{MN\sigma} \nu x.(\{^x/_u\} \mid \tau)}\,\textsc{Res}$$

There exists a state $B_\tau\sigma \triangleq \nu x.(\{x/u\} \mid \tau)$ to which the extended process $B_y\sigma \triangleq \nu x.\,(\{x/u\} \mid a(y){+}a(y).\tau)$ can make the transition $M\,N\sigma$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{M =_E M}\,\textsc{Inp}}{x\colon \{x/u\} \mid M(y).\tau \xrightarrow{MN\sigma} \{x/u\} \mid \tau}\,\textsc{Sum-L}}{x\colon \{x/u\} \mid M(y){+}M(y).\tau \xrightarrow{MN\sigma} \{x/u\} \mid \tau \qquad x \# \emptyset, M, N\sigma}\,\textsc{Res}}{\emptyset\colon \nu x.\,(\{x/u\} \mid M(y){+}M(y).\tau) \xrightarrow{MN\sigma} \nu x.(\{x/u\} \mid \tau)}$$

By (4) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A_\tau\sigma\ \mathfrak{R}\ B_\tau\sigma$.

Case 2.3. If the process $A_y\sigma$ takes the third branch, the choice of the input affects the further behaviour of the process. We will make a distinction between the cases $N\sigma =_E u$ and $N\sigma \neq_E u$.

Case 2.3.a. The process $A_y\sigma$ can make the transition $M\,N\sigma$, where N is s.t. $N\sigma =_E u$ to the state $A^a_{\text{match}}\sigma \triangleq \nu x.(\{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau)$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{M =_E M}\,\textsc{Inp}}{x\colon \{x/u\} \mid M(y).[x = y]\,\tau \xrightarrow{MN\sigma} \{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau}\,\textsc{Sum-L}}{x\colon \{x/u\} \mid M(y){+}M(y).\tau + M(y).[x = y]\,\tau \xrightarrow{MN\sigma} \{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau \qquad x \# \emptyset, M, N\sigma}\,\textsc{Res}}{\emptyset\colon \nu x.\,(\{x/u\} \mid M(y){+}M(y).\tau + M(y).[x = y]\,\tau) \xrightarrow{MN\sigma} \nu x.(\{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau)}$$

Notice that in Sum-L we have used the following observation: $N\sigma\{x/u\} = N\{x/u\}\sigma$. This is true since σ by assumption contains no u in the domain or the range and has no effect on x since x is a bound variable. Therefore the set of free variables in N affected by $\{x/u\}$ does not intersect the set of free variables in N affected by σ and the result of the application of σ and $\{x/u\}$ to N does not depend on the order of application of substitutions.

There exists a state $B_\tau\sigma \triangleq \nu x.(\{x/u\} \mid \tau)$ to which the extended process $B_y\sigma \triangleq \nu x.(\{x/u\} \mid M(y){+}M(y).\tau)$ can make the transition $M\,N\sigma$, where N is s.t. $N\sigma =_E u$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{M =_E M}\,\textsc{Inp}}{x\colon \{x/u\} \mid M(y).\tau \xrightarrow{MN\sigma} \{x/u\} \mid \tau}\,\textsc{Sum-L}}{x\colon \{x/u\} \mid M(y){+}M(y).\tau \xrightarrow{MN\sigma} \{x/u\} \mid \tau \qquad x \# \emptyset, M, N\sigma}\,\textsc{Res}}{\emptyset\colon \nu x.\,(\{x/u\} \mid M(y){+}M(y).\tau) \xrightarrow{MN\sigma} \nu x.(\{x/u\} \mid \tau)}$$

By (5) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A^a_{\text{match}}\sigma\ \mathfrak{R}\ B_\tau\sigma$.

Case 2.3.b. The process $A_y\sigma$ can make the transition $M\,N\sigma$, where N is s.t. $N\sigma \neq_E u$ to the state $A^b_{\text{match}}\sigma \triangleq \nu x.(\{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau)$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{M =_E M}\,\textsc{Inp}}{x\colon \{x/u\} \mid M(y).[x = y]\,\tau \xrightarrow{MN\sigma} \{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau}\,\textsc{Sum-L}}{x\colon \{x/u\} \mid M(y){+}M(y).\tau + M(y).[x = y]\,\tau \xrightarrow{MN\sigma} \{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau \qquad x \# \emptyset, M, N\sigma}\,\textsc{Res}}{\emptyset\colon \nu x.\,(\{x/u\} \mid M(y){+}M(y).\tau + M(y).[x = y]\,\tau) \xrightarrow{MN\sigma} \nu x.(\{x/u\} \mid [x = N\{x/u\}\sigma]\,\tau)}$$

Notice that in SUM-L we have also used the following observation: $N\sigma\{^x/_u\} = N\{^x/_u\}\sigma$, as in 2.2.3.a.

There exists a state $B_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ to which the extended process $B_y\sigma \triangleq \nu x.\,(\{^x/_u\} \mid M(y)+M(y).\tau)$ can make the transition $M\,N\sigma$, where N is s.t. $N\sigma \neq_E u$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{x\colon \{^x/_u\} \mid M(y) \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\ \text{Inp}}{x\colon \{^x/_u\} \mid M(y)+M(y).\tau \xrightarrow{MN\sigma} \{^x/_u\} \mid 0}\ \text{Sum-L} \qquad x \# \emptyset, M, N\sigma}{\emptyset\colon \nu x.\,(\{^x/_u\} \mid M(y)+M(y).\tau) \xrightarrow{MN\sigma} \nu x.(\{^x/_u\} \mid 0)}\ \text{Res}$$

By (6) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A^b_{\text{match}}\sigma\ \mathfrak{R}\ B_0\sigma$.

Case 3. $A_0 \triangleq \nu x.(\{^x/_u\} \mid 0)\ \mathfrak{R}\ \nu x.(\{^x/_u\} \mid 0) \triangleq B_0$.

Notice that any substitution $\sigma \# u$ has no effect on A_0 and B_0: $A_0\sigma = A_0$, $B_0\sigma = B_0$.

There are no valid transitions that the process $A_0\sigma$ can make since $A_0\sigma$ is deadlocked. There is no state to which the process $B_0\sigma$ can make a transition since $B_0\sigma$ is deadlocked.

Case 4. $A_\tau \triangleq \nu x.(\{^x/_u\} \mid \tau)\ \mathfrak{R}\ \nu x.(\{^x/_u\} \mid \tau) \triangleq B_\tau$.

By definition $\mathfrak{R}$ is open, hence for a substitution $\sigma \# u$ we have $A_\tau\sigma\ \mathfrak{R}\ B_\tau\sigma$.

The process $A_\tau\sigma \triangleq \nu x.(\{^x/_u\} \mid \tau)$ can make the transition τ to the state $A_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$. There is a state $B_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ to which the process $B_\tau\sigma \triangleq \nu x.(\{^x/_u\} \mid \tau)$ can make the transition τ.

By (3) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A_0\sigma\ \mathfrak{R}\ B_0\sigma$.

Case 5. $A^a_{\text{match}} \triangleq \nu x.(\{^x/_u\} \mid [x = N\{^x/_u\}]\,\tau)\ \mathfrak{R}\ \nu x.(\{^x/_u\} \mid \tau) \triangleq B_\tau$, and $N =_E u$.

By definition $\mathfrak{R}$ is open, hence for a substitution $\sigma \# u$ we have $A^a_{\text{match}}\sigma\ \mathfrak{R}\ B_\tau\sigma$.

The process $A^a_{\text{match}}\sigma \triangleq \nu x.(\{^x/_u\} \mid [x = N\{^x/_u\}\sigma]\,\tau)$ can make the transition τ to the state $A_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ justified by the following proof tree.

$$\dfrac{\dfrac{\dfrac{}{x\colon \{^x/_u\} \mid \tau \xrightarrow{\tau} \{^x/_u\} \mid 0} \qquad x =_E N\{^x/_u\}\sigma\text{ , since } N\sigma =_E u}{x\colon \{^x/_u\} \mid [x = N\{^x/_u\}\sigma]\,\tau \xrightarrow{\tau} \{^x/_u\} \mid 0}\ \text{Match} \qquad x \# \emptyset, n(\tau)}{\emptyset\colon \nu x.(\{^x/_u\} \mid [x = N\{^x/_u\}\sigma]\,\tau) \xrightarrow{\tau} \nu x.\,(\{^x/_u\} \mid 0)}\ \text{Res}$$

There is a state $B_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ to which the process $B_\tau\sigma \triangleq \nu x.(\{^x/_u\} \mid \tau)$ can make the transition τ justified by the following proof tree.

$$\dfrac{\dfrac{}{x\colon \{^x/_u\} \mid \tau \xrightarrow{\tau} \{^x/_u\} \mid 0} \qquad x \# \emptyset, n(\tau)}{\emptyset\colon \nu x.(\{^x/_u\} \mid \tau) \xrightarrow{\tau} \nu x.(\{^x/_u\} \mid 0)}\ \text{Res}$$

By (3) in Fig. 2.14 and the openness of $\mathfrak{R}$ we have $A_0\sigma\ \mathfrak{R}\ B_0\sigma$.

Case 6. $A^b_{\text{match}} \triangleq \nu x.(\{^x/_u\} \mid [x = N\{^x/_u\}]\,\tau)\ \mathfrak{R}\ \nu x.(\{^x/_u\} \mid 0) \triangleq B_0$, and $N \neq_E u$.

By definition $\mathfrak{R}$ is open, hence for a substitution $\sigma \# u$ we have $A^b_{\text{match}}\sigma\ \mathfrak{R}\ B_0\sigma$.

Given the condition $N \neq_E u$ there is no state to which the process $A^b_{\text{match}}\sigma \triangleq \nu x.(\{^x/_u\} \mid [x = N\{^x/_u\}\sigma]\,\tau)$ can make a transition.

There is no state to which the process $B_0\sigma \triangleq \nu x.(\{^x/_u\} \mid 0)$ can make a transition since it is deadlocked either.

Openness Above we have already considered the general case where the appropriate substitution is applied to either side of the relation. The proof trees presented illustrate that it is enough to consider transitions for the core cases comprising the proof certificate, and to check that accessibility mechanism as in Def. 4 does not invalidate these proof trees.

Static equivalence. We conclude the proof by demonstrating that any two processes related by $\mathfrak{R}$ are statically equivalent. There are only two cases to consider.

Case 1. $\nu x.\overline{a}\langle x\rangle.\,(a(y)+a(y).\tau + a(y).\,[x = y]\,\tau)\ \ \mathfrak{R}\ \nu x.\overline{a}\langle x\rangle.\,(a(y)+a(y).\tau)$. The substitution of the process on the left considered as a state consists of identity substitution id. The same holds for the process on the right. Therefore for any pair of messages M and N such that $\vec{x}\ \#\ M, N$ the equality $Mid =_E Nid$ holds if and only if $Mid =_E Nid$. Static equivalence also holds if a substitution σ, fresh for the domain of id, is applied to the related processes since such substitution doesn't affect id.

Case 2. Any extended process from (2)-(6) in Fig. 2.14 related by $\mathfrak{R}$ contains the substitution $\theta := \{^{x}/_{u}\}$. Therefore for any pair of messages M and N such that $M, N\ \#\ \vec{x}$ the equality $M\theta =_E N\theta$ holds if an only if $M\theta =_E N\theta$. Static equivalence still holds for any pair of extended processes in the lines (2)-(6) obtained by the application of the substitution σ fresh for the domain of θ (which is u) since such σ does not affect the substitution θ as x is a bound variable.

The example above demonstrates how one verifies that two processes are quasi-open bisimilar. It requires two main steps: (*i*) defining a candidate relation like in Fig. 2.14, and then (*ii*) checking that this relation indeed satisfies the definition for quasi-open bisimulation 5. We would like to draw the attention of the reader that the ingenious step is to provide a candidate relation while verifying it is a quasi-open bisimulation is tedious yet repetitive. In the next chapters, where we consider processes with replication (thereby representing infinitely many runs of the protocol) that exchange messages subject to a non-trivial equational theory, the reader will notice that the complexity of the step *i* increases, while the step *ii* still remains repetitive and mechanical, which yet again justifies the importance of proof certificates certifying two processes are indistinguishable.

2.3 Quasi-open bisimilarity, an elevator pitch

In this chapter we have introduced and justified the bisimilarity congruence called quasi-open bisimilarity (Def. 5) as a method for reasoning about protocols expressed using the applied π-calculus. A key innovation in the definition of quasi-open bisimilarity over classical labelled bisimilarity is that our relation is open (Def. 4), which endows an attacker with a power to systematically consider different execution environments by manipulating messages that depend on free variables. Perhaps surprisingly, it gives rise to two seemingly unrelated properties of quasi-open bisimilarity.

- Firstly, the openness endows quasi-open bisimilarity with a sound and complete characterisation by a certain modal logic as we explain in Sec. 2.2.3, where the completeness as in Theorem 2 is especially valuable since it ensures that we can describe attacks when the verification fails. This means that the logic is capable of describing all attacks on such privacy properties, even if a coarser equivalence is employed. For the counterexample, we explain in Sec. 2.2.5 why the failure of a privacy property in terms of a finer equivalence, such as open bisimilarity, does not necessarily indicate the presence of an attack strategy.

- Secondly, the openness makes quasi-open bisimilarity congruence (Theorem 3), which enables compositional reasoning allowing us to reduce the amount of work needed for verification by reducing to a smaller system, which we have demonstrated with the example of a generic authentication protocol without shared secrets in Theorem 4.

Lastly, in Sec. 2.2.6, we explain that modal logic formulas and quasi-open bisimulations serve as the basis of proof certificates which can be valuable for automation that we see as the main direction for future work. In the next two chapters we offer two detailed checks of certificates witnessing that a privacy property holds. We will see that the ingenious step is to provide a candidate for quasi-open bisimulation, while checking that it satisfies three conditions in Def. 5 is somewhat mechanical.

Chapter 3

Case study: smartcard-based payments

In the following chapter, we develop and verify a privacy-preserving smartcard-based payment protocol. This discussion, however, would not be complete without explaining EMV, the most prevalent [emv22] payment method using smartcards – as of 2022 more than 91% of transactions are EMV. In this chapter we introduce the basics of EMV: we describe the stages of a typical EMV transaction and present an attack, allowing to bypass PIN verification, thereby demonstrating that to this date EMV cannot be called secure. We also show, that EMV lacks privacy either, as the communication between the card and the terminal is not encrypted, allowing the identifying information about the card to be exposed during the transaction. Finally, we discuss the Blinded Diffie-Hellman (BDH) protocol, a privacy enhancement proposed by the developers of the standard, show that BDH does not protect the privacy of the cardholder in the presence of realistic active attackers, and propose a minor enhancement of BDH that does account for active attackers.

As a payment method, EMV came into place in the mid-1990s to replace magnetic stripe cards as they are incapable of computation and easy to clone. Payment providers such as American Express, Banrisul, Dankort, JCB, Mastercard, MIR, UnionPay, Visa, etc. follow the EMV standard [emv11], a series of documents that specify how exactly payments should be done with the main focus on card-terminal communication. The standard is maintained by EMVCo, a consortium of payment processing companies that include several ones from the above list. While there are other than EMV smartcard payment methods, e.g. parking lot cards or top-up shopping centre cards, their use is limited, and they follow proprietary protocols – in contrast, EMV is widely accepted and open. The standard is quite flexible – only minimal requirements must be respected, so it is up to the payment system that implements EMV, which additional options to include. Hence, the standard describes not a single protocol but a whole variety of configurations.

To start off, we introduce transaction flow and the respective infrastructure assumed by EMV in Fig. 3.1. The card C is manufactured by the *issuing bank* BC in collaboration with the payment system *PaySys* (e.g. Visa). The terminal T is connected to an *acquiring bank* BT supporting *PaySys* that processes payments on behalf of the terminal. The acquiring bank BT processes payments by connecting to the *PaySys* network that exchanges messages between banks.

A successful run of the protocol results in the generation of an Application Cryptogram AC by the card C. AC is eventually sent by the terminal T to the

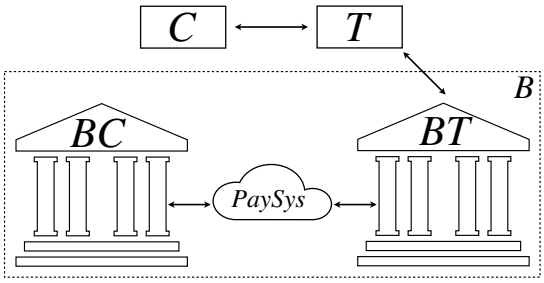


Figure 3.1: Payment architecture.

acquiring bank *BT*, either before or after the payment is approved by the terminal, depending on whether the payment is online or offline, respectively. The issuing bank *BC* receiving *AC*, decides to decline or accept the transaction, and replies with the appropriate message. Since the processing method on the banks' side specified in the standard is not mandatory as "issuers may decide to adopt other methods" [emv11, Book 2, Section 8], and the internal processing used by *PaySys* is proprietary and *PaySys*-specific, when modelling the system, *BT*, *PaySys*, and *BC* are often merged into a single agent *B*, modelling their common interface with the terminal when processing payments, as indicated in Fig. 3.1.

It was shown several times that some configurations allowed by the standard are not secure [DM07, MDAB10, BST21b, BST21a, RCN+22], making fraudulent transactions possible. The important aspect of EMV is that it also supports contactless cards, making it easier for a man-in-the-middle attacker to interact with the card without the cardholder being aware, thereby lowering the sophistication of, e.g. eavesdropping or relaying[1] the communication between the card and the terminal. Examples of attacks on EMV leading to fraudulent payment include PIN bypassing [MDAB10, BST21b, BST21a], downgrade attacks [MDAB10], pre-play attacks [BCM+14, BCM+15], etc. The majority of attacks alike are rooted in the failure of the selected EMV configuration to deliver authentication guarantees, i.e. messages between the parties can be altered on the way leading to different views on the same transaction. In contrast, relay attacks [DM07, BCDD20], where messages can be unaltered, but simply relayed (e.g. between the card in the wallet and the terminal), stand out and require the introduction of distance-bounding techniques [CdRS18, MSTPTR18], s.t. the parties are guaranteed to be close in order to execute the transaction. Finally, relaying can be combined with a man-in-the-middle attack to execute a fraudulent payment, as has been demonstrated by, e.g. Radu et al. in [RCN+22] where they describe how to bypass the Apple Pay (working with Visa card) lock screen. The term skimming is often used to describe the situation when an attacker secretly activates a contactless card and communicates with it. A skimming attack may be a part of a relay attack and will serve as the basis of the attack on the unlinkability for the EMV privacy update we discuss later in this chapter. Habraken et al. constructed an antenna in the form of a gate of up

[1] Being contactless, however, is not a requirement for passive eavesdropping, or active man-in-the-middle attacks. In case of contact transactions, an unnoticeable "shim" properly installed inside the reader is enough for the eavesdropping/transactions data collection [BCM+14]; suitable devices that can alter messages and, for instance, trick the terminal that the right PIN was entered when the PIN is unknown, exist [MDAB10].

to 100cm width that can power the card and communicate with it [HDPdR15]. For a passive counterpart of skimming, an *eavesdropping attack*, that does not require powering up a contactless card, Engelhardt et al. achieved a distance of almost 20m [EPFB13].

Both eavesdropping and skimming play an important role when it comes to the privacy of EMV payments. To this day, the communication between the card and the terminal is not encrypted and valuable sensitive data, as we will see below, such as the card number PAN (Primary Account Number), the amount, the country code, and the time, are exposed, and an attacker eavesdropping on wireless communications can profile cardholders engaged in transactions. In addition, an active attacker can power up a contactless card without a cardholder being aware using an antenna [HDPdR15], e.g., at a doorway or by a seat on public transport. A powered-up card is ready to start an EMV session and to present its PAN – a strong form of identity. This enables an attacker to silently track the movements of anyone who holds a payment card, even without a genuine EMV transaction involved. In EMV both passive eavesdroppers and active skimmers have access to PAN and can link sessions made with the same card, exactly as in the unlinkability example in Fig. 2.1 we have started the privacy discussion with in the previous chapter. In addition, since an attacker is able to observe the card's identity PAN at each run of the protocol, the anonymity, as we have discussed briefly in Sec. 3.4 is also violated.

Such total lack of privacy should not come up as a surprise, as the primary objective of EMV is the security of money in the cardholder's account, not privacy. However, even to achieve this primary goal, one should carefully select a secure configuration and avoid insecure ones.

In Sec. 3.2 we will demonstrate yet another insecure configuration that is still permitted by the standard, however, firstly, we describe the main stages of a generic EMV transaction in Sec. 3.1. Finally, in Sec. 3.3 and Sec. 3.5 we apply techniques described in Chapter 2 to formally show that the key establishment proposed by EMVCo to enhance the privacy of payments [rfc12], does not protect from active attackers; and that it is possible to update this key establishment to mitigate them.

3.1 EMV standard overview

The official specification of the EMV protocol is about 2000 pages long [emv11] and is spread across multiple volumes. For the sake of completeness, in this section we give a high-level description of a generic EMV transaction. We warn the reader that our goal is to demonstrate how convoluted the EMV standard is, and how easily an insecure configuration can be picked up for implementation; rather than giving a comprehensive description of all possible configurations of EMV. For the latter, we refer the reader to the technical report [vdBOYPdR16] by van den Breekel et al., where such description is given in a clear and concise form that still takes 33 pages of text. More than one hundred citations [emv] of this report back up the claim made by the authors in the abstract: "it is hard to grasp the essence of the protocols from the – long and complex – official specifications".

We reiterate that the EMV standard mainly specifies communications between the card and the terminal. This communication is a series of Application Protocol

Data Unit (APDU) command/response exchanges. Each command may come with a payload that either the expected data, specified by the protocol, or, the data requested on a previous step with a so-called Data Object List (DOL), a list of data elements that must be sent in response. For instance, using PDOL, as we will see below in detail, a card may request specific transaction data that the terminal should present.

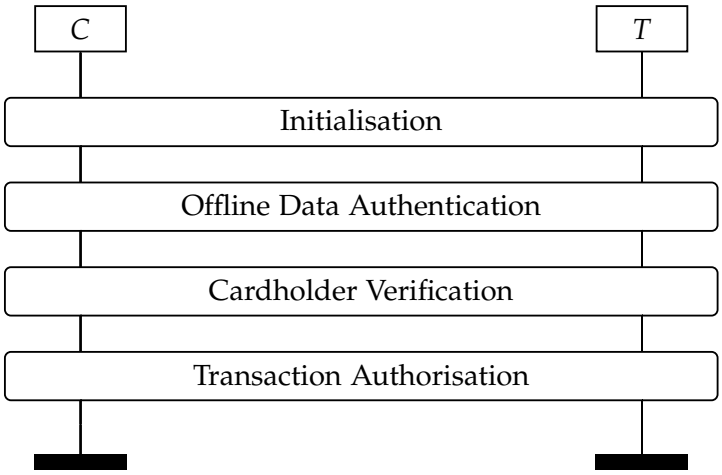


Figure 3.2: The EMV 1st Gen protocol stages.

Cryptography cheat sheet. This overview is informal, but for the sake of completeness we recall three main cryptographic primitives the reader will encounter in this section and provide the respective equations. Asymmetric encryption is already formalised in Sec. 2.1.1, and assumes two algorithms: encryption, which produces an encrypted message from a given message and a public key, and decryption which produces the message from the given encrypted message and the respective secret key.

$$\mathtt{adec}(\mathtt{aenc}(M, \mathtt{pk}(K)), K) =_E M$$

Digital signature also assumes two algorithms: signing, which produces the signature from the given message and a secret key, and checking which produces the message from the signature and the respective public key. Hence, we need two ingredients for signature verification: the original message and the signature itself. The signature is verified if the message coincides with the result of applying the `check` function to the signature using the appropriate public key. In what follows, the *certificate* stands for the message-signature pair.

$$\mathtt{check}(\mathtt{sig}(M, K), \mathtt{pk}(K)) =_E M$$

Finally, we have Hash-Based Message Authentication Code (HMAC) that requires a pre-shared secret. HMAC of a message is the hash of a message paired with the secret $M_{hmac} = \mathtt{h}(\langle M, sk \rangle)$. To verify the authenticity of the message M upon receiving $\langle M, M_{hmac} \rangle$, one checks that the message M hashed with the secret (the receiving party already has) coincides with M_{hmac}, the second element of the received pair.

A transaction consists of at most four phases presented in Fig. 3.2 among which only the Initialisation, where the card receives the transaction details, and, the Transaction Authorisation, where the card generates the *AC* are mandatory. Optional intermediate steps include Offline Data Authentication(ODA), where the terminal validates the card using the public key of the payment system, and, Cardholder Verification, where the terminal verifies the cardholder via, e.g. the PIN entered in the terminal's pad. The information the card permanently stores include several certificates, the signing secret key c, and the shared secret mk with the bank (often called a master key). Other information the card may store will be clear from the context below. The terminal only stores the payment system's public key $\texttt{pk}(s)$ used to verify the data on the card. Now we will take a closer look at each step and summarise the security and privacy issues that may arise.

3.1.1 Initialisation

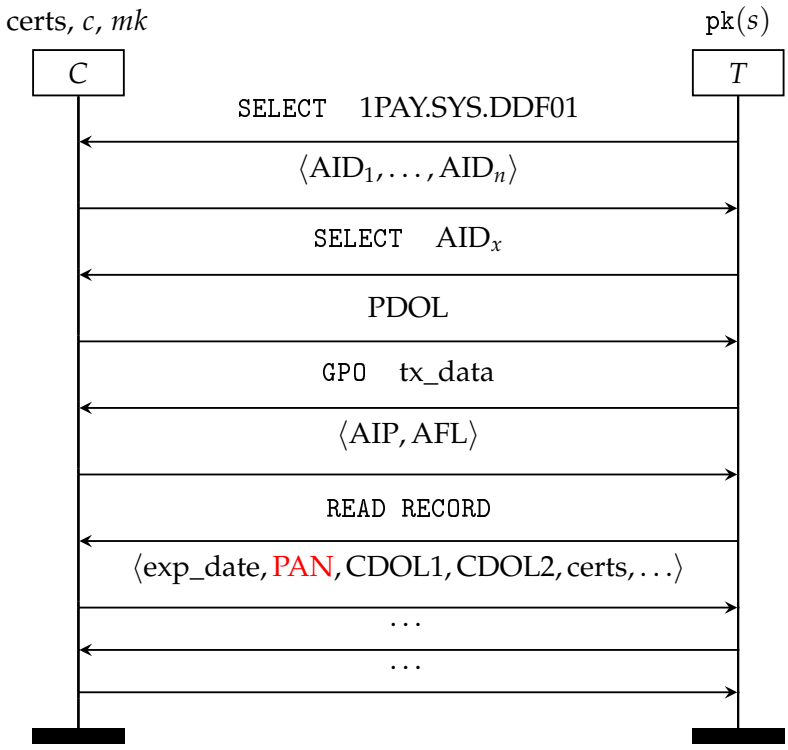


Figure 3.3: Initialisation of the EMV protocol.

The initialisation phase is presented in Fig. 3.3. Firstly, the terminal asks which applications the card supports by issuing the `SELECT` command with a payload depending on the transaction type – if the transaction is contact, the `1PAY.SYS.DDF01` message is added to the command, if the transaction is contactless, `2PAY.SYS.DDF01` is added. The card responds with a list of payment application identifiers (AID), e.g. Visa Debit, Maestro, etc. Then the terminal selects a particular application from the presented list.

Having selected the application, the card sends the PDOL specifying which transaction details (e.g. the amount, the date, the currency, etc.) the terminal should

send to the card. Next, in one message, the terminal sends the requested PDOL data and requests the AIP list, specifying the functions supported by the card including authentication methods and the AFL list, specifying memory addresses where the card stores the data needed to complete the transaction. Finally, the terminal uses the addresses from the AFL list to read the actual data from the card. Only the following is mandatory for the card to have as specified in the standard [emv11, Book 3, Section 7].

- Application Expiry Date
- The card number PAN
- Card Risk Management Data Object Lists (CDOLs)

The terminal uses the expiry date to check that the card is valid at the time of the purchase, the PAN is used to route the AC through the network to the issuing bank at the last stage of the transaction, and CDOLs specify the information the card needs from the terminal to generate the AC, e.g. the country code, the terminal nonce, etc. Typically, certain certificates, digitally signed data, that the terminal should check using the appropriate public key is also among the data indicated by the AFL for the terminal to then read.

Since in what follows, the use of certain secret keys and the corresponding public keys will be important, we explain the so-called chain of certificates that cards usually have among the data retrieved by the terminal. Let the payment system, the issuing bank, and the card hold the secret keys s, b, and c, respectively. Also, let the public key of the payment system be already loaded in the terminal supporting this payment system such that the terminal can use it to check signatures issued using s. The chain of certificates, denoted as *certs* in figures in this section, comprised of two ingredients.

- The certificate on the issuing bank's public key $\mathtt{pk}(b)$ signed with the private key of the payment system s.
- The certificate on the card's public key $\mathtt{pk}(c)$ signed with the private key of the issuing bank b.

The terminal uses these certificates to verify the legitimacy of the card. Firstly it checks the bank's certificate using the public key of the payment system, thereby ensuring that the bank's public key is legitimate. Secondly, it checks the car's certificate using the bank's public key from the bank's certificate, thereby ensuring that the card's public key is legitimate.

Notice that none of the data provided by the card is authenticated at this stage and will be explained below. From the privacy standpoint, we can see already that a lot of sensitive data is exposed to the environment as the communication is in cleartext. Data elements that are unique only to a group of cards such as the list of supported applications, or the AIP, contribute to the card's fingerprint, thus enabling profiling. Data elements that are unique to a card, such as the PAN or the chain of certificates *certs* can be used to link sessions made with the same card, thus violating both anonymity and unlinkability as in Fig. 2.1.

3.1.2 Offline data authentication

Offline Data Authentication (ODA) is an optional step in the EMV protocol at which point the terminal authenticates the data previously received from the card during the Initialisation step. The ODA can be completed in several ways. We briefly describe each.

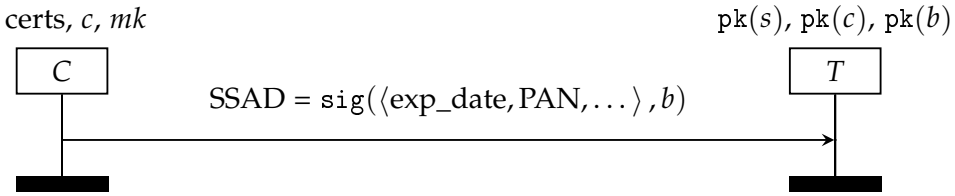


Figure 3.4: ODA: Static Data Authentication mode.

Static Data Authentication (SDA). SDA is schematically presented in Fig. 3.4. In this mode the terminal uses the bank's public key to check the signature on the card's data retrieved by the terminal at the initialisation step (indicated by the AFL) issued with the bank's secret key b. Security-wise, the authentication property called *aliveness* fails if SDA is selected – as the signature is constant, from one session to the next, after executing the protocol, it is not necessarily the case that the real card was alive and communicating, thereby the use of SDA enables card cloning as both the card's data, and the signature is exposed. Privacy-wise, since the signature is unique to a card, it can be used to link sessions with that card.

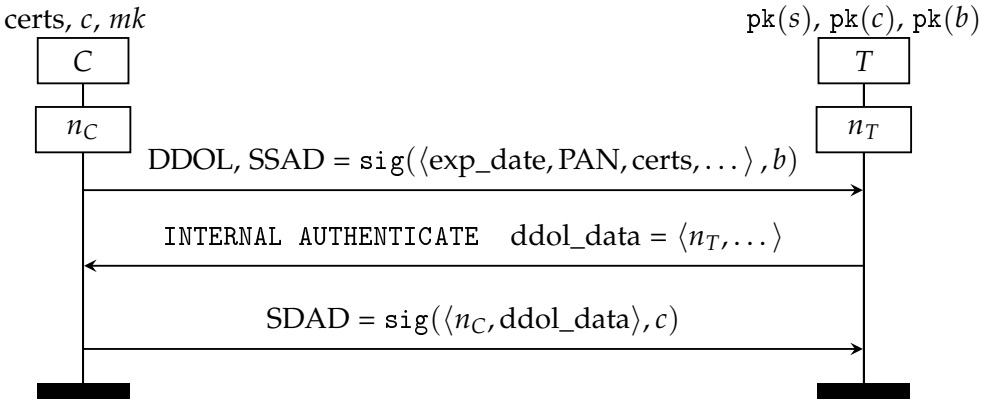


Figure 3.5: ODA: Dynamic Data Authentication mode.

Dynamic Data Authentication (DDA). DDA mode is presented schematically in Fig. 3.5. It mitigates the card cloning vulnerability of SDA and is essentially a terminal's check of a signature generated by the card over certain dynamic data. At the initialisation phase, the card could provide the DDOL indicating the data elements that the terminal must send to the card if DDA is selected. The only

mandatory element that the terminal should provide (even if the DDOL is not present) is a nonce n_T. To perform DDA the terminal firstly authenticates the AFL data, as in SDA, secondly, it provides the DDOL data together with issuing the `INTERNAL AUTHENTICATE` command to the card and thirdly, it uses the card's public key to check the signature generated with the card's secret key c over the provided DDOL data and the card's dynamic data, typically including the card's nonce n_C. In EMV lingo the data with the card's signature over it is called SDAD (Signed Dynamic Authentication Data).

In contrast to SDA, the signature checked uses the session-specific data generated by both parties during the session, hence cannot be forged and used in card cloning. From the privacy perspective, while the DDOL (or the lack of) identifies the card to a degree and contributes to the card's fingerprint.

Combined Data Authentication (CDA). This mode is similar to DDA, however, it does not require additional messages. The signature, in this case, is generated by the card over the card's nonce and includes transaction-specific data, e.g. the AC and is checked by the terminal during the Transaction Authorisation phase explained below in Sec. 3.1.4.

We conclude by summarising or main security and privacy observations for this step of an EMV transaction. SDA is vulnerable to card cloning, and each ODA mode requires the data unique to a card being revealed – in case SDA is selected, the static signature that includes the mandatory PAN is unique, and in case DDA/CDA is selected, the card's unique public key (transferred at the initialisation) is required to check the card-generated signature. Given that the communication is in cleartext, even an eavesdropper can recognise if the same card is used across different sessions and, thereby, track the cardholder.

3.1.3 Cardholder Verification

Cardholder Verification is an optional step in the current EMV transaction process intended to prove that the person using the card is indeed a legitimate cardholder. Cards supporting cardholder verification provide the Cardholder Verification Methods list CVM_list that the card supports in the Initialisation step. The main security issue here is that it is not required by the EMV standard, that the CVM_list is part of the static data to be authenticated, hence, if not, it can be altered by man-in-the-middle with the goal to bypass cardholder verification[2]. Privacy-wise CVM_list varies from card to card, hence contributing to its fingerprint. Verification methods include a handwritten signature, PIN, and verification via the consumer's device (e.g. through biometric data entered via a mobile phone), which is out of the scope of the standard. An important dimension of the cardholder verification phase is who is held liable for the disputed transaction. In EMV, it depends on the method – if the cardholder was verified using the paper signature, the merchant is liable, if instead, PIN has been used, the cardholder is liable. Below we describe PIN-based CVMs.

[2]The card brand mixup attack [BST21a], however, demonstrates that even if this list is authenticated, it can be downgraded to offline cleartext PIN-only list.

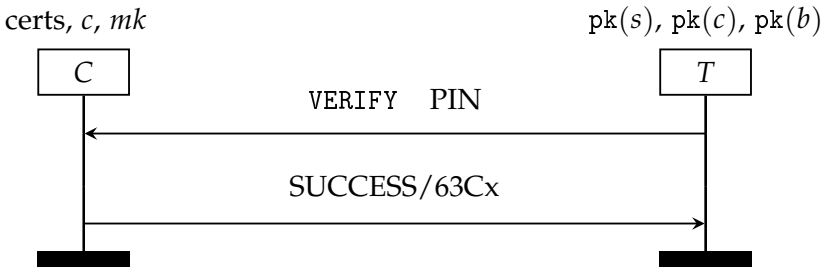


Figure 3.6: CVM: Offline cleartext PIN.

Offline Cleartext PIN. In this mode, the PIN, entered into the terminal by the cardholder, is sent to the card in clear text together with the `VERIFY` command as indicated in Fig. 3.6. If the PIN is correct, the card responds with a constant message indicating success. Otherwise, the card responds with `63Cx` message, where x is the number of tries left.

The first security threat here is that the PIN is exposed to eavesdroppers, making it straightforward to use the stolen/cloned card or relay messages from someone's pocket to a terminal to make payment. We stress the passive nature of the eavesdroppers here, as entering the PIN assumes a user action and the awareness of the cardholder being in the middle of a purchase. In contrast, the PIN is not among the data that an attacker secretly activating the card can silently gather. The second security threat is that the card's response is not authenticated, making it possible for a man-in-the-middle attacker to lie about the PIN verification outcome. At the same time, while the PIN is not necessarily unique to a card, the likelihood of transactions in which the same PIN is used are made with the same card is high, hence the PIN could be used to link sessions and track the cardholder. Moreover, since the response may contain the number of tries left (if the entered PIN is wrong), it also reveals the identifying information about the card – not all cards have their try counter synchronised.

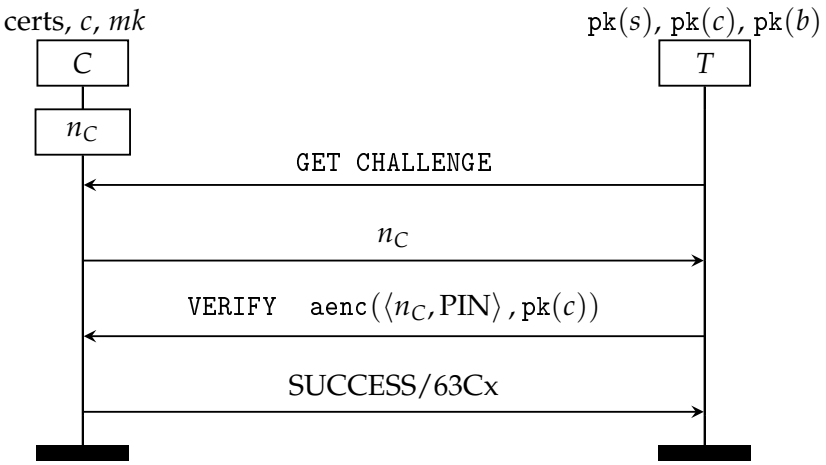


Figure 3.7: CVM: Offline encrypted PIN mode. The digit x is the number of tries left.

Offline Encrypted PIN. The PIN leak vulnerability of the above Offline Cleartext PIN CVM can be addressed by encrypting the PIN, as shown in Fig. 3.7. To perform the Offline Encrypted PIN CVM the terminal firstly issues the `GET CHALLENGE` command to the card, and then uses the public key of the card to encrypt the received card's nonce n_C and the PIN. The verification result is then received unauthenticated.

While this CVM mode indeed hides the PIN from the eavesdropper, it also makes the encryption of the PIN local to the current session, thus avoiding the replay attack and making it impossible to use the encrypted PIN to link sessions. However, the response from the card can still be blocked and replaced with, e.g. the `SUCCESS` even if the entered PIN was wrong.

Online encrypted PIN. Finally, if the terminal can perform online transactions instead of the card, the (encrypted) PIN is transmitted to the issuing bank for verification. In this case, the CVM is part of the Transaction Authorisation phase.

3.1.4 Transaction Authorisation

Transaction Authorisation (TA) is the ultimate and mandatory phase at which the terminal asks the card to generate the Application Cryptogram AC. The cryptogram is then sent to the issuing bank to either claim the funds of the cardholder, request authorisation or log the failed attempt. Hence there are three respective types of cryptogram that the terminal may ask the card to generate based on the internal policies of the terminal, e.g. the ceiling limit for offline transactions: a success cryptogram TC (Transaction Cryptogram), an authorisation request $ARQC$ (Authorisation Request Cryptogram) for online transaction, and the failure cryptogram AAC (Application Authentication Cryptogram).

A cryptogram is typically an HMAC generated over the data coming both from the card and the terminal (specified by CDOL objects the terminal has received in the Initialisation phase). The key for this HMAC is derived from the shared secret mk between the card and the issuing bank called the master key, and the Application Transaction Counter (ATC), a counter on the card that increments each time the `GET PROCESSING OPTIONS` command is issued[3]. The main intention behind the use of monotonically increasing counter ATC is to prevent the re-use of old cryptograms. The minimum set of data elements to be included in the cryptogram recommended by the EMVCo [emv11, Book 2, Section 8] contains, for instance, ATC, AIP, terminal country code, transaction date, amount, etc. Together with the cryptogram, the data that it was generated over must be provided to verify the cryptogram. As this data is not encrypted and may contain identifying information about the card (e.g. the counter ATC), linkability issues may arise. In what follows, when we talk about cryptograms we consider both its elements: the data and the HMAC it is generated over.

Before explaining both offline and online modes, we clarify how the CDA mode for offline data authentication works, as it is part of the Transaction Authorisation

[3]Notice, that an attacker can use a device capable of communicating with the card to issue enough `GET PROCESSING OPTIONS` commands to max-out the two-byte ATC.

step. If CDA is selected, the SDAD (similar to as in DDA explained in Sec. 3.1.2) is sent to the terminal instead of the cryptogram, but the SDAD should be generated over the cryptogram, the terminal's nonce n_C, the card's nonce n_T, the type of the cryptogram CID (Cryptogram Information Data), and the hash of the transaction details. The SDAD is only generated when the cryptogram is $TC/ARQC$, and is never generated if the requester cryptogram is AAC. In what follows, we assume that CDA is not selected for offline authentication of the card's data (or ODA is skipped).

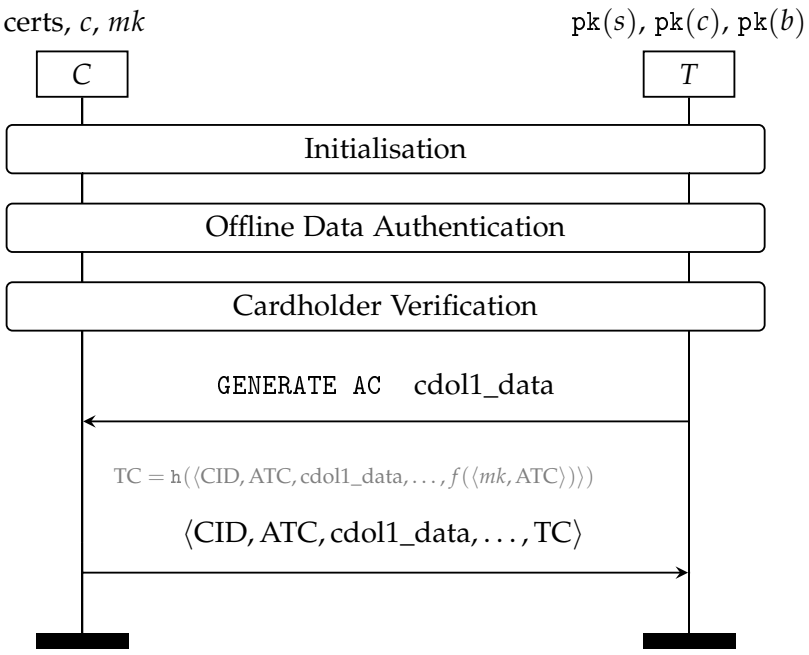


Figure 3.8: Transaction authorisation in the offline mode.

Offline authorisation. We present the offline TA in Fig. 3.8. The terminal asks the card to generate the TC by issuing the `GENERATE AC` command together with the data indicated in CDOL1 and the payload indicating the type of the requested cryptogram and a parameter indicating whether the CDA is requested. The card responds with the Cryptogram Information Data (CID) indicating the type of the cryptogram, the counter ATC, the cryptogram itself, and, optionally, proprietary Issuer Application Data (IAD) to be sent to the issuing bank. Though the terminal does not check the TC, and sends it to the issuing bank later, an offline transaction at this stage is complete, and the goods are released to the cardholder.

Online authorisation. Online TA proceeds as in Fig. 3.9. In an online transaction, two cryptograms are generated. Firstly the terminal requests the card to generate $ARQC$ cryptogram by issuing `GENERATE AC` and providing CDOL1 data, the cryptogram type and the CDA flag similarly to the offline mode. Then the terminal immediately requests the issuing bank to authorise the current transaction by forwarding the $ARQC$ to the bank, together with the transaction details and the (encrypted) PIN in case online PIN CVM was selected. When the confirmation

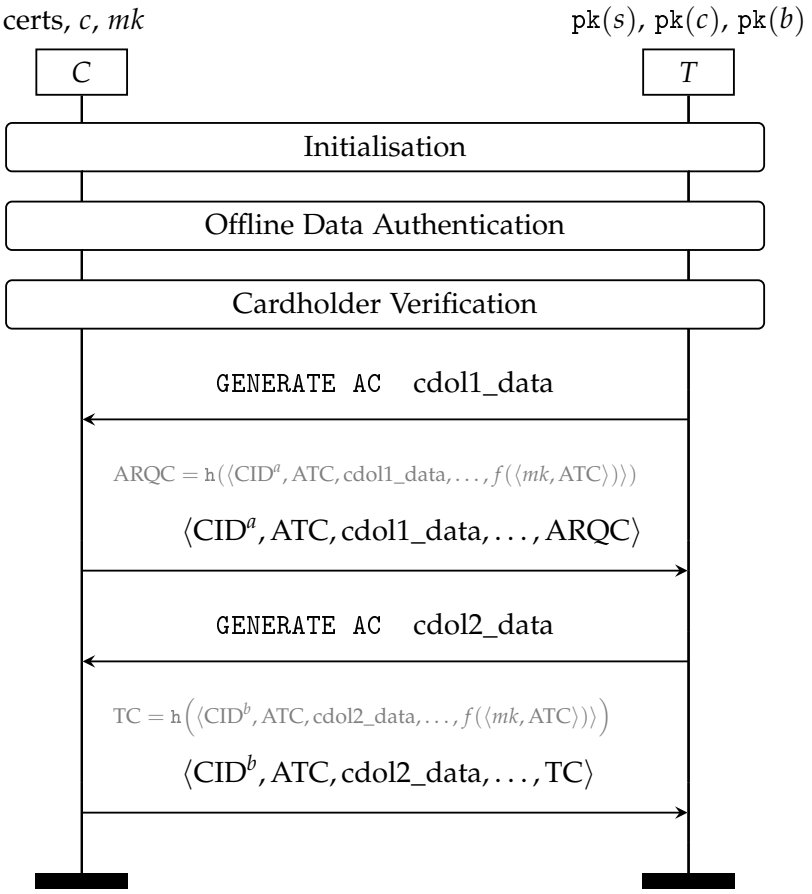


Figure 3.9: Transaction authorisation in the online mode.

from the bank and further data[4] has been received, the terminal issues the second `GENERATE AC` command to the card asking the card to generate the *TC* and providing CDOL2 data which includes the bank's response. When the *TC* is received by the terminal, the transaction is completed, and the goods are released to the cardholder.

Contactless transactions. In the above, we have described contact transactions. The main difference of the contactless case is that the application cryptogram is generated right away when the transaction data is received and is sent to the terminal in response to the `GPO` command (see Sec. 3.1.1). Moreover, neither user input nor reply from the bank can be received by the card, as the reader should keep the card within the reader's field throughout the whole session.

As the reader may observe by now, the EMV standard is flexible. A transaction contains a variety of optional messages that the card and the terminal may or may not include in the communication. Moreover, there are optional phases (ODA, CV) perfectly allowed to skip if a particular payment system decides to implement the

[4]The data received from the bank may also contain instructions for the card to update certain data inside it. Despite being part of the online TA, in EMV, this script processing step is often regarded as a separate optional phase of the transaction.

"reduced" version of the standard. Formulations such as "the recommended minimum set of data elements to be included in . . .", " it is strongly recommended . . .", etc., common for the EMV books suggest that deviating from recommendations is allowed since there are also explicit mandatory requirements. Insecure configurations to this date are part of the EMV standard, as has been recently demonstrated by Basin, Sasse, and Toro-Pozo in "The EMV Standard: Break, Fix, Verify" [BST21b] where in particular they analysed 24 EMV contact configurations among which only three has been found to guarantee secure transactions. One such configuration has been independently discovered by the author of this dissertation by studying the EMV standard with a naked eye and is presented in the next section.

3.2 An insecure EMV configuration

In this section we focus on a few observations made above when explaining the main stages of the EMV transaction, and combine them to come up with an EMV configuration that does not contradict the standard, however, is vulnerable to a PIN bypassing attack. This attack belongs to the same family as the long-known vulnerability disclosed by Murdoch et al. [MDAB10] and root in the fact that certain message(s) coming from the card are not authenticated. We are clear that this attack is theoretical and can be executed only for a very particular configuration. We can neither guarantee the existence of cards which carry this configuration (hence vulnerable to the attack) nor the opposite. The point we are stressing in this section is that insecure configurations are still part of the EMV standard, while they should not be allowed in the first place.

The first observation regards the Initialisation and the ODA phases. Notice that the CVM_list is not required to be included in SSAD, the data to be authenticated by the terminal, hence, if not, it can be altered by the man-in-the-middle attacker. The second observation regards the CV phase – the response to PIN verification from the card is never authenticated, hence, again, it can be altered by the man-in-the-middle attacker. Finally, the third observation is also regards the CV phase – the PIN is sent to the card in case of offline PIN verification, while it is sent to the bank in case of online PIN verification. Hence, the strategy of an attacker is to introduce the disagreement between the online and offline modes as presented in Fig. 3.10.

Figure 3.10: Overview of the strategy of an attacker.

To develop this strategy into a full attack, we assume particularly configured card and terminal. The CVM_list on the card should not be included in the SSAD and should comprise a unique CVM, online PIN. The terminal should be offline and support offline PIN verification. We then present a full attack in Fig. 3.11, which proceeds as follows. At the Initialisation, the man-in-the-middle attacker replaces the online PIN entry in the CVM_list with any version of the offline PIN verification,

e.g. offline encrypted PIN. The terminal then executes the chosen offline CVM while the man-in-the-middle blocks any messages regarding the PIN verification from reaching the card and ultimately responds with the SUCCESS message. Then the terminal completes the offline transaction asking the card to generate the *TC* allowing an attacker to successfully use the card without knowing the PIN while the cardholder would be liable for this fraudulent transaction as the PIN-based CVM has been used.

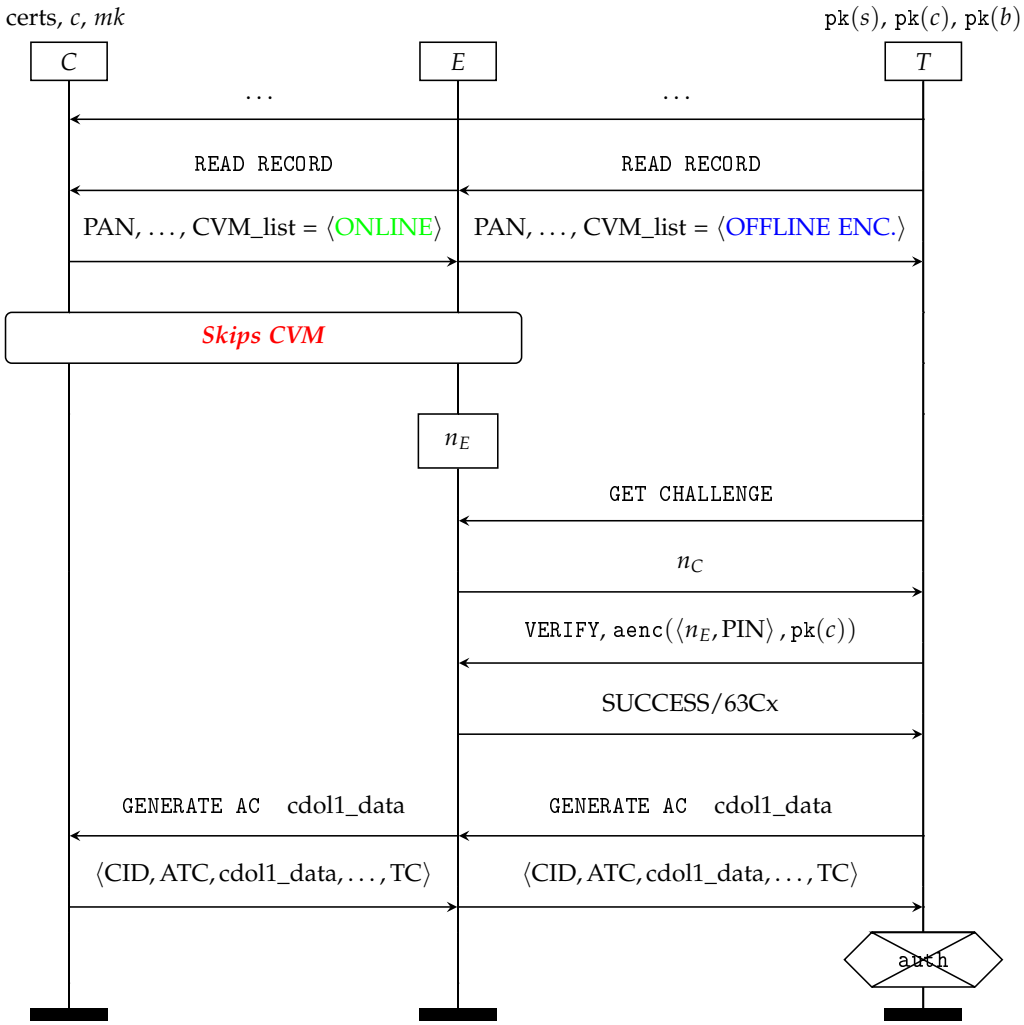


Figure 3.11: Attack bypassing the PIN in an offline transaction.

Finally, we make an observation that the card never authenticates the terminal and its choice of the CVM. Hence, in case `CVM_list` $= [ONLINE, OFFLINE]$ is authenticated, i.e. is included if the SSAD, the described attack is still possible.

This section completes the description of the current EMV standard and highlights the security and privacy issues it has. Ill-configured versions of the protocol still exist in the standard without the indication that they should never be implemented. Sensitive data allowing either to identify or profile cards is exposed to eavesdroppers. Relatively recently, the EMVCo proposed to enhance the privacy

of transactions, and mitigate eavesdroppers by encrypting the communication between the card and the terminal [rfc12]. In the next sections we will analyse this enhancement using the machinery developed in Chapter 2.

3.3 Enhancing the privacy of EMV transactions: Blinded Diffie-Hellman key establishment proposal

As we have just seen, currently, the EMV standard trivially does not satisfy privacy properties such as anonymity and unlinkability due to transferring the card number in cleartext during a transaction. Hence transaction data allows us to link transactions made with the same card and effortlessly track cardholders. The fact that no actual payments need to be made eases the task of the adversary when tracking a contactless card, as it is ready to present its identity to any device.

In 2011 EMVCo launched the development of a new version of the standard, the EMV 2nd Gen, where the card should be protected against eavesdropping. To facilitate this, EMVCo proposed the use of *secret channels*. A secret channel is a symmetric key that the card and the terminal establish at the start of each session and use to encrypt further communications. A channel establishment procedure is based on Diffie-Hellman key agreement with a twist: the card uses a freshly blinded static certified public key instead of an ephemeral public key. Hence, the name of the proposed protocol, *Blinded Diffie-Hellman* (BDH) [rfc12]. In this section we analyse this proposal in detail.

The BDH protocol is meant to satisfy the official requirements for channel establishment from the architecture overview of the EMV 2nd Gen [ove14]:

- Use elliptic-curve based cryptography (ECC).
- Computational resources of the card are respected.
- An attacker who passively eavesdrops on communications cannot identify a particular card.

Several authors published a security proof for the Blinded Diffie-Hellman protocol [BSWW13, GZZH14] and established that a passive eavesdropper, that only listens to transmitted messages, cannot reidentify a card, therefore BDH satisfies the above requirements. Brzuska, Smart, Warinschi, and Watson [BSWW13] named this property of BDH "external unlinkability".

It is natural to discuss a potential strengthening of the requirements for BDH listed above, i.e. to lift the limitation of attackers being passive. In the context of contactless payments the requirement that any attacker cannot identify a particular card is realistic, since it is easy for an attacker to initiate sessions with contactless cards using devices, such as smartphones, that need not be official terminals. In fact, different capabilities must be considered in a wireless environment depending on their distance from the card as we have briefly mentioned in the introduction to this chapter. It is difficult to perform an eavesdropping attack outside of the approximately 20m radius [PFB12]. However, successful attacks executed by *a passive attacker* are reported within the range between 20m and 100cm [NGFR08, EPFB13]

and, with the right equipment, within 100cm *an active attacker* can power up the card and start executing the protocol [HDPdR15]. A close active attacker is a real threat to the privacy of anyone having a card in their pocket, since the distance of 100cm is easily achievable, e.g. at doorways or checkouts.

Unsurprisingly, the proposed BDH protocol is no longer secure in such strictly stronger threat model. To show that we firstly introduce the Blinded Diffie-Hellman key establishment protocol from the original EMVCo request for comments [rfc12], secondly, we define our target privacy notion, unlinkability, that accounts for active attackers, and thirdly, we demonstrate an attack invalidating our strong unlinkability goal on Blinded Diffie-Hellman, in the form initially proposed by EMVCo described by a modal logic formula.

3.3.1 Blinded Diffie-Hellman and external unlinkability

To present the BDH protocol we define the syntax of messages in Fig. 3.12. The syntax for messages includes abstractions for the arithmetic operations on elliptic curves enabling us to represent protocols symbolically. We leave the cryptographic details for multiplication, scalar multiplication and public key operations together with ECC domain parameters as a footnote[5]. The exact signing mechanism modelled by $\mathtt{sig}(M, N)$ is not specified by EMVCo in the proposal [rfc12]. Hash, pair and symmetric encryption are standard and `auth` is a message that upon being output indicates that the terminal believes it has authenticated the card.

$M, N ::=$	$\mathtt{g}$	DH group generator (constant)
$\mid$	x	variable
$\mid$	$M \cdot N$	multiplication
$\mid$	$\phi(M, N)$	scalar multiplication
$\mid$	$\mathtt{pk}(M)$	public key
$\mid$	$\mathtt{sig}(M, N)$	signature
$\mid$	$\mathtt{h}(M)$	hash (for key derivation)
$\mid$	$\langle M, N \rangle$	pair
$\mid$	$\{M\}_N$	symmetric encryption
$\mid$	$\mathtt{check}(M, N)$	check signature
$\mid$	$\mathtt{fst}(M)$	get first
$\mid$	$\mathtt{snd}(M)$	get second
$\mid$	$\mathtt{dec}(M, N)$	symmetric decryption
$\mid$	`auth`	authenticate

Figure 3.12: Blinded Diffie-Hellman syntax.

The equational theory E_0 axiomatising the properties of the cryptographic functions is given in Fig. 3.13 The first three equations capture the interaction between field arithmetic and scalar multiplication followed by standard destructors:

[5]The public parameters are as follows: a finite field $\mathbb{F}_p$; a Diffie-Hellman group G, defined over an elliptic curve $E(\mathbb{F}_p)$; the (prime) order q of G; the generator $\mathtt{g} \in G$; the key-derivation function h; the public key of the payment system $\mathtt{pk}(s)$ for the certificate verification. We employ (left) group action notation $\phi\colon \mathbb{F}_q^\times \times G \to G$ for group operation: we write $\phi(r, Q)$ for the element Q added with itself r times and call ϕ *scalar multiplication*. The symbol $\cdot$ denotes multiplication between two scalars (field elements). All freshly generated values are picked uniformly at random from $\mathbb{F}_q$. The secret key k is an element of $\mathbb{F}_q$ and the corresponding public key is of the form $\phi(k, \mathtt{g})$. Blinding of the element Q uses a fresh scalar a and internally works as a scalar multiplication: $\phi(a, Q)$.

projections, decryption, and signature check. We model signature verification in a manner that is standard when symbolically verifying protocols: the signature is verified iff the message is successfully extracted by applying $\mathtt{check}$ from $\mathtt{sig}(K, M)$ using the corresponding public key $\mathtt{pk}(K)$ as we have already described informally in the introduction to Sec. 3.1.

$$
\begin{aligned}
&M \cdot N =_{E_0} N \cdot M \\
&(M \cdot N) \cdot K =_{E_0} M \cdot (N \cdot K) \\
&\phi(M \cdot N, K) =_{E_0} \phi(M, \phi(N, K)) \\
&\mathtt{fst}(\langle M, N \rangle) =_{E_0} M \\
&\mathtt{snd}(\langle M, N \rangle) =_{E_0} N \\
&\mathtt{dec}(\{M\}_K, K) =_{E_0} M \\
&\mathtt{check}(\mathtt{sig}(M, K), \mathtt{pk}(K)) =_{E_0} M
\end{aligned}
$$

Figure 3.13: Equational theory E_0 for the Blinded Diffie-Hellman protocol.

The Blinded Diffie-Hellman protocol is presented in Fig. 3.14. There are two agents in the system that participate in the execution of the protocol: the card C and the terminal T. The payment system holds a secret key s and acts as a certification authority. The private key c, the public key $\phi(c, \mathfrak{g})$ and the certificate $\langle \phi(c, \mathfrak{g}), \mathtt{sig}(\phi(c, \mathfrak{g}), s) \rangle$ are permanently embedded in the card when it is manufactured. The card can only be issued by the bank in cooperation with payment systems like Amex, Visa, etc. The terminal, in contrast to the card, can be manufactured by anyone. To verify the legitimacy of the card, the terminal uses a public key of the payment system $\mathtt{pk}(s)$ that is available on the system's website. Notice that in our model of BDH we have reduced the chain of certificates described in Sec. 3.1.1 to just one certificate. This simplification is in line with the already published computational analysis of BDH [BSWW13, GZZH14].

The card starts the communication by sending its public key $\phi(c, \mathfrak{g})$ blinded with a fresh scalar a to the terminal. In response, the terminal sends ephemeral public key $\phi(t, \mathfrak{g})$ to the card. This is enough to establish a common secret key $k_c = k_t$. The card uses this key to encrypt the authentication data: blinding scalar a, static public key $\phi(c, \mathfrak{g})$, and the certificate $\langle \phi(c, \mathfrak{g}), \mathtt{sig}(\phi(c, \mathfrak{g}), s) \rangle\rangle$. Finally, the terminal verifies the received certificate by checking the signature against the public key of the payment system $\mathtt{pk}(s)$, checks that $\phi(c, \mathfrak{g})$ blinded with a coincides with the first message z_1 received from the card. Upon success, the terminal authenticates the card and is ready to continue with the transaction on the encrypted channel.

3.3.2 Blinded Diffie-Hellman and active attackers

In order to verify that blinding the card's public key protects against eavesdroppers external to the execution, the property of *external unlinkability* was introduced in [BSWW13]. In an externally unlinkable payment system, an attacker observing a message exchange between a card and a terminal cannot link that card's current session with a previous session from the same card.

In the real world, anyone could build a device imitating the terminal, for instance, an app on a smartphone supporting NFC or a skimming gate [HDPdR15].

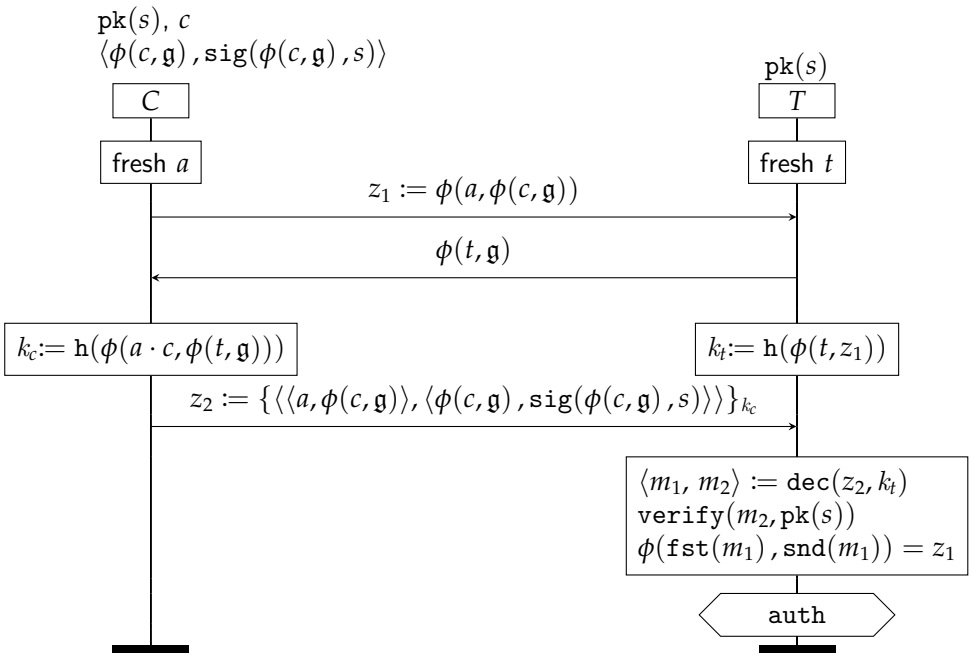


Figure 3.14: EMV 2nd Gen key establishment.

Such a device need not be certified or connected to any bank. Taking this into account, there is a straightforward attack on the BDH protocol (Fig. 3.14) in the presence of malicious terminals:

1. A malicious terminal establishes a key with an honest card, then successfully decrypts the message z_2 and obtains the card's public key $\phi(c, \mathfrak{g})$.

2. Another terminal operated by the attacker runs a new session with the same card to obtain again the card's public key $\phi(c, \mathfrak{g})$; and hence recognises the card.

This attack however would not be considered to be an attack on external unlinkability, due to the fact that the attacker actively starts communicating with the card. Since it is easy to activate a contactless card, e.g. while the card is in the wallet, external unlinkability is too weak. This compels us to adopt a stronger notion of unlinkability which can be used to discover the above attack formally.

The above attack suggests that any network of malicious powerful terminal-like devices unrelated to any payment system may track selected contactless cards in real-time without the cardholder being aware simply by starting sessions with the card in the cardholder's pocket. Thus we propose to view unlinkability as a *property of the card* in a hostile environment that should hold with or without the presence of honest terminals supporting the compositionality argument we have made in Sec. 2.2.4. The attack also highlights why the BDH protocol is not unlinkable in the presence of active attackers – the ability of the terminal to obtain the card's public key which serves as the card's identity.

3.3.3 Blinded Diffie-Hellman is not unlinkable

To show formally that the BDH protocols fails to deliver unlinkability we firstly give the formal specification (that uses the equational theory E_0 from Fig. 3.13) for the roles in the BDH protocol presented in Fig. 3.14. Then we provide the unlinkability definition that accounts for active attackers. In the specification we also include the so-called events, starting with `ev:`, needed later for the verification of security properties. Until Sec. 3.5.3, where we clarify their meaning, events can be ignored.

$$
\begin{aligned}
&C_{\text{rfc}}(s,c,ch) \triangleq \nu a.\overline{ch}\langle\phi(a,\phi(c,\mathfrak{g}))\rangle.\\
&\quad ch(y).\\
&\quad \texttt{let}\ k_c := \mathtt{h}(\phi(a\cdot c,y))\ \texttt{in}\\
&\quad \texttt{let}\ cert := \langle\phi(c,\mathfrak{g}),\mathtt{sig}(\phi(c,\mathfrak{g}),s)\rangle\ \texttt{in}\\
&\quad \mathtt{ev{:}CRunning}\,(\phi(a,\phi(c,\mathfrak{g})),y,\{\langle\langle a,\phi(c,\mathfrak{g})\rangle,cert\rangle\}_{k_c})\\
&\quad \overline{ch}\langle\{\langle\langle a,\phi(c,\mathfrak{g})\rangle,cert\rangle\}_{k_c}\rangle
\end{aligned}
$$

$$
\begin{aligned}
&T_{\text{rfc}}(pk_s,ch) \triangleq \nu t.ch(z_1).\\
&\quad \overline{ch}\langle\phi(t,\mathfrak{g})\rangle.\\
&\quad ch(z_2).\\
&\quad \texttt{let}\ k_t := \mathtt{h}(\phi(t,z_1))\ \texttt{in}\\
&\quad \texttt{let}\ \langle m_1,m_2\rangle :=\\
&\quad \langle\mathtt{fst}(\mathtt{dec}(z_2,k_t)),\mathtt{snd}(\mathtt{dec}(z_2,k_t))\rangle\ \texttt{in}\\
&\quad \texttt{if}\ \mathtt{snd}(m_1) = \mathtt{check}(\mathtt{snd}(m_2),pk_s)\ \texttt{then}\\
&\quad \texttt{if}\ \phi(\mathtt{fst}(m_1),\mathtt{snd}(m_1)) = z_1\ \texttt{then}\\
&\quad \mathtt{ev{:}TCommit}\,(z_1,\phi(t,\mathfrak{g}),z_2)\\
&\quad \overline{ch}\langle\mathtt{auth}\rangle
\end{aligned}
$$

The card role process is parametrised by the secret key s of the payment system, the secret key c of the card and the session channel ch. The terminal role is parametrised only by the system's public key pk_s and ch. The action $\overline{ch}\langle\mathtt{auth}\rangle$ is an event used to indicate at what point the terminal believes it has authenticated the card.

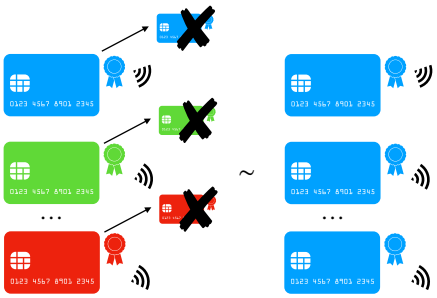
To define unlinkability for our key agreement we follow the pattern in Fig. 2.1, and consider the idealised situation. Obviously, if cards immediately expire after one use, it is impossible to link two sessions. Hence, if the real world system (implementation), where cards are used multiple times, is indistinguishable by an attacker from an idealised unlinkable world system (specification), in which cards are disposed of after each use, then unlinkability of is achieved.

Let $C(s,c,ch)$ be the card process scheme parametrised by the payment system's secret key s, communication channel ch and the card's secret key c. Then we have the following.

Definition 9. (unlinkability) *A card process scheme C is unlinkable whenever*

$$\nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.\nu ch.\overline{card}\langle ch\rangle.C(s,c,ch)$$
$$\sim$$
$$\nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.!\nu ch.\overline{card}\langle ch\rangle.C(s,c,ch)$$

The process on the left of the above relation models the idealised world where a card participates in no more than one transaction. This process starts by creating the secret key of the payment system s. Then the public key $\mathtt{pk}(s)$ of the payment system is made available via the output on the public channel out. Each newly manufactured card c is allowed to participate in the execution of the payment protocol just once. The process on the right of the above relation models the more realistic situation where each card c may participate in several runs of the protocol. If the idealised situation is equivalent to the real world one, where the equivalence we employ is quasi-open bisimilarity (Def. 5), we say that the payment system satisfies unlinkability. Notice that this definition is simply the revised, according to the remark in the end of Sec. 2.2.4, version of the AU-unlinkability (Def. 7). The Theorem 4, indeed allows us to check unlinkability for a subsystem comprising cards only and be sure that the presence of any terminals would not make the whole system linkable.



Now, given the formal definition of unlinkability in Def. 9 we can establish that the Blinded Diffie-Hellman protocol from Fig. 3.14 is not unlinkable.

Theorem 6. $C_{\mathrm{rfc}}(s,c,ch)$ *violates unlinkability.*

Proof. To describe the attack on unlinkability of the BDH protocol we present the proof certificate in the form of modal logic formula, as Sec. 2.2.3. Consider the following processes, where C_{rfc} is defined in the beginning of this section (Sec. 3.3.3).

$$RFC_{\mathrm{spec}} \triangleq \nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.\nu ch.\overline{card}\langle ch\rangle.C_{\mathrm{rfc}}$$
$$RFC_{\mathrm{impl}} \triangleq \nu s.\overline{out}\langle \mathtt{pk}(s)\rangle.!\nu c.!\nu ch.\overline{card}\langle ch\rangle.C_{\mathrm{rfc}}$$

To show that $RFC_{\mathrm{spec}} \not\sim RFC_{\mathrm{impl}}$ we present a formula that is satisfied by RFC_{impl}, but not by RFC_{spec}. Let the formula ψ be as follows.

$$\langle \overline{out}(pk_s)\rangle$$
$$\langle \overline{card}(u_1)\rangle\langle \overline{u_1}(v_1)\rangle\langle u_1\, \phi(y_1,\mathfrak{g})\,\rangle\langle \overline{u_1}(w_1)\rangle$$
$$\langle \overline{card}(u_2)\rangle\langle \overline{u_2}(v_2)\rangle\langle u_2\, \phi(y_2,\mathfrak{g})\,\rangle\langle \overline{u_2}(w_2)\rangle$$
$$\big(\mathtt{snd}(\mathtt{dec}(w_1,\mathtt{h}(\phi(y_1,v_1)))) = \mathtt{snd}(\mathtt{dec}(w_2,\mathtt{h}(\phi(y_2,v_2))))\,\big)$$

The above formula exactly reflects the attack described informally in Sec. 3.3.2 and represents two sessions of the BDH protocol, which, for RFC_{impl}, can be with the same card, say c_1. The equality test at the end of ψ compares the certificates obtained from each session to each other, which the terminal can decrypt in both sessions. This certificate can be the same for both sessions of RFC_{impl} involving the same card, since it is bound to the card's identity c_1. Therefore $RFC_{\text{impl}} \models \psi$. Indeed, at the point of the equality test the accumulated substitution comprising the messages available on the network is as follows.

$$\left\{ \mathtt{pk}(s)/_{pk_s} \right\} \circ \left\{ ch_1, \phi(a_1, \phi(c_1, \mathfrak{g})), n^1(a_1, \phi(y_1, \mathfrak{g})) /_{u_1, v_1, w_1} \right\} \circ \left\{ ch_2, \phi(a_2, \phi(c_2, \mathfrak{g})), n^2(a_2, \phi(y_2, \mathfrak{g})) /_{u_2, v_2, w_2} \right\}$$

Where $n^d(a, x) := \{\langle \langle a, \phi(c_d, \mathfrak{g}) \rangle, \langle \phi(c_d, \mathfrak{g}), \mathtt{sig}(\phi(c_d, \mathfrak{g}), s) \rangle \rangle\}_{\mathtt{h}(\phi(a \cdot c_d, x))}$. Applying the substitution above to the equality from the formula we verify that the two sides are equal given the equational theory E_0. In contrast, $RFC_{\text{spec}} \not\models \psi$ since every session is with a new card and hence the equality test never holds, since the certificates will always differ. □

Since we have just demonstrated that active attackers playing the role of honest terminals can track cardholders by the certificate the card gives away in each session, it is natural to consider an improvement to BDH that protects from such attackers. In the next section we presents an updated version of BDH that satisfies the unlinkability Def. 9. However, before doing that we would like to take a break and reflect on different privacy notions. The readers who are not interested in this discussion can safely skip and continue directly to Sec. 3.5.

3.4 On different privacy notions

As mentioned in the introduction to the previous chapter, our main reference for privacy is the ISO/IEC standard 15408 [cc17], which includes informal definitions for anonymity and unlinkability. Arapinis et al. [ACRR10] formalised unlinkability and anonymity making the distinction between weak and strong variants of each (not to be confused with weak/strong bisimilarity, see the remark in Sec. 2.2.4). his section is an informal comment to their paper.

Strong unlinkability vs weak unlinkability. Since our running example is unlinkability, called "strong unlinkability" in [ACRR10], we would like to draw a clear distinction with the weak variant. The weak unlinkability, formulated as a trace property in [ACRR10], assumes that an attacker can "link two particular messages as being part of different sessions executed by the same principal", while the unlinkability we are discussing in this thesis requires that a world where an agent can participate in the protocol multiple times is no different from the idealised world where an agent can participate in the protocol more than once as in Fig. 2.1, Def. 7, Def. 9 While weak unlinkability follows from strong unlinkability as Arapinis et al. prove, the converse does not hold demonstrated by the following example, where the process on the right outputs the `OK` message only if it receives the identity of the left process twice.

$$WU_{impl} \triangleq \nu a.\,(!\nu id.!\overline{a}\langle id\rangle |\; !a(x).a(y).\texttt{if } x = y \texttt{ then } \overline{c}\langle \texttt{OK}\rangle)$$

The protocol above satisfies weak unlinkability – since the input for the right process is on the private channel a, there are no message belonging to the same id for an attacker to link, the input is executed silently using τ-transition. Strong unlinkability, however, is violated. To see that, let us define the respective idealised world, where the left process cannot execute the protocol more then once by removing the second replication.

$$WU_{spec} \triangleq \nu a.\,(!\nu id.\overline{a}\langle id\rangle |\; !a(x).a(y).\texttt{if } x = y \texttt{ then } \overline{c}\langle \texttt{OK}\rangle)$$

Since in WU_{spec} the process on the right can never receive the same identity twice, it never outputs any message, and we have the formula $\langle\tau\rangle\langle\tau\rangle\langle\overline{c}u\rangle(u = \texttt{OK})$, satisfied by WU_{impl}, but not WU_{spec} demonstrating $WU_{impl} \not\approx WU_{spec}$.

Anonymity. We would like to discuss anonymity, only informally. As written in the ISO/IEC standard 15408 [cc17], "anonymity ensures that a subject may use a resource or service without disclosing its user identity". Arapinis et al. interpret this formulation such that in a weakly anonymous protocol an attacker cannot tell when a transition is initiated by one agent or another; and in a strongly anonymous protocol, a real world where an agent simply executes its role is no different from the idealised world, where the same role can be executed by an agent with another identity, never seen before. Similarly to unlinkability, there is a counter-example demonstrating a protocol satisfying weak, but not strong anonymity.

Unlinkability $\not\Longleftrightarrow$ anonymity? We repeat the point made in [ACRR10], that neither unlinkability guarantee anonymity nor vice-versa. Consider a protocol described by the following process.

$$\nu k.!\nu id.!\overline{c}\langle\{id\}_k\rangle$$

An attacker here can identify two messages coming from the same identity, yet they cannot infer any identity itself, hence unlinkability is not ensured by anonymity. At the same time the following process represents an unlinkable, yet not anonymous protocol.

$$U_{impl} \triangleq \nu a, b\left(!\nu id.\overline{a}\langle \texttt{id}\rangle.!b(x).\texttt{if } x = \texttt{id} \texttt{ then } \overline{c}\langle id\rangle |\; !a(x).\overline{b}\langle x\rangle\right)$$

The process on the left can output the identity at at most once, hence anonymity is violated, while the unlinkability is preserved exactly because no identity is revealed twice, and an attacker observes only a stream of distinct identities. To demonstrate this formally, we define the idealised protocol and a bisimulation $\mathcal{R}$ as the least symmetric open relation satisfying conditions in Fig. 3.15.

$$U_{spec} \triangleq \nu a, b\left(!\nu id.\overline{a}\langle \texttt{id}\rangle.b(x).\texttt{if } x = \texttt{id} \texttt{ then } \overline{c}\langle id\rangle |\; !a(x).\overline{b}\langle x\rangle\right)$$

$$U_{spec} \;\mathcal{R}\; U_{impl}$$

$$\nu a, b, id_1 \ldots id_l.\Big(\sigma \mid P_1 \mid \ldots P_l \mid !T_{spec} \mid !R\Big) \;\mathcal{R}\; \nu a, b, id_1 \ldots id_l.\Big(\theta \mid Q_1 \mid \ldots Q_l \mid !T_{impl} \mid !R\Big)$$

$$T_{spec} \triangleq \nu id.\overline{a}\langle \mathtt{id}\rangle.b(x).\mathtt{if}\ x = \mathtt{id}\ \mathtt{then}\ \overline{c}\langle id\rangle$$

$$T_{impl} \triangleq \nu id.\overline{a}\langle \mathtt{id}\rangle.!b(x).\mathtt{if}\ x = \mathtt{id}\ \mathtt{then}\ \overline{c}\langle id\rangle$$

$$R \triangleq a(x).\overline{b}\langle x\rangle$$

for any A, B, Γ, Δ partitioning $L \coloneqq \{1, \ldots l\}$ and a derangement f of Δ such that the following hold

$$u_i\sigma = id_i \quad \text{if } i \in \Gamma \qquad u_i\theta = id_i \quad \text{if } i \in \Gamma$$

$$P_i \triangleq \begin{cases} b(x).\mathtt{if}\ x = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid \overline{b}\langle id_i\rangle & \text{if } i \in A \\ \mathtt{if}\ \mathtt{id}_i = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid 0 & \text{if } i \in B \\ 0 \mid 0 & \text{if } i \in \Gamma \\ \mathtt{if}\ \mathtt{id}_{f(i)} = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid 0 & \text{if } i \in \Delta \end{cases}$$

$$Q_i \triangleq \begin{cases} !b(x).\mathtt{if}\ x = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid \overline{b}\langle id_i\rangle & \text{if } i \in A \\ \mathtt{if}\ \mathtt{id}_i = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid !b(x).\mathtt{if}\ x = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid 0 & \text{if } i \in B \\ 0 \mid !b(x).\mathtt{if}\ x = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid 0 & \text{if } i \in \Gamma \\ \mathtt{if}\ \mathtt{id}_{f(i)} = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid !b(x).\mathtt{if}\ x = \mathtt{id}_i\ \mathtt{then}\ \overline{c}\langle id_i\rangle \mid 0 & \text{if } i \in \Delta \end{cases}$$

$c \# a, b, id_1, \ldots id_l$ and u_i, $i \in \Gamma$ are distinct variables s.t. $u_i \# c, a, b, id_1, \ldots id_l$

Figure 3.15: Relation $\mathcal{R}$ verifying $U_{spec} \sim U_{impl}$

Or... Unlinkability $\implies$ anonymity? The first example in the previous paragraph works due to the notion of identity. In the protocol described by $\nu k.!\nu id.!\overline{c}\langle\{id\}_k\rangle$ the identity id is formally always hidden, while intuitively it is clear that the term $\{id\}_k$ plays the role of the user's identity as it uniquely identifies the user.

With an enforced notion of identity[6]as a term that an attacker uses to identify the user unambiguously, we can informally declare the property of per-session anonymity. Per-session anonymity holds if an attacker cannot construct a message term allowing to re-identify the user from the messages contributing a protocol run.

For the examples let us extend temporarily the message theory with the associative and commutative XOR operator $\oplus$ which have a property $M \oplus N \oplus M \oplus K =_E N \oplus K$. Then consider the protocol described by the following process.

$$\nu k.!\nu id.!\nu a.\overline{c}\langle\phi(a, id)\rangle\overline{c}\langle\phi(a, \mathtt{sig}(id, k))\rangle$$

It is per-session anonymous since both forms of identity: id and $\mathtt{sig}(id, k)$ are hidden from an attacker and, as we hypothesise, it is impossible to construct from the messages of the form $\phi(a, id)$ and $\phi(a, \mathtt{sig}(id, k))$ a term that could allow to identify user in the next session. At the same time the following protocol where we use $\oplus$

[6]A term "pseudonym" can also be found in the literature, while ISO/IEC standard 15408 [cc17] makes no distinction between terms "identity" and "pseudonym" – identity is defined as a representation "uniquely identifying entities ... For a human user, the representation can be the full or abbreviated name or a (still unique) pseudonym".

instead of ϕ, is not per-session anonymous.

$$\nu k.!\nu id.!\nu a.\overline{c}\langle a \oplus id\rangle \overline{c}\langle a \oplus \mathtt{sig}(id,k)\rangle$$

Indeed, since the term $a \oplus id \oplus a \oplus \mathtt{sig}(id,k) =_E id \oplus \mathtt{sig}(id,k)$ is constant throughout all sessions and is tied to the identity of the user, an attacker can identify sessions with that user.

With the notion of per-session anonymity we conclude this section with the following informal hierarchy of privacy properties.

$$\begin{array}{ccccc} \text{Unlinkability} & \Longrightarrow & \text{Weak unlinkability} & \Longrightarrow & \text{Per-session anonymity} \\ \Updownarrow & & & & \\ \text{Anonymity} & \Longrightarrow & \text{Weak Anonymity} & & \end{array}$$

3.5 An unlinkable key agreement for EMV payments

In this section, we propose our improvement to the BDH protocol proposed by EMVCo called *Unlinkable BDH* (UBDH). This improvement makes use of a certification scheme with certificates invariant under blinding. We point to an existing instance of such a certification scheme, the Verheul certification scheme, and, finally, we prove that our improvement indeed makes the Blinded Diffie-Hellman protocol unlinkable [rfc12].

3.5.1 Unlinkable Blinded Diffie-Hellman UBDH

Recall from Section 3.3.2 and Theorem 6 that the reason behind the failure of unlinkability of the BDH protocol proposed by EMVCo is that the card gives away its static certificate and its blinding factor. While this allows an honest terminal to authenticate the card, the public key of the card ultimately obtained by the terminal may be used to track the card in the future. We demonstrate in this section that authentication can still be performed without disclosing the public key or the signature. In order to achieve this, we specify more precisely the signature scheme (initially unspecified by EMVCo) used for certificate verification. In particular, we require that blinding and signing operations must commute. In this case, the signature can be blinded with the same nonce as the card's public key at the beginning of the session and later checked against the public key of the payment system directly in its blinded form. As a result, only the blinded version of the card's public key is ever revealed.

The equational theory E for the improved protocol is the equational theory E_0 in Fig. 3.13 extended with the property expressed in Fig. 3.16, which permits scalar multiplication and signing to commute.

$$\phi(M, \mathtt{sig}(N,K)) =_E \mathtt{sig}(\phi(M,N), K)$$

Figure 3.16: Equation for blinding extending the equational theory in Fig. 3.13.

It now follows from the blinding condition above and the last equation in Fig. 3.13 that the check of the signature, blinded with some blinding factor, returns the message, blinded with the same factor. This property of signatures in the equational theory E is used by the terminal when authenticating the card in our proposed update of the BDH protocol. The updated BDH protocol is presented informally in Fig. 3.17 and the corresponding formal π-calculus specification of the two roles involved is presented below.

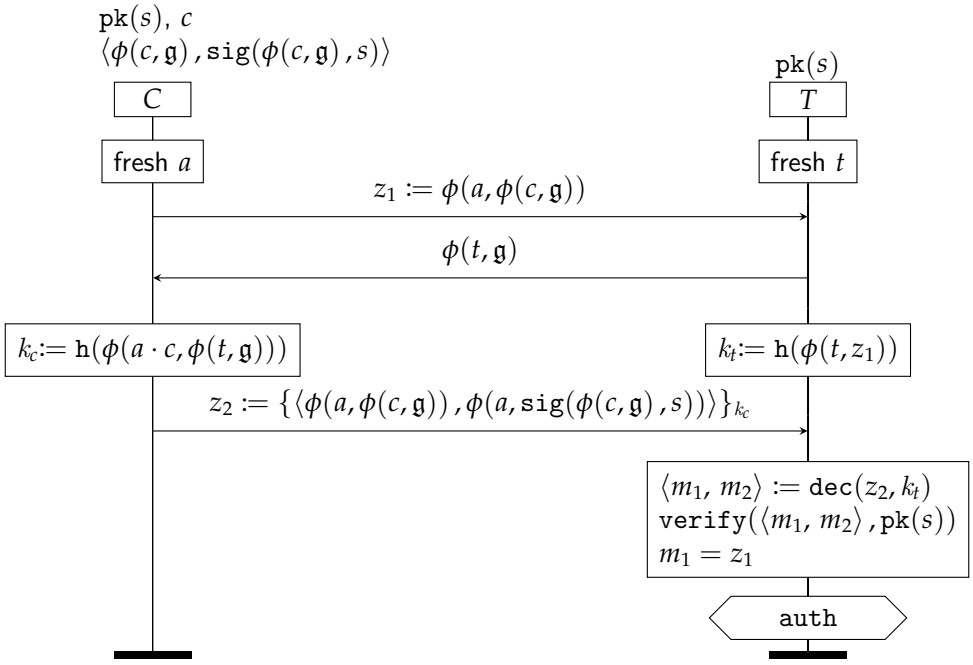


Figure 3.17: The Unlinkable BDH protocol.

Our version differs from the original proposal in message z_2 sent by the card to the terminal, i.e. now only the (encrypted) blinded certificate is transferred. At no point in the protocol, can the terminal unblind the card's public key since the blinding factor a is never revealed to any terminal.

We conclude this subsection by mentioning that there is a signature scheme satisfying both the blinding condition in Fig. 3.16, and the technical requirements of the BDH protocol [rfc12], namely the Verheul certification scheme [Ver01]. The scheme has been implemented on smart cards [BHJ+10] by Batina et al., using BN[7] curves [BN06] with time presenting one blinded certificate of 0.45 seconds, which is within the limit of 500ms of the card present in the reader field [emv21]. In the proposal [rfc12] EMVCo intends to use p256 curve, however switching over to a pairing-friendly BN curve would not introduce any slow-downs, compared to p256 curve, in on-card computation as was shown by Dzurenda et al. in the performance analysis [DRHM17a] of different elliptic curves on smart cards. It is now left to verify that UBDH is indeed unlinkable.

[7]The original paper [Ver01], in contrast, describes the system using symmetric pairings on a supersingular curve. This approach historically precedes the asymmetric pairings, making certain Decisional Diffie-Hellman problem simple and requires greater field size (which would slow down on-card computation) to achieve the same level of security as a non-supersingular curve based system [FST10].

$$
\begin{aligned}
&C_{\text{upd}}(s,c,ch) \triangleq \nu a.\overline{ch}\langle\phi(a,\phi(c,\mathfrak{g}))\rangle.\\
&\quad ch(y).\\
&\quad \texttt{let } k_c := \texttt{h}(\phi(a\cdot c,y)) \texttt{ in}\\
&\quad \texttt{let } m := \langle\phi(a,\phi(c,\mathfrak{g})),\phi(a,\texttt{sig}(\phi(c,\mathfrak{g}),s))\rangle \texttt{ in}\\
&\quad \texttt{ev:CRunning}\,(\phi(a,\phi(c,\mathfrak{g})),y,\{m\}_{k_c})\\
&\quad \overline{ch}\langle\{m\}_{k_c}\rangle
\end{aligned}
$$

$$
\begin{aligned}
&T_{\text{upd}}(pk_s,ch) \triangleq \nu t.ch(z_1).\\
&\quad \overline{ch}\langle\phi(t,\mathfrak{g})\rangle.\\
&\quad ch(z_2).\\
&\quad \texttt{let } k_t := \texttt{h}(\phi(t,z_1)) \texttt{ in}\\
&\quad \texttt{let } \langle m_1,m_2\rangle :=\\
&\quad \langle\texttt{fst}(\texttt{dec}(z_2,k_t)),\texttt{snd}(\texttt{dec}(z_2,k_t))\rangle \texttt{ in}\\
&\quad \texttt{if } m_1 = \texttt{check}(m_2,pk_s) \texttt{ then}\\
&\quad \texttt{if } m_1 = z_1 \texttt{ then}\\
&\quad \texttt{ev:TCommit}\,(z_1,\phi(t,\mathfrak{g}),z_2)\\
&\quad \overline{ch}\langle\texttt{auth}\rangle
\end{aligned}
$$

3.5.2 The proof of unlinkability of UBDH

This section is dedicated entirely to a detailed proof of unlinkability of the UBDH protocol in Fig. 3.17. Firstly we define UPD_{spec} and UPD_{impl} processes representing the idealised and the real-world behaviours of UBDH, and then formulate the main theorem of this chapter.

$$
\begin{aligned}
UPD_{\text{spec}} &\triangleq \nu s.\overline{out}\langle\texttt{pk}(s)\rangle.!\nu c.\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c,ch)\\
UPD_{\text{impl}} &\triangleq \nu s.\overline{out}\langle\texttt{pk}(s)\rangle.!\nu c.!\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c,ch)
\end{aligned}
$$

Theorem 7. *$C_{\text{upd}}(s,c,ch)$ satisfies unlinkability.*

Proof. By Def. 9 of unlinkability, we must show that $UPD_{\text{spec}} \sim UPD_{\text{impl}}$. Therefore we shall provide a quasi-open bisimulation relation $\mathfrak{R}$ such that $UPD_{\text{spec}}\ \mathfrak{R}\ UPD_{\text{impl}}$.

To define such $\mathfrak{R}$ we have to introduce some notation. Let L, $D \in \mathbb{N}$ be the number of sessions and the number of cards in the system, respectively. We use indices $l \in \{1,\ldots,L\}$ and $d \in \{1,\ldots,D\}$ to track sessions and cards.

Define $m^d(a,y)$ as the encrypted blinded certificate parametrised by the blinding factor a and the input y:

$$
m^d(a,y) := \{\langle\phi(a,\phi(c_d,\mathfrak{g})),\phi(a,\texttt{sig}(\phi(c_d,\mathfrak{g}),s))\rangle\}_{\texttt{h}(\phi(a\cdot c_d,y))}
$$

Define a partition $\Psi := \{\alpha, \beta, \gamma\ \delta\}$ of the set of all sessions $\{1,\ldots,L\}$, where α is the set of sessions in which the channel is created, but no message has been sent; β is the set of sessions in which the blinded public key has been sent but the response has not been received; γ is the set of all sessions in which the response

has been received but the encrypted blinded certificate has not been sent; δ is the set of all sessions in which the encrypted blinded certificate has been sent.

Define a partition $\Omega := \{\zeta^1, \ldots, \zeta^D\}$ of the set of all sessions $\{1, \ldots, L\}$, where ζ^d is the set of all sessions with the card d.

Let $\vec{Y} := (Y_1, \ldots, Y_L)$ be the list of inputs, where Y_l is the input in session l. Recall that Y_l can refer to messages already output on the network (as reflected in the last line of Fig. 3.18). Let $K := |\beta \cup \gamma \cup \delta|$ be the number of *started* sessions. Since we consider processes up to α-conversion and permutation of names (aka equivariance), we assume that a_l is the blinding factor in session l.

Finally, we define the following process subterms, which correspond to the elements of the partition Ψ.

$$\begin{aligned}
&\mathcal{E}^d(ch) \triangleq \nu a.\overline{ch}\langle \phi(a, \phi(c_d, \mathfrak{g}))\rangle.\mathcal{F}^d(ch, a)\\
&\mathcal{F}^d(ch, a) \triangleq ch(y).\mathcal{G}^d(ch, a, y)\\
&\mathcal{G}^d(ch, a, y) \triangleq \overline{ch}\langle m^d(a, y)\rangle\\
&\mathcal{H}^d \triangleq 0
\end{aligned}$$

The bisimulation relation $\mathfrak{R}$ is defined as *the least symmetric open relation* satisfying the constraints[8] in Fig. 3.18. Spelled out, we pair the reachable states of UPD_{spec} and UPD_{impl} based on the number of sessions and the respective stages of the card in a session. Notice that $UPD_{\text{spec}}\ \mathfrak{R}\ UPD_{\text{impl}}$ by the definition of $\mathfrak{R}$.

To prove that $\mathfrak{R}$ is indeed a quasi-open bisimulation, according to Def. 5, we must demonstrate

1. *(bisimulation)* Whenever $A\ \mathfrak{R}\ B$, and $A \xrightarrow{\pi} A'$, there exists B' such that $B \xrightarrow{\pi} B'$ and $A'\ \mathfrak{R}\ B'$.

2. *(openness)* $\mathfrak{R}$ is closed under the application of a substitution fresh for the domain of the frame of any of the related states.

3. *(static equivalence)* Whenever $A\ \mathfrak{R}\ B$, A is statically equivalent to B.

Bisimulation. Since $\mathfrak{R}$ is by definition a symmetric relation, we provide proof only for the cases when the left-side process starts first. Below we present the exhaustive list of possible transitions for the defining conditions of the relation $\mathfrak{R}$ in Fig. 3.18. Each transition is justified by the dedicated proof tree. In the proof trees by $\textsc{Rule}^n$ below we assume n applications of the transition rule Rule from Fig. 2.9. In case of n consecutive applications of rules Par-L, Par-R we write $\textsc{Par}^n$. Notice that α-conversion is often used – in particular when the rule Extrude is applied. For better presentation of proof trees we define the following subprocesses of UPD_{spec} and UPD_{impl} and two lists of private values.

$$\begin{aligned}
&S \triangleq \nu c.\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s, ch, c)\\
&I \triangleq \nu c.!\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s, ch, c)\\
&\vec{sp} = (s, c_1, \ldots, c_L, ch_1, \ldots, ch_L, a_{l_1}, \ldots, a_{l_K})\\
&\vec{im} = (s, c_1, \ldots, c_D, ch_1, \ldots, ch_L, a_{l_1}, \ldots, a_{l_K})
\end{aligned}$$

[8]The relation $\mathfrak{R}$ may not be the *smallest* quasi-open bisimilarity satisfying $UPD_{\text{spec}}\ \mathfrak{R}\ UPD_{\text{impl}}$.

$$UPD_{\text{spec}} \;\mathfrak{R}\; UPD_{\text{impl}}$$

$$UPD^{\Psi}_{\text{spec}}(\vec{Y}) \triangleq \nu s, c_1, \ldots, c_L, ch_1, \ldots, ch_L,$$
$$a_{l_1}, \ldots, a_{l_K}.(\sigma$$
$$\mid C_1 \mid \ldots \mid C_L$$
$$\mid !\nu c.\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s, c, ch))$$

$$\mathfrak{R}$$

$$UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \triangleq \nu s, c_1, \ldots, c_D, ch_1, \ldots, ch_L,$$
$$a_{l_1}, \ldots, a_{l_K}.(\theta$$
$$\mid \ldots \mid C^d_l \mid \ldots \mid !\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s, c_d, ch)$$
$$\mid !\nu c.!\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}((s, ch, c)))$$

$$C_l = \begin{cases} \mathcal{E}^l(ch_l) & \text{if } l \in \alpha \\ \mathcal{F}^l(ch_l, a_l) & \text{if } l \in \beta \\ \mathcal{G}^l(ch_l, a_l, Y_l\sigma) & \text{if } l \in \gamma \\ \mathcal{H}^l & \text{if } l \in \delta \end{cases}$$

$$C^d_l = \begin{cases} \mathcal{E}^d(ch_l) & \text{if } l \in \zeta^d \cap \alpha \\ \mathcal{F}^d(ch_l, a_l) & \text{if } l \in \zeta^d \cap \beta \\ \mathcal{G}^d(ch_l, a_l, Y_l\theta) & \text{if } l \in \zeta^d \cap \gamma \\ \mathcal{H}^d & \text{if } l \in \zeta^d \cap \delta \end{cases}$$

$$\begin{array}{ll} pk_s\sigma = \mathtt{pk}(s) & \\ u_l\sigma = ch_l & \text{if } l \in \{1, \ldots, L\} \\ v_l\sigma = \phi(a_l, \phi(c_l, \mathfrak{g})) & \text{if } l \in \beta \cup \gamma \cup \delta \\ w_l\sigma = m^l(a_l, Y_l\sigma) & \text{if } l \in \delta \end{array}$$

$$\begin{array}{ll} pk_s\theta = \mathtt{pk}(s) & \\ u_l\theta = ch_l & \text{if } l \in \{1, \ldots, L\} \\ v_l\theta = \phi(a_l, \phi(c_d, \mathfrak{g})) & \text{if } l \in \zeta^d \cap (\beta \cup \gamma \cup \delta) \\ w_l\theta = m^d(a_l, Y_l\theta) & \text{if } l \in \zeta^d \cap \delta \end{array}$$

$$\Psi := \{\alpha, \beta, \gamma, \delta\}, \;\; \Omega := \{\zeta^1, \ldots, \zeta^D\} \text{ are partitions of } \{1,\ldots,L\}$$

$$K := |\beta \cup \gamma \cup \delta| \quad l_1, \ldots, l_K \in \beta \cup \gamma \cup \delta$$

$$pk_s, u_l, v_l, w_l \;\#\; \{card, s\} \cup \{c_l, ch_l, a_l | l \in \{1, \ldots, L\}\}$$

$$Y_l \;\#\; \{s\} \cup \{c_l, ch_l, a_l | l \in \{1, \ldots, L\}\}$$

$$\text{fv}(Y_l) \cap (\{v_i | i \in \alpha\} \cup \{w_i | i \in \alpha \cup \beta \cup \gamma \cup \{l\}\}) = \emptyset$$

Figure 3.18: Defining conditions for the bisimulation relation $\mathfrak{R}$.

To make it easier for the reader to navigate, each transition case starts with a new page.

Case 1. $\overline{out}(pk_s)$, $UPD_{\text{spec}}\ \mathfrak{R}\ UPD_{\text{impl}}$.

The process UPD_{spec} can do the transition $\overline{out}(pk_s)$ to the state $UPD^{\emptyset}_{\text{spec}}(\emptyset)$, as justified by the proof tree in Fig. 3.19.

There is a state $UPD^{\emptyset,\emptyset}_{\text{impl}}(\emptyset)$ to which the process UPD_{impl} can do the transition $\overline{out}(pk_s)$, as justified by the proof tree in Fig. 3.20.

By the definition of $\mathfrak{R}$ we have $UPD^{\emptyset}_{\text{spec}}(\emptyset)\ \mathfrak{R}\ UPD^{\emptyset,\emptyset}_{\text{impl}}(\emptyset)$.

$$\dfrac{\dfrac{pk_s\ \#\ out, s, !S \quad out =_E out}{s\colon \overline{out}\langle \mathtt{pk}(s)\rangle.!S \xrightarrow{\overline{out}(pk_s)} \left(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\}\right) \mid !S \qquad s\ \#\ out, pk_s}\ \textsc{Res}}{\emptyset\colon UPD_{\text{spec}} \xrightarrow{\overline{out}(pk_s)} \nu s.\left(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\}\right) \mid !S}\ \textsc{Out}$$

Figure 3.19: *Case* 1. Transition $UPD_{\text{spec}} \xrightarrow{\overline{out}(pk_s)} UPD^{\emptyset}_{\text{spec}}(\emptyset)$.

$$\dfrac{\dfrac{pk_s\ \#\ out, s, !I \quad out =_E out}{s\colon \overline{out}\langle \mathtt{pk}(s)\rangle.!I \xrightarrow{\overline{out}(pk_s)} \left(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\}\right) \mid !I \qquad s\ \#\ out, pk_s}\ \textsc{Res}}{\emptyset\colon UPD_{\text{impl}} \xrightarrow{\overline{out}(pk_s)} \nu s.\left(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\}\right) \mid !I}\ \textsc{Out}$$

Figure 3.20: *Case* 1. Transition $UPD_{\text{impl}} \xrightarrow{\overline{out}(pk_s)} UPD^{\emptyset,\emptyset}_{\text{impl}}(\emptyset)$.

Case 2. $\overline{card}(u_{L+1})$, $UPD^{\Psi}_{\text{spec}}(\vec{Y})\ \mathfrak{R}\ UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$.

The process $UPD^{\Psi}_{\text{spec}}(\vec{Y})$ can do the transition $\overline{card}(u_{L+1})$ to the state $CH_{\text{spec}} \triangleq UPD^{\{\alpha\cup\{L+1\},\beta,\gamma,\delta\}}_{\text{spec}}((Y_1,\ldots,Y_L,\emptyset))$ as justified by the proof tree in Fig. 3.21.

Following the transition $\overline{card}(u_{L+1})$, in the process $UPD^{\Psi,\Omega}_{\text{impl}}$ some card d starts a new session with the following resulting state.

$$CH_{\text{impl}} \triangleq UPD^{\{\alpha\cup\{L+1\},\beta,\gamma,\delta\},\{\ldots,\zeta^d\cup\{L+1\},\ldots\}}_{\text{impl}}((Y_1,\ldots,Y_L,\emptyset))$$

The transition is justified by the proof tree in Fig. 3.22.

Or, following the same transition $\overline{card}(u_{L+1})$ in the process $UPD^{\Psi,\Omega}_{\text{impl}}$ the new card is created and the resulting state is as follows.

$$CHC_{\text{impl}} \triangleq UPD^{\{\alpha\cup\{L+1\},\beta,\gamma,\delta\},\Omega\cup\{\{L+1\}\}}_{\text{impl}}((Y_1,\ldots,Y_L,\emptyset))$$

The transition is justified by the proof tree in Fig. 3.23.

In both cases we have $CH_{\text{spec}}\ \mathfrak{R}\ CH_{\text{impl}}$ and $CH_{\text{spec}}\ \mathfrak{R}\ CHC_{\text{impl}}$ by the definition of $\mathfrak{R}$.

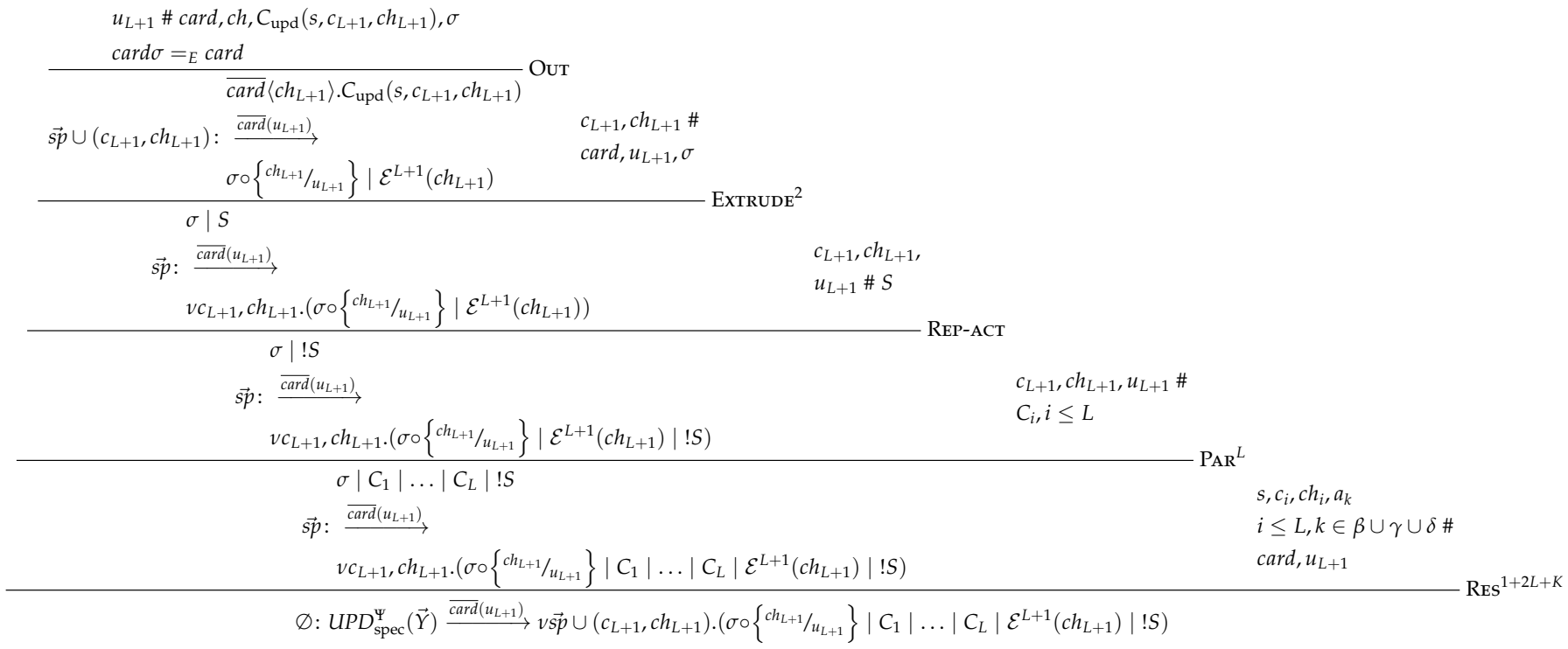


Figure 3.21: *Case* 2. Transition $UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{\overline{card}(u_{L+1})} UPD^{\{\alpha \cup \{L+1\}, \beta, \gamma, \delta\}}_{\text{spec}}((Y_1, \ldots, Y_L, \emptyset))$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\theta \mid \overline{card}\langle ch_{L+1}\rangle.C_{\text{upd}}(s,c_d,ch_{L+1})}\ \textsc{Out}\quad \begin{array}{l} u_{L+1} \# card, ch_{L+1}, C_{\text{upd}}(s,c_d,ch_{L+1}), \theta \\ card\theta =_E card \end{array}}{\vec{im} \cup (ch_{L+1}) \xrightarrow{\overline{card}(u_{L+1})} \theta\circ\left\{{}^{ch_{L+1}}/_{u_{L+1}}\right\} \mid \mathcal{E}^d(ch_{L+1})}\ \textsc{Extrude}\quad \begin{array}{l} ch_{L+1} \# \\ card, u_{L+1}, \theta \end{array}}{\theta \mid \nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c_d,ch)}\ \textsc{Rep-act}\quad \begin{array}{l} ch_{L+1}, u_{L+1} \# \\ \nu ch.\overline{card}\langle ch\rangle. \\ C_{\text{upd}}(s,c_d,ch) \end{array}}{\vec{im}: \xrightarrow{\overline{card}(u_{L+1})} \nu ch_{L+1}.(\theta\circ\left\{{}^{ch_{L+1}}/_{u_{L+1}}\right\} \mid \mathcal{E}^d(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c_d,ch))}\ \textsc{Rep-act}\quad \begin{array}{l} ch_{L+1}, u_{L+1} \# \\ C^i_j, i \le D, j \le \max\limits_{i\le D} L_i; \\ \nu ch.\overline{card}\langle ch\rangle. \\ C_{\text{upd}}(s,c_i,ch), \\ i \le D, i \ne d; !I \end{array}}{\theta \mid \ldots \mid !\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c_d,ch) \mid \ldots \mid !I}\ \textsc{Par}^{D+L}\quad \begin{array}{l} s, c_i, ch_j, a_k, \\ i \le D, j \le L, k \in \beta \cup \gamma \cup \delta \# \\ card, u_{L+1} \end{array}}{\vec{im}: \xrightarrow{\overline{card}(u_{L+1})} \nu ch_{L+1}.(\theta\circ\left\{{}^{ch_{L+1}}/_{u_{L+1}}\right\} \mid \ldots \mid \mathcal{E}^d(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c_d,ch) \mid \ldots \mid !I)}\ \textsc{Res}^{1+D+L+K}$$

$$\emptyset: UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{card}(u_{L+1})} \nu\vec{im} \cup (ch_{L+1}).(\theta\circ\left\{{}^{ch_{L+1}}/_{u_{L+1}}\right\} \mid \ldots \mid \mathcal{E}^d(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.C_{\text{upd}}(s,c_d,ch) \mid \ldots \mid !I)$$

Figure 3.22: *Case* 2. Transition $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{card}(u_{L+1})} UPD^{\{\alpha\cup\{L+1\},\beta,\gamma,\delta\},\{\ldots,\zeta^d\cup\{L+1\},\ldots\}}_{\text{impl}}((Y_1,\ldots,Y_L,\emptyset))$: card d starts new session.

$$\dfrac{u_{L+1} \,\#\, card, ch_{L+1}, \mathrm{C_{upd}}(s, c_{D+1}, ch_{L+1}), \theta \qquad card\theta =_E card}{\vec{im} \cup (c_{D+1}, ch_{L+1}) : \theta \mid \overline{card}\langle ch_{L+1}\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch_{L+1}) \xrightarrow{\overline{card}(u_{L+1})} \theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \mathcal{E}^{D+1}(ch_{L+1})}\ \textsc{Out}$$

$$\dfrac{\cdots \qquad ch_{L+1} \,\#\, card, u_{L+1}, \theta}{\vec{im} \cup (ch_{L+1}) : \theta \mid \nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch) \xrightarrow{\overline{card}(u_{L+1})} \nu ch_{L+1}.(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \mathcal{E}^{D+1}(ch_{L+1})}\ \textsc{Extrude}$$

$$\dfrac{\cdots \qquad ch_{L+1}, u_{L+1} \,\#\, \nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch)}{\vec{im} \cup (ch_{L+1}) : \theta \mid !\nu ch.\overline{card}\langle ch_{L+1}\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch) \xrightarrow{\overline{card}(u_{L+1})} \nu ch_{L+1}.(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \mathcal{E}^{D+1}(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch))}\ \textsc{Rep-act}$$

$$\dfrac{\cdots \qquad c_{D+1} \,\#\, card, u_{L+1}, \theta}{\vec{im} : \theta \mid I \xrightarrow{\overline{card}(u_{L+1})} \nu c_{D+1}, ch_{L+1}.(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \mathcal{E}^{D+1}(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch))}\ \textsc{Extrude}$$

$$\dfrac{\cdots \qquad c_{D+1}, ch_{L+1}, u_{L+1} \,\#\, I}{\vec{im} : \theta \mid !I \xrightarrow{\overline{card}(u_{L+1})} \nu c_{D+1}, ch_{L+1}.(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \mathcal{E}^{D+1}(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch) \mid !I)}\ \textsc{Rep-act}$$

$$\dfrac{\cdots \qquad c_{D+1}, ch_{L+1}, u_{L+1} \,\#\, C^i_j,\ i \leq D, j \leq \max_{i \leq D} L_i;\ \nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_d, ch)}{\vec{im} : \theta \mid \ldots \mid !I \xrightarrow{\overline{card}(u_{L+1})} \nu c_{D+1}, ch_{L+1}.(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \ldots \mid \mathcal{E}^{D+1}(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch) \mid !I)}\ \textsc{Par}^{D+L}$$

$$\dfrac{\cdots \qquad s, c_i, ch_j, a_k,\ i \leq D, j \leq L,\ k \in \beta \cup \gamma \cup \delta \,\#\, card, u_{L+1}}{\emptyset : UPD^{\Psi,\Omega}_{\mathrm{impl}}(\vec{Y}) \xrightarrow{\overline{card}(u_{L+1})} \nu \vec{im} \cup (c_{D+1}, ch_{L+1}).(\theta \circ \left\{ {}^{ch_{L+1}}/_{u_{L+1}} \right\} \mid \ldots \mid \mathcal{E}^{D+1}(ch_{L+1}) \mid !\nu ch.\overline{card}\langle ch\rangle.\mathrm{C_{upd}}(s, c_{D+1}, ch) \mid !I)}\ \textsc{Res}^{1+D+L+K}$$

Figure 3.23: *Case* 2. Transition $UPD^{\Psi,\Omega}_{\mathrm{impl}}(\vec{Y}) \xrightarrow{\overline{card}(u_{L+1})} UPD^{\{\alpha \cup \{L+1\}, \beta, \gamma, \delta\}, \Omega \cup \{\{L+1\}\}}_{\mathrm{impl}}((Y_1, \ldots, Y_L, \emptyset))$: a new card is created.

Case 3. $\overline{u_l}(v_l)$, $UPD^{\Psi}_{\text{spec}}(\vec{Y})\ \mathfrak{R}\ UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$, and $l \in \alpha$.

The process $UPD^{\Psi}_{\text{spec}}(\vec{Y})$ can make the transition $\overline{u_l}(v_l)$ to the state $APK_{\text{spec}} \triangleq UPD^{\{\alpha\setminus\{l\},\beta\cup\{l\},\gamma,\delta\}}_{\text{spec}}(\vec{Y})$ as justified by the proof tree in Fig. 3.24.

There is a state $APK_{\text{impl}} \triangleq UPD^{\alpha\setminus\{l\},\beta\cup\{l\},\gamma,\delta\},\Omega}_{\text{impl}}(\vec{Y})$ to which the process $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$ can do the transition $\overline{u_l}(v_l)$ as justified by the proof tree in Fig. 3.25.

By the definition of $\mathfrak{R}$ we have $APK_{\text{spec}}\ \mathfrak{R}\ APK_{\text{impl}}$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vec{sp} \cup (a_l) \colon \sigma \mid \overline{ch_l}\langle \phi(a_l, \phi(c_l, \mathfrak{g}))\rangle.\mathcal{F}^l(ch_l, a_l) \xrightarrow{\overline{u_l}(v_l)} \sigma \circ \left\{ {}^{\phi(a_l,\phi(c_l,\mathfrak{g}))}/_{v_l} \right\} \mid \mathcal{F}^l(ch_l, a_l)} \text{Out} \quad a_l \# u_l, v_l, \sigma}{\vec{sp} \colon \sigma \mid \nu a.\overline{ch_l}\langle \phi(a, \phi(c_l, \mathfrak{g}))\rangle.\mathcal{F}^l(ch_l, a) \xrightarrow{\overline{u_l}(v_l)} \nu a_l.(\sigma \circ \left\{ {}^{\phi(a_l,\phi(c_l,\mathfrak{g}))}/_{v_l} \right\} \mid \mathcal{F}^l(ch_l, a_l))} \text{Extrude} \quad a_l, v_l \# C_i, i \le L, i \ne l; !S}{\vec{sp} \colon \sigma \mid C_1 \mid \ldots \mid \mathcal{E}^l(ch_l) \mid \ldots \mid C_L \mid !S \xrightarrow{\overline{u_l}(v_l)} \nu a_l.(\sigma \circ \left\{ {}^{\phi(a_l,\phi(c_l,\mathfrak{g}))}/_{v_l} \right\} \mid \ldots \mid C_K \mid \ldots \mid \mathcal{F}^l(ch_l, a_l) \mid \ldots \mid !S)} \text{Par}^L \quad \begin{array}{l} s, c_i, ch_i, a_k \\ i \le L, k \in \beta \cup \gamma \cup \delta \# u_l, v_l \end{array}}{\varnothing \colon UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{\overline{u_l}(v_l)} \nu \vec{sp} \cup (a_l).(\sigma \circ \left\{ {}^{\phi(a_l,\phi(c_l,\mathfrak{g}))}/_{v_l} \right\} \mid \ldots \mid C_K \mid \ldots \mid \mathcal{F}^l(ch_l, a_l) \mid \ldots \mid !S)} \text{Res}^{1+2L+K}$$

$$v_l \# u_l, a_l, \mathcal{F}^l(ch_l, a_l), \sigma \qquad u_l\sigma =_E ch_l$$

Figure 3.24: *Case* 3. Transition $UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{\overline{u_l}(v_l)} UPD^{\{\alpha\setminus\{l\},\beta\cup\{l\},\gamma,\delta\}}_{\text{spec}}(\vec{Y})$, $l \in \alpha$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\vec{im} \cup (a_l) \colon \theta \mid \overline{ch_l}\langle \phi(a_l, \phi(c_d, \mathfrak{g}))\rangle.\mathcal{F}^d(ch_l, a_l) \xrightarrow{\overline{u_l}(v_l)} \theta \circ \left\{ {}^{\phi(a_l,\phi(c_d,\mathfrak{g}))}/_{v_l} \right\} \mid \mathcal{F}^d(ch_l, a_l)} \text{Out} \quad a_l \# u_l, v_l, \theta}{\vec{im} \colon \theta \mid \nu a.\overline{ch_l}\langle \phi(a, \phi(c_d, \mathfrak{g}))\rangle.\mathcal{F}^d(ch_l, a) \xrightarrow{\overline{u_l}(v_l)} \nu a_l.(\theta \circ \left\{ {}^{\phi(a_l,\phi(c_d,\mathfrak{g}))}/_{v_l} \right\} \mid \mathcal{F}^d(ch_l, a_l))} \text{Extrude} \quad \begin{array}{l} a_l, v_l \# C^i_j, \\ i \le D, j \le \max\limits_{i \le D} L_i, \\ j \ne l; !I \end{array}}{\vec{im} \colon \theta \mid \ldots \mid \mathcal{E}^d(ch_l) \mid \ldots \mid !I \xrightarrow{\overline{u_l}(v_l)} \nu a_l.(\theta \circ \left\{ {}^{\phi(a_l,\phi(c_d,\mathfrak{g}))}/_{v_l} \right\} \mid \ldots \mid \mathcal{F}^d(ch_l, a_l) \mid \ldots \mid !I)} \text{Par}^{D+L} \quad \begin{array}{l} s, c_i, ch_j, a_k \\ i \le D, j \le L, k \in \beta \cup \gamma \cup \delta \# \\ u_l, v_l \end{array}}{\varnothing \colon UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{u_l}(v_l)} \nu \vec{im} \cup (a_l).(\theta \circ \left\{ {}^{\phi(a_l,\phi(c_d,\mathfrak{g}))}/_{v_l} \right\} \mid \ldots \mid \mathcal{F}^d(ch_l, a_l) \mid \ldots \mid !I)} \text{Res}^{1+D+L+K}$$

$$v_l \# u_l, a_l, \mathcal{F}^d(ch_l, a_l), \theta \qquad u_l\theta =_E ch_l$$

Figure 3.25: *Case* 3. Transition $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{u_l}(v_l)} UPD^{\alpha\setminus\{l\},\beta\cup\{l\},\gamma,\delta\},\Omega}_{\text{impl}}(\vec{Y})$, $l \in \alpha$.

Case 4. $u_l\, Y_l$, $UPD^{\Psi}_{\text{spec}}(\vec{Y})\ \mathfrak{R}\ UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$, and $l \in \beta$.

Let $\chi_l(\vec{Y}, M)$ be the list of message terms obtained from $\vec{Y}$ by the replacement of lth entry in $\vec{Y}$ with M. The process $UPD^{\Psi}_{\text{spec}}(\vec{Y})$ can make the transition $u_l\, Y_l$ to the state $IN_{\text{spec}} \triangleq UPD^{\{\alpha,\beta\setminus\{l\},\gamma\cup\{l\},\delta\}}_{\text{spec}}(\chi_l(\vec{Y}, Y_l))$ as justified by the proof tree in Fig. 3.26.

There is a state $IN_{\text{impl}} \triangleq UPD^{\{\alpha,\beta\setminus\{l\},\gamma\cup\{l\},\delta\},\Omega}_{\text{impl}}(\chi_l(\vec{Y}, Y_l))$ to which the initial process $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$ can make the transition $u_l\, Y_l$, as justified by the proof tree in Fig. 3.27.

By the definition of $\mathfrak{R}$ we have $IN_{\text{spec}}\ \mathfrak{R}\ IN_{\text{impl}}$.

$$\dfrac{\dfrac{\dfrac{u_l\sigma =_E ch_l}{\vec{sp}\colon \sigma \mid ch_l(y).\mathcal{G}^l(ch_l,a_l,y) \xrightarrow{u_l\, Y_l} \sigma \mid \mathcal{G}^l(ch_l,a_l,Y_l\sigma)}\,\textsc{Inp}}{\vec{sp}\colon \sigma \mid \mathsf{C}_1 \mid \ldots \mid \mathcal{F}^l(ch_l,a_l) \mid \ldots \mid \mathsf{C}_L \mid !S \xrightarrow{u_l\, Y_l} \sigma \mid \ldots \mid \mathcal{G}^l(ch_l,a_l,Y_l\sigma) \mid \ldots \mid !S}\,\textsc{Par}^L \quad \begin{array}{l} s, c_i, ch_i, a_k \\ i \le L, k \in \beta \cup \gamma \cup \delta \,\#\, u_l, Y_l \end{array}}{\varnothing\colon UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{u_l\, Y_l} \nu \vec{sp}.\{\sigma \mid \ldots \mid \mathcal{G}^l(ch_l,a_l,Y_l\sigma) \mid \ldots \mid !S\}}\,\textsc{Res}^{1+2L+K}$$

Figure 3.26: *Case* 4. Transition $UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{u_l\, Y_l} UPD^{\{\alpha,\beta\setminus\{l\},\gamma\cup\{l\},\delta\}}_{\text{spec}}(\chi_l(\vec{Y},Y_l))$, $l \in \beta$.

$$\dfrac{\dfrac{\dfrac{u_l\theta =_E ch_l}{\vec{im}\colon \theta \mid ch_l(y).\mathcal{G}^d(ch_l,a_l,y) \xrightarrow{u_l\, Y_l} \theta \mid \mathcal{G}^d(ch_l,a_l,Y_l\sigma)}\,\textsc{Inp}}{\vec{im}\colon \theta \mid \ldots \mid \mathcal{F}^d(ch_l,a_l) \mid \ldots \mid !I \xrightarrow{u_l\, Y_l} \theta \mid \ldots \mid \mathcal{G}^d(ch_l,a_l,Y_l\sigma) \mid \ldots \mid !I}\,\textsc{Par}^{D+L} \quad \begin{array}{l} s, c_i, ch_j, a_k \\ i \le D, j \le L, k \in \beta \cup \gamma \cup \delta \,\# \\ u_l, Y_l \end{array}}{\varnothing\colon UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{u_l\, Y_l} \nu \vec{im}.\{\theta \mid \ldots \mid \mathcal{G}^d(ch_l,a_l,Y_l\sigma) \mid \ldots \mid !I\}}\,\textsc{Res}^{1+D+L+K}$$

Figure 3.27: *Case* 4. Transition $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{u_l\, Y_l} UPD^{\{\alpha,\beta\setminus\{l\},\gamma\cup\{l\},\delta\},\Omega}_{\text{impl}}(\chi_l(\vec{Y},Y_l))$ if there is a card at the stage $\mathcal{F}$.

Case 5. $\overline{u_l}(w_l)$, $UPD^{\Psi}_{\text{spec}}(\vec{Y})$ $\mathfrak{R}$ $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$, and $l \in \beta$.

The process $UPD^{\Psi}_{\text{spec}}(\vec{Y})$ can make a transition $\overline{u_l}(w_l)$ to the state $CRT_{\text{spec}} \triangleq UPD^{\{\alpha,\beta,\gamma\setminus\{l\},\delta\cup\{l\}\}}_{\text{spec}}(\vec{Y})$, as justified by the proof tree in Fig. 3.28, where σ' defines the substitution accumulated at the point of input of Y_l.

There is a state $CRT_{\text{impl}} \triangleq UPD^{\{\alpha,\beta,\gamma\setminus\{l\},\delta\cup\{l\}\},\Omega}_{\text{impl}}(\vec{Y})$ to which $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y})$ can make a transition $\overline{ch_l}(w_l)$, as justified by the proof tree in Fig. 3.29, where θ' the frame accumulated at the point of input of Y_l.

By the definition of $\mathfrak{R}$ we have CRT_{spec} $\mathfrak{R}$ CRT_{impl}.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\begin{array}{l} w_l \# u_l, m^l(a_l, Y_l\sigma'), \sigma \\ u_l\theta =_E ch_l \end{array}}\ \textsc{Out}}{\vec{sp}\colon \sigma \mid \overline{ch_l}\left\langle m^l(a_l, Y_l\sigma')\right\rangle \xrightarrow{\overline{u_l}(w_l)} \sigma\circ\left\{{}^{m^l(a_l,Y_l\sigma')}/_{w_l}\right\} \mid \mathcal{H}^l}\ \textsc{Par}^L}{\vec{sp}\colon \sigma \mid C_1 \mid \ldots \mid \mathcal{G}^l(ch_l, a_l, Y_l\sigma') \mid \ldots \mid C_L \mid !S \xrightarrow{\overline{u_l}(w_l)} \sigma\circ\left\{{}^{m^l(a_l,Y_l\sigma')}/_{w_l}\right\} \mid \ldots \mid \mathcal{H}^l \mid \ldots \mid !S}\ \begin{array}{l} s, c_i, ch_i, a_k \\ i \le L, k \in \beta \cup \gamma \cup \delta \,\#\, u_l, w_l \end{array}\ \textsc{Res}^{1+2L+K}$$

$$\emptyset\colon UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{\overline{u_l}(w_l)} \nu\vec{sp}.\{\sigma\circ\left\{{}^{m^l(a_l,Y_l\sigma')}/_{w_l}\right\} \mid \ldots \mid \mathcal{H}^l \mid \ldots \mid !S\}$$

Figure 3.28: *Case* 5. Transition $UPD^{\Psi}_{\text{spec}}(\vec{Y}) \xrightarrow{\overline{u_l}(w_l)} UPD^{\{\alpha,\beta,\gamma\setminus\{l\},\delta\cup\{l\}\}}_{\text{spec}}(\vec{Y})$, $l \in \gamma$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\begin{array}{l} w_l \# u_l, m^d(a_l, Y_l\theta'), \theta \\ u_l\theta =_E ch_l \end{array}}\ \textsc{Out}}{\vec{im}\colon \theta \mid \overline{ch_l}\left\langle m^d(a_l, Y_l\theta')\right\rangle \xrightarrow{\overline{u_l}(w_l)} \theta\circ\left\{{}^{m^d(a_l,Y_l\theta')}/_{w_l}\right\} \mid \mathcal{H}^d}\ \textsc{Par}^{D+L}}{\vec{im}\colon \theta \mid \ldots \mid \mathcal{G}^d(ch_l, a_l, Y_l\theta') \mid \ldots \mid !I \xrightarrow{\overline{u_l}(w_l)} \theta\circ\left\{{}^{m^d(a_l,Y_l\theta')}/_{w_l}\right\} \mid \ldots \mid \mathcal{H}^d \mid \ldots \mid !I}\ \begin{array}{l} s, c_i, ch_j, a_k \\ i <\!\!\le D, j \le L, k \in \beta \cup \gamma \cup \delta \,\# \\ u_l, w_l \end{array}\ \textsc{Res}^{1+D+L+K}$$

$$\emptyset\colon UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{u_l}(w_l)} \nu\vec{im}.\{\theta\circ\left\{{}^{m^d(a_l,Y_l\theta')}/_{w_l}\right\} \mid \ldots \mid \mathcal{H}^d \mid \ldots \mid !I\}$$

Figure 3.29: *Case* 5. Transition $UPD^{\Psi,\Omega}_{\text{impl}}(\vec{Y}) \xrightarrow{\overline{u_l}(w_l)} UPD^{\{\alpha,\beta,\gamma\setminus\{l\},\delta\cup\{l\}\},\Omega}_{\text{impl}}(\vec{Y})$, $l \in \gamma$.

Openness. $\mathfrak{R}$, by definition, is open: whenever $A\ \mathfrak{R}\ B$, then $A'\ \mathfrak{R}\ B'$ for any A', B' accessible from A, B by the environment extension $\nu\vec{z}.\rho$ satisfying conditions in Def. 4. Accessibility cannot disable transitions available from the initial extended process – the only variables that can be affected by $\nu\vec{z}.\rho$ are channel names *out*, *card*, and free variables in the inputs Y_l. Recall, that if the free channel name is affected, as briefly explained in Sec. 2.2.2 after Def. 4, then the channel name is available for using as the message alias, hence the transition label remains the same. In case free variables in the entries of the input vector $\vec{Y}$ are affected, it is straightforward to replace $\vec{Y}$ with $\vec{Y}\rho := (Y_1\rho, \ldots, Y_L\rho)$, where $\emptyset\rho := \emptyset$ and modify freshness conditions in the proof trees above by applying the substitution ρ. Notice that by α-conversion we may assume that the range of ρ does not contain variables "reserved" for future outputs or private nonces: $\mathrm{fv}(y\rho) \cap (\{s, pk_s\} \cup \{c_i, ch_i, a_i, u_i, v_i, w_i | l \in \mathbb{N}\}) = \emptyset$ for any y.

Static equivalence. To conclude, we prove that A is statically equivalent to B whenever $A\ \mathfrak{R}\ B$. There is nothing to prove in the case of $UPD_{\mathrm{spec}}\ \mathfrak{R}\ UPD_{\mathrm{impl}}$ since frames are empty. The proof for the case $UPD^{\Psi}_{\mathrm{spec}}(\vec{Y})\ \mathfrak{R}\ UPD^{\Psi,\Omega}_{\mathrm{impl}}(\vec{Y})$ is presented separately in Lemma 1. □

Recall that the static equivalence (Def. 3) captures the indistinguishability between two snapshots (frames) of the system, i.e. $\nu\vec{x}.(\sigma \mid \_)$ is statically equivalent to $\nu\vec{y}.(\theta \mid \_)$ whenever $M\sigma =_E N\sigma$ if and only if $M\theta =_E N\theta$ for any messages M, N that cannot refer to private names in $\vec{x}$ and $\vec{y}$ called recipes. Our high-level strategy of proving static equivalence is as follows. Firstly, we identify building blocks of message terms in the range of substitutions of the extended processes $UPD^{\Psi}_{\mathrm{spec}}(\vec{Y})$ and $UPD^{\Psi,\Omega}_{\mathrm{impl}}(\vec{Y})$, moreover we enforce these building blocks to be in a certain normal form that we call weak. Secondly, we prove static equivalence by induction on the structure of the weak normal form of $N\sigma$ exploring all cases allowed by the grammar in Fig. 3.12.

We start by introducing the weak notion of the *normal form* $M\downarrow$ of a message term M, that captures the least complex, up to multiplication, expression of M; and the notion of the *normalisation* of a frame $\nu\vec{x}.(\sigma \mid \_)$ with respect to the equational theory E, which is a saturation of the range of σ with weak normal forms of messages that have a recipe under σ. Recipes can be conveniently recorded in the domain of normalisation. While both notions depend on the message theory, we will drop the name E of the message theory everywhere except definitions as it is clear from the context. First we introduce few technical notions.

Definition 10. *(m-atomic, ϕ-atomic) A message term M is m-atomic if there are no such M_1, M_2, s.t. $M =_E M_1 \cdot M_2$; it is ϕ-atomic if there are no such M_1, M_2, s.t. $M =_E \phi(M_1, M_2)$*

Definition 11. *(m-atomic, ϕ-atomic) A subterm N of M is an* immediate *m*-factor *if it is m-atomic and there is a message term K, s.t. $N \cdot K = M$.*

Definition 12. *(non-trivial recipe) The recipe M is* non-trivial *under σ if* $\mathrm{fv}(M\downarrow) \cap \mathrm{dom}(\sigma) \neq \emptyset$.

The *weak normal form* $M\!\downarrow$ of a term M with respect to E captures the least complex (up to multiplication) expression of M. Notice that we do not require the weak normal form to be unique.

Definition 13. *(E-weak normal form) The E-weak normal $M\!\downarrow$ of a message term M is defined inductively on the structure of M:*

- $M = \mathfrak{g}$ *or M is a variable, then* $M\!\downarrow = M$.
- $M = M_1 \cdot M_2$, *then* $M\!\downarrow = M_1\!\downarrow \cdot M_2\!\downarrow$.
- $M = \phi(M_1, M_2)$, *then* $M\!\downarrow = \phi(M_1\!\downarrow, M_2\!\downarrow)$ *if $M_2\!\downarrow$ is ϕ-atomic. Otherwise* $M\!\downarrow = \phi(M_1\!\downarrow \cdot M_2'\!\downarrow, M_2''\!\downarrow)$, *where* $M_2 =_E \phi(M_2', M_2'')$ *and $M_2''\!\downarrow$ is ϕ-atomic.*
- $M = \langle M_1, M_2 \rangle$, *then* $M\!\downarrow = \langle M_1\!\downarrow, M_2\!\downarrow \rangle$.
- $M = \mathtt{h}(M_1)$ *or* $M = \mathtt{pk}(M_1)$, *then* $M\!\downarrow = \mathtt{h}(M_1\!\downarrow)$ *or* $M\!\downarrow = \mathtt{pk}(M_1\!\downarrow)$ *respectively.*
- $M = \mathtt{sig}(M_1, M_2)$, *then* $M\!\downarrow = \mathtt{sig}(M_1\!\downarrow, M_2\!\downarrow)$ *if $M_1\!\downarrow$ is ϕ-atomic. Otherwise* $M\!\downarrow = \phi(M_1'\!\downarrow, \mathtt{sig}(M_1''\!\downarrow, M_2\!\downarrow))$, *where* $M_1 = \phi(M_1', M_1'')$ *and $M_1''\!\downarrow$ is ϕ-atomic.*
- $M = \mathtt{fst}(\langle M_1, M_2 \rangle)$ *or* $M = \mathtt{snd}(\langle M_1, M_2 \rangle)$ *then* $M\!\downarrow = M_1\!\downarrow$ *or* $M\!\downarrow = M_2\!\downarrow$ *respectively.*
- $M = \mathtt{dec}(\{M_1\}_{M_2}, M_2)$, *then* $M\!\downarrow = M_1\!\downarrow$.
- $M = \mathtt{check}(\mathtt{sig}(M_1, M_2), \mathtt{pk}(M_2))$, *then* $M\!\downarrow = M_1\!\downarrow$.
- *Otherwise* $M\!\downarrow = M$.

The normalisation of a frame $\nu\vec{z}.(\rho \mid \_)$ denoted as $\rho\!\downarrow_E^{\vec{z}}$, is a frame with recipes allowed in the domain constructed as follows.

1. $u\rho = M$ for any $u \in \mathrm{dom}(\rho)$ is replaced by $u\rho = M\!\downarrow$.
2. If $u\rho = K_1 \cdot K_2$ and there is a recipe M_1 for an immediate m-factor K_1, then $M_1\rho$ is added to the normalisation. If there is a recipe M_2 for an immediate m-factor K_2, then $M_2\rho$ is also added to the normalisation.
3. If $u\rho = \langle K_1, K_2 \rangle$, then $u\rho$ is replaced by $\mathtt{fst}(u)\,\rho = K_1$ and $\mathtt{snd}(u)\,\rho = K_2$.
4. If $u\rho = \{K_1\}_{K_2}$ and there is a recipe M_2 for K_2, then $u\rho$ is replaced by $\mathtt{dec}(u, M_2)\,\rho = K_1$.
5. If $u\rho = \mathtt{sig}(N_1, N_2)$ and there is a recipe M_2 for N_2, then $u\rho$ is replaced by $\mathtt{check}(u, \mathtt{pk}(M_2))\,\rho = N_1$.
6. If $u\rho = \mathtt{sig}(N_1, N_2)$ and there is a recipe M_2 for $\mathtt{pk}(N_2)$, then $\mathtt{check}(u, M_2)\,\rho = N_1$ is added to the normalisation.

To give an example, consider the following frame.

$$\nu\vec{z}.(\rho \mid \_) \triangleq \nu s, a, b.\left(\left\{{}^{\mathtt{pk}(s),\{\mathtt{h}(a)\}_b,b,\mathtt{sig}(\langle a,x\rangle,s)}/_{pk_s,u_1,u_2,u_3}\right\} \mid \_\right)$$

Then the normalisation of ρ is given below.

$$
\begin{array}{l}
pk_s\rho = \mathtt{pk}(s) \\
\mathtt{dec}(u_1, u_2)\,\rho = \mathtt{h}(a) \\
u_2\rho = b \\
\mathtt{fst}(\mathtt{check}(u_3, pk_s))\,\rho = a \\
\mathtt{snd}(\mathtt{check}(u_3, pk_s))\,\rho = x \\
u_3\rho = \mathtt{sig}(\langle a, x\rangle, s)
\end{array}
$$

We conclude the proof of Theorem 7 with the following.

Lemma 1. *$UPD^{\Psi}_{spec}(\vec{Y})$ is statically equivalent to $UPD^{\Psi,\Omega}_{impl}(\vec{Y})$.*

Proof. Considering the definition of $\mathfrak{R}$ in Fig. 3.18, let $\nu\vec{x}.\,(\sigma \mid P) \triangleq UPD^{\Psi}_{\mathrm{spec}}(\vec{Y})$ and $\nu\vec{y}.\,(\theta \mid Q) \triangleq UPD^{\Psi,\Omega}_{\mathrm{impl}}(\vec{Y})$. We aim to show that $\nu\vec{x}.\,(\sigma \mid \_)$ is statically equivalent to $\nu\vec{y}.\,(\theta \mid \_)$. Since $\vec{x}$ is always a superset of $\vec{y}$, we prove that for all messages M and N, s.t. $\vec{x} \,\#\, M, N$, we have $M\sigma =_E N\sigma$ if and only if $M\theta =_E N\theta$.

Recall the definition of $m^d(a, y)$:

$$
m^d(a, y) \coloneqq \{\langle \phi(a, \phi(c_d, \mathfrak{g})), \phi(a, \mathtt{sig}(\phi(c_d, \mathfrak{g}), s))\rangle\}_{\mathtt{h}(\phi(a\cdot c_d, y))}
$$

Since it is sufficient to consider normalisations when proving static equivalence, we present the normalisations $\sigma\downarrow^{\vec{x}}_E$ and $\theta\downarrow^{\vec{x}}_E$ of $\nu\vec{x}.\,(\sigma \mid \_)$ and $\nu\vec{x}.\,(\theta \mid \_)$ respectively for a fixed partitions $\{\alpha, \beta, \gamma, \delta\}$, $\{\zeta^1, \ldots, \zeta^D\}$ of the set of all sessions $\{1, \ldots, L\}$ of $\nu\vec{x}.\,(\sigma \mid \_)$ and $\nu\vec{x}.\,(\theta \mid \_)$ with respect to E below.

$$
\begin{array}{l}
pk_s\sigma = \mathtt{pk}(s) \\
u_l\sigma = ch_l \text{ if } l \in \{1, \ldots, L\} \\
v_l\sigma = \phi(a_l \cdot c_l, \mathfrak{g}) \text{ if } l \in \beta \cup \gamma \cup \delta \\
\text{if } l \in \delta \text{ and } Y_l = \phi(T_l, \mathfrak{g}) \\
\mathtt{fst}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l))))\,\sigma = \phi(a_l \cdot c_l, \mathfrak{g}) \\
\mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l))))\,\sigma = \phi(a_l \cdot c_l, \mathtt{sig}(\mathfrak{g}, s)) \\
\mathtt{check}(\mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l)))), pk_s)\,\sigma = \phi(a_l \cdot c_l, \mathfrak{g}) \\
\text{if } l \in \delta \text{ and } Y_l \neq \phi(T_l, \mathfrak{g}) \\
w_l\sigma = m^l(a_l, Y_l\sigma)
\end{array}
$$

$$
\begin{array}{l}
pk_s\theta = \mathtt{pk}(s) \\
u_l\theta = ch_l \text{ if } l \in \{1, \ldots, L\} \\
v_l\theta = \phi(a_l \cdot c_d, \mathfrak{g}) \text{ if } l \in \zeta^d \cap (\beta \cup \gamma \cap \delta) \\
\text{if } l \in \zeta^d \cap \delta \text{ and } Y_l = \phi(T_l, \mathfrak{g}) \\
\mathtt{fst}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l))))\,\theta = \phi(a_l \cdot c_d, \mathfrak{g}) \\
\mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l))))\,\theta = \phi(a_l \cdot c_d, \mathtt{sig}(\mathfrak{g}, s)) \\
\mathtt{check}(\mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l)))), pk_s)\,\theta = \phi(a_l \cdot c_d, \mathfrak{g}) \\
\text{if } l \in \zeta^d \cap \delta \text{ and } Y_l \neq \phi(T_l, \mathfrak{g}) \\
w_l\theta = m^d(a_l, Y_l\theta)
\end{array}
$$

Notice that a message term in the range of the normalisation can have more then one recipe, e.g. there are three ways to obtain $\phi(a_l \cdot c_l, \mathfrak{g})$. We then proceed by induction on the structure of the weak normal form of $N\sigma$ considering all messages

allowed by the grammar in Fig. 3.12. We present proofs starting from the equation under the frame σ. The argument for the converse case is the same. From now on M, M_k, N, N_k are always fresh for $\vec{x}$.

Case 1. $N\sigma =_E \mathfrak{g}$.

Case 1.1. $N = \mathfrak{g}$. If M is a recipe for $\mathfrak{g}$, then $M = \mathfrak{g}$, since there is no non-trivial recipe for $\mathfrak{g}$ under the normalisation of σ. Then we have $\mathfrak{g}\sigma =_E \mathfrak{g}\sigma$ if and only if $\mathfrak{g}\theta =_E \mathfrak{g}\theta$ as required.

Case 1.2. $N \neq \mathfrak{g}$. There is nothing to prove in this case, since there is no non-trivial recipe for $\mathfrak{g}$ under the normalisation of σ.

Case 2. $N\sigma =_E z$, z is a variable.

Case 2.1. $N = z$. If M is a recipe for z, then $M = z$, since there is no non-trivial recipe for z under the normalisation of σ. Then we have $z\sigma =_E z\sigma$ if and only if $z\theta =_E z\theta$ as required.

Case 2.2. $N\sigma =_E ch_l$. Since N is fresh for $\vec{x}$, $N = u_l$. There is unique recipe $M = u_l$ for ch_l and we have $u_l\sigma =_E u_l\sigma$ if and only if $u_l\theta =_E u_l\theta$ as required.

Case 3. $N\sigma = K_1 \cdot K_2$.

Any message term in the normalisation of σ is m-atomic, hence no message is an immediate m-factor of another message in the normalisation of σ. Therefore there is only one case to consider.

Case 3.1. $N = N_1^{j_1} \cdot \ldots \cdot N_k^{j_k}$, that is $N\sigma$ is generated by m-factors which have a recipe under the normalisation of σ: $N\sigma = N_1^{j_1}\sigma \cdot \ldots \cdot N_k^{j_k}\sigma$. By the induction hypothesis suppose that for all recipes M_i for an m-factor $N_i\sigma$ of $N\sigma$, we have $M_i\sigma =_E N_i\sigma$ if and only if $M_i\theta =_E N_i\theta$, $i \in \{1, \ldots, k\}$. By applying multiplication, we have $M_1^{j_1}\theta \cdot \ldots \cdot M_k^{j_k}\theta = (M_1^{j_1} \cdot \ldots \cdot M_k^{j_k})\theta =_E (N_1^{j_1} \cdot \ldots \cdot N_k^{j_k})\theta = N_1^{j_1}\theta \cdot \ldots \cdot N_k^{j_k}\theta$ as required, and $N_i\theta$ is an m-factor of $N\theta$.

Case 4. $N\sigma = \phi(K_1, K_2)$.

Let us define

$$
\begin{aligned}
&V_1 \coloneqq v_l,\ V_2 \coloneqq \mathtt{fst}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l)))),\\
&V_3 \coloneqq \mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l))))\\
&V_4 \coloneqq \mathtt{check}(\mathtt{snd}(\mathtt{dec}(w_l, \mathtt{h}(\phi(T_l, v_l)))), pk_s)
\end{aligned}
$$

Case 4.1. $N\sigma = \phi(a_l \cdot c_l, \mathfrak{g})$ and $Y_l = \phi(T_l, \mathfrak{g})$. Since N is fresh for $\vec{x}$, $N \in \{V_1, V_2, V_4\}$. Let M be a recipe for $\phi(a_l \cdot c_l, \mathfrak{g})$, then $M \in \{V_1, V_2, V_4\}$ and we have $M\sigma =_E N\sigma$ if and only if $M\theta =_E N\theta$ for any N and M as required. In case $Y_l \neq \phi(T_l, \mathfrak{g})$, $N = V_1$, there is unique recipe $M_1 = V_1$ and the argument is the same.

Case 4.2. $N\sigma = \phi(a_l \cdot c_l, \mathtt{sig}(\mathfrak{g}, s))$ and $Y_l = \phi(T_l, \mathfrak{g})$. Since N is fresh for $\vec{x}$, $N = V_3$ and there is unique recipe $M = V_3$ for $\phi(a_l \cdot c_l, \mathfrak{g})$, and we have $V_3\sigma =_E V_3\sigma$ if and only if $V_3\theta =_E V_3\theta$ as required. If $Y_l \neq \phi(T_l, \mathfrak{g})$, there is no recipe for $\phi(a_l \cdot c_l, \mathtt{sig}(\mathfrak{g}, s))$ and there is nothing to prove.

Case 4.3. $N = \phi(N_1, N_2)$, $N_2 \in \{V_1, V_2, V_3, V_4\}$ and $Y_l = \phi(T_l, \mathfrak{g})$. By the induction hypothesis suppose that for all recipes M_1 for $N_1\sigma$, we have $M_1\sigma =_E N_1\sigma$ if and only if $M_1\theta =_E N_1\theta$, then multiply N_2 by a scalar M_1 and obtain $\phi(M_1\theta, N_2\theta) = \phi(M_1, N_2)\,\theta =_E \phi(N_1, N_2)\,\theta = \phi(N_1\theta, N_2\theta)$ for any N_2 as required. In case $Y_l \neq \phi(T_l, \mathfrak{g})$, $N_2 = V_1$ and the argument is the same.

Case 4.4. $N = \mathtt{sig}(\ldots \mathtt{sig}(N_1, N_2) \ldots, N_k)$, $N_1 \in \{V_1, V_2, V_3, V_4\}$ and $Y_l = \phi(T_l, \mathfrak{g})$. By the induction hypothesis suppose that for all recipes M_i for $N_i\sigma$ we have $M_i\sigma =_E N_i\sigma$ if and only if $M_i\theta =_E N_i\theta$, $i \in \{2, \ldots, k\}$. By applying the signature operation to N_1, we have

$$\begin{aligned}
&\mathtt{sig}(\ldots \mathtt{sig}(N_1, M_2) \ldots, M_k)\theta = \\
&\mathtt{sig}(\ldots \mathtt{sig}(N_1\theta, M_2\theta) \ldots, M_k\theta) =_E \\
&\mathtt{sig}(\ldots \mathtt{sig}(N_1\theta, N_2\theta) \ldots, N_k\theta) = \\
&\mathtt{sig}(\ldots \mathtt{sig}(N_1, N_2) \ldots, N_k)\theta
\end{aligned}$$

as required. In case $Y_l \neq \phi(T_l, \mathfrak{g})$, $N_1 = V_1$ and the argument is the same.

Case 4.5. $N = \phi(N_1, N_2)$. Similar to case 3.1 with $j_i = 1, i \in \{1, 2\}$.

Case 5. $N\sigma = \langle K_1, K_2 \rangle$.

Since no pair is contained in the normalisation of σ, there is only one case to consider.

Case 5.1. $N = \langle N_1, N_2 \rangle$. Similar to case 4.5.

Case 6. $N\sigma = \mathtt{h}(K_1)$. Similar to case 5 with $j_i = 1, i = 1$.

Case 7. $N\sigma = \mathtt{pk}(K_1)$.

Case 7.1. $N\sigma = \mathtt{pk}(s)$. Then $N = pk_s$, since N is fresh for $\vec{x}$. There is a unique recipe $M = pk_s$ for $\mathtt{pk}(s)$ and we have $pk_s\sigma =_E pk_s\sigma$ if and only if $pk_s\theta =_E pk_s\theta$ as required.

Case 7.2. $N = \mathtt{pk}(N_1)$. Similar to case 6.

Case 8. $N\sigma = \mathtt{sig}(K_1, K_2)$. Similar to case 5.

Case 9. $N\sigma = \{K_1\}_{K_2}$.

If $Y_l = \phi(T_l, \mathfrak{g})$ no encrypted message term is contained in the normalised frame and there is only one case to consider.

Case 9.1. $N = \{N_1\}_{N_2}$. Similar to case 5.

If $Y_l \neq \phi(T_l, \mathfrak{g})$, there is also the following.

Case 9.2. $N\sigma =_E m^l(a_l, Y_l\sigma)$. Since N is fresh for $\vec{x}$, $N = w_l$. There is unique recipe $M = w_l$ for $m^l(a_l, Y_l\sigma)$ and we have $w_l\sigma = w_l\sigma$ if and only if $w_l\theta = w_l\theta$ as required.

□

Lemma 1 concludes the proof of the unlinkability of UBDH. To conclude this section we would like to highlight few crucial moments of our method of verifying the unlinkability of the UBDH protocol.

- We have identified that the key agreement proposed by EMVCo exactly matches the AU-authentication situation described is Sec. 2.2.4. Hence in the definition of unlinkability we can consider only one party, and be sure (by the compositional approach enabled by the quasi-open bisimilarity) that the unlinkability would be preserved in the full system.

- The proof consists of identifying a candidate relation for quasi-open bisimulation between two worlds described in Def. 9, and verifying that such candidate satisfies the definition of quasi-open bisimilarity Def. 5. The ingenious step here is to define the candidate relation, for which we followed the pattern

that has already appeared in the private server example in Fig. 2.13, i.e. we parametrise the paired states by the partition of the set of started sessions.

- While the matching transitions and the openness is straightforward to verify, the static equivalence requires more effort. The key step here is to define the normalisations of the frames associated to the related states that captures all possible ways to simplify messages, thereby considering all recipes for the messages that cannot be simplified further.

In the next chapter we will apply our method verify the unlinkability of the full scale payment protocol. However, firstly we should complete the discussion of the EMVCo key establishment proposal and verify that the twofold aim of the initial BDH protocol – to guarantee unlinkability of the card, while allowing the terminal to authenticate the card, is not broken.

3.5.3 Authentication in BDH and UBDH

In this section we will verify, using the ProVerif tool, that both BDH and UBDH protocols satisfy the target authentication property – injective agreement [Low97]. The process scheme below specifies the behaviour of honest terminals and honest cards.

$$SYS \triangleq \nu s.\Big(!\nu c.!\nu ch_c.\overline{card}\langle ch_c\rangle.C(s,c,ch_c) \mid \overline{out}\langle \mathtt{pk}(s)\rangle.!\nu ch_t.\overline{term}\langle ch_t\rangle.T(\mathtt{pk}(s),ch_t)\Big)$$

The attacker model we use for verification of is a standard Dolev-Yao attacker [DY83] who controls the communications between the card and the terminal. Such attackers can intercept, block, modify, and inject messages. In the presence of contactless payments the Dolev-Yao attacker is particularly relevant since, within a range close enough, an attacker can force the card to run the protocol, explaining why we insist on this attacker model when verifying key establishment protocols in this chapter and the full transaction protocol in the next.

In the process scheme above, the processes C and T can be instantiated with C_{rfc} and T_{rfc} or with C_{upd} and T_{upd} to obtain SYS_{rfc} and SYS_{upd}, respectively. The injective agreement property is standard [Low97, CM12]. *Agreement* here means that when a terminal thinks it has authenticated a card, an honest card really executed the protocol while exchanging the same messages as the terminal. *Injectivity* strengthens agreement by ensuring that every successfully authenticating run of a terminal corresponds to a separate run of a card.

The formal model specifying the injective agreement relies on events recorded during the execution of the protocol. An event is declared in the specification using a construct of the form $\mathtt{ev{:}EVT}\,(t_1,\ldots,t_n)$, where $\mathtt{EVT}$ is the name of the corresponding event, and $t_1,\ldots,t_n$ are the corresponding parameters. We rely on the following two events to be included at relevant points in the specification of BDH and UBDH.

- $\mathtt{TCommit}(cbkey, tkey, ecert)$: the terminal commits to the session with the card after receiving the message *cbkey*, sending its own message *tkey*, and receiving (and checking) the encrypted certificate *ecert*, the third message, meaning that

it is convinced that the card is involved in the corresponding key establishment session.

- $\texttt{CRunning}(cbkey, tkey, ecert)$: the card is running a BDH/UBDH session after sending its own message *cbkey*, receiving *tkey* and generating the encrypted certificate *ecert*.

To specify the injective agreement of BDH and UBDH formally in ProVerif we rely on a restricted class of corresponding assertions, whose syntax and semantics we introduce informally in the following, referring to e.g. [Bla09] for more details. A (simplified) correspondence assertion is a formula of the form $\Phi_0 \implies \Phi_1$, where Φ_0 is a conjunction of events, and Φ_1 is a conjunction of disjunctions of events. A protocol specification satisfies such a formula if, for any execution trace of the protocol where the events in Φ_0 are true, it is the case that one of the events in each conjunct of Φ_1 is true. The variables of Φ_0 have an implicit universal quantifier, while the variables in Φ_1 (that are not in Φ_0) are quantified existentially. Using the events introduced above, authentication in both BDH and UBDH is specified by the correspondence assertion presented in Fig. 3.30. Notice that the list of parameters repeating on both sides indicates the card exchanging the same messages as the terminal.

$$\texttt{TCommit}(cbkey, tkey, ecert) \Rightarrow \texttt{CRunning}(cbkey, tkey, ecert)$$

Figure 3.30: The correspondence assertion for authentication in BDH/UBDH.

The message theory that faithfully models the cryptographic primitives used in BDH and UBDH is currently outside the scope of ProVerif. There are two sources of complexity in the theory. One is scalar multiplication of an unbounded number of terms with an element of the elliptic curve group, as modelled by the first three equations in Fig. 3.13. We handle this problem by extending a standard abstraction used in ProVerif to model the Diffie-Hellman based key agreement. Instead of the first three equations from Fig. 3.13, we have the following two equations:

$$\begin{aligned} \phi(M, \phi(N, \mathfrak{g})) &=_E \phi(M, \phi(N, \mathfrak{g})) \\ \phi(L, \phi(M, \phi(N, \mathfrak{g}))) &=_E \phi(M, \phi(L, \phi(N, \mathfrak{g}))) \end{aligned}$$

which together cover all permutations of three scalar multipliers on top of the group generator $\mathfrak{g}$. They are sufficient for modelling the blinded Diffie-Hellman key agreement as used in our protocol: the three scalars are the secret key of the card, the blinding factor chosen by the card, and the scalar chosen by the terminal in the first message to the card. A second source of complexity is the equation in Fig. 3.16, which allows us to homomorphically push scalar multiplication inside a signature. We handle this second problem by replacing the homomorphic equation $\phi(M, \texttt{sig}(N, K)) =_E \texttt{sig}(\phi(M, N), K)$ with the equation $\texttt{check}(\phi(N, \texttt{sig}(M, K)), \texttt{pk}(K)) =_E \phi(N, M)$, which allows us to verify a multiplied signature directly, without pushing the multiplication inside the signature.

It takes less then 1 minute for ProVerif to verify that our target functional property holds [see the repository [repa] for the code], i.e. we can establish that both SYS_{rfc} and SYS_{upd} satisfy injective agreement. The immediate consequence of this is that indeed the card and the terminal share the same key since they share the same messages.

3.6 Highlighting what we have learned

The first, high-level, take away message from the this chapter is the characterisation of EMV [emv11] as a protocol with questionable security and non-existent privacy. As we demonstrate in Sec. 3.2 to this day insecure configurations are the part of the EMV standard [emv11], i.e. it is possible to come up with a version of the EMV protocol that does not contradict the standard, but is vulnerable to PIN bypassing attack. As we demonstrate in Sec. 3.1, the data allowing to identify the card, e.g. the card number, is exposed to the environment at each transaction.

The second, core, take away message from the chapter is related to the threat model in the EMVCo proposal of secure channel establishment [rfc12]. The anti-eavesdropping requirement in the BDH and 2nd Gen specifications [rfc12, ove14] form a reasonable privacy enhancement guided by the current state of the EMV standard and the infrastructure already deployed. We, however, have taken the liberty to look beyond passive eavesdropping and considered the implications of realistic active attackers, as captured in Def. 9. We support this investigation by observing that, the anti-tracking requirement explicitly mentioned in the BDH proposal remains open to interpretation, specifically [rfc12], "The protocol is designed to protect against eavesdropping and card tracking." We conclude that unlinkable EMV key agreement, under such assumptions, is feasible as we prove formally in Theorem 6. Yet, unlinkability in the presence of an active attacker is difficult to extend to the full EMV 1st Gen due to EMVCo never has made precise how BDH and the rest of the current EMV protocol coexist. Given the overview of the EMV standard in Sec. 3.1 it is straightforward that the full protocol would not satisfy unlinkability if the existing communication would be simply encrypted, regardless of the key establishment employed. Since the PAN should always be transferred to the terminal in EMV, an active attacker can establish the symmetric key with the card, e.g., using UBDH, and obtain the same PAN in two different sessions, thereby linking this sessions repeating an attack scheme explained in Sec. 3.3.2, 3.3.3.

The third, more general, take away message concerns our method for verifying properties defined in terms of a process equivalence problem. Our unlinkability definition, Def. 9, based on the notion of quasi-open bisimilarity, which is a congruence, enables compositional reasoning about protocols, such as UBDH, without a shared key. Our approach to bisimilarity checking facilitates proving that the property holds for the unbound number of sessions, where the main challenge is to define the relation in Fig. 3.18, after which we apply the method to show that the relation is a quasi-open bisimulation. Furthermore, the equational theory we employ is not yet covered by equivalence checking tools, so our proof proof may help inform the extension of tools to this class of problems.

Chapter 4

How to design an unlinkable smartcard-based payment protocol.

In the previous chapter, we have shown that the ubiquitous smart card-based payment method, EMV [emv11], lacks any privacy, as the valuable information such as transaction details and the card number is sent in cleartext, enabling the profiling and tracking of cardholders. Such lack of privacy in EMV should not be viewed as the fundamental flaw the developers of the standard have overlooked – the privacy of payments has never been among the explicit requirements of EMV. In this chapter, we consider adding a privacy requirement, namely, unlinkability, to the initial EMV requirements and investigate what it takes to design an unlinkable smartcard-based payment protocol in the presence of a realistic active attacker without compromising essential security guarantees and functional properties of EMV.

Our position is that unwanted data collection should be mitigated at the protocol level since legal sanctions are not enough to ensure privacy – we have examples of their violation [ger22, rsf21]. EMVCo has already made a step towards enhancing privacy in EMV and has proposed to encrypt communications by employing the BDH protocol, an authenticated key establishment protocol [rfc12] that protects from eavesdropping. However, even in the presence of encryption, the problem of active attackers persists. As discussed in the previous chapter, the card still sends its signed public key, a strong form of identity, to the terminal allowing the attacker to trace the card. The threat of active attackers pretending to be honest terminals and capable of starting communicating with the card threat is real, as attackers may use unauthorised devices like smartphones or custom-made antennas [HDPdR15] to secretly power up the card, run the protocol session and thereby track the cardholder using the data transferred. To rule out such active attackers while maintaining the initial EMV privacy goals [rfc12], we have proposed the UBDH protocol, an *unlinkable* version of the BDH protocol, where an attacker cannot link key establishment sessions with the same card. The essence of UBDH is that the public key of the card appears to be fresh in each session, yet a terminal can still authenticate that the card was issued by a recognised payment system.

According to the proposal of EMVco [rfc12], a transaction in the 2nd generation EMV would consist of a key establishment phase followed by data exchange. Hence we need to consider the unlinkability of the full protocol, as an active attacker in the second phase could gather the information allowing to link payment sessions even if the first phase, key agreement, is unlinkable. If we simply follow what EMV

offers now, this information includes the card's explicit identity PAN used by the payment system to route payments through the network, hence neither BDH nor UBDH would prevent an attacker to silently obtain the PAN in each session and track the cardholder.

In what follows, we identify the requirements for an unlinkable payment protocol, discuss how exactly the information that enables linking payment sessions with the same card can be hidden from the terminal, and present a protocol that we prove to satisfy these privacy requirements while preserving key requirements of EMV. The protocol we introduce respects the current flow of EMV discussed in Sec. 3.1 and its infrastructure in Fig. 3.1, supports contactless online transactions, as nowadays they prevail [for21], as well as offline transactions requiring a PIN.

We begin by presenting a design space where we determine the requirements of an unlinkable payment protocol in Sec. 4.1, and draw attention to trade-offs between functional and privacy requirements. We then present an unlinkable payment protocol UTX in Sec. 4.2, and provide formal analysis in Sec. 4.3.

4.1 Design space for unlinkable transactions

In this section, we explore the design space for a privacy-preserving payment protocol. This top-level design space is narrowed down later to guide the design of our proposed protocol. In what follows, we assume the architecture of a payment system explained in the introduction to Chapter 3 and emphasise the functional, security and privacy requirements.

An unlinkable protocol should satisfy three types of requirements: functional, security and privacy requirements. Functional and security requirements are extracted from the current EMV specification [emv11] explained briefly in the previous chapter. Some security requirements are strengthened, and privacy requirements that were not previously present in EMV are introduced.

4.1.1 Functional requirements.

We consider smart card-based payments, hence we rely only on the computational resources of the smart card and the terminal. Devices like smartphones that can establish direct communication between the card and the bank are excluded from the discussion in this paper. We also prohibit indirect card-bank communication by means of, e.g. synchronised clocks since the card has no long-term power source.

The card should use ECC-based cryptography, as already required for the new iteration of the EMV standard [ove14] and in the Blinded Diffie-Hellman proposal [rfc12] we have discussed. Since, currently, the card must be present within the reader's field for at most 500ms [emv21, Section 10], computationally-heavy general-purpose zero-knowledge proofs are out of scope.

The protocol should support contact and contactless transactions. For the purpose of this analysis, we consider the PIN as the only cardholder verification method, and the PIN is always required for high-value transactions. Hardware solutions that might help to replace the PIN, e.g. cards with built-in fingerprint sensors [biob, bioa] are beyond our scope.

Cards can optionally support offline transactions, which carry two risks resulting in the terminal not being paid (when the cryptogram AC is finally processed by the bank): either there is not enough money in the cardholder's account, or the card is blocked, e.g. reported as stolen. If offline transactions are supported, the insurance policy must cover these risks.

4.1.2 Security requirements.

Recall that some configurations of EMV have been shown to be insecure [BST21b]. The primary security goals we extract from good configurations of EMV are the following authentication and secrecy properties.

- T must be sure that the presented card is a legitimate card that was issued by the *PaySys* that T supports and that C is not expired.
- If B accepts the transaction, then T, C, and B must agree on the transaction.
- Keys for message authentication and PIN are secret.

Notice that the card does not authenticate the terminal. The reason is, in the philosophy of the EMV standard, that the payment system allows anyone to manufacture terminals. We strengthen these requirements by assuring the card that if the cryptogram is processed, then it is processed by a legitimate bank.

In addition to the requirements extracted from EMV above, we introduce the additional requirement that the application cryptogram AC must be secret. This is in line with the proposal of secret channel establishment [rfc12], where a session-specific secret channel is used to protect all messages between the card and the terminal from eavesdroppers. As we have said, currently, the communication between the card and the terminal is in cleartext, and the AC, which contains transaction details, is always exposed. Formal security definitions reflecting these requirements are introduced in Section 4.3.6, where we present the analysis of our proposal for a protocol.

4.1.3 Privacy requirements.

As thoroughly explained in the previous chapter, currently, no privacy properties are preserved by EMV. The privacy property we aim for is unlinkability in a sense expressed in Fig. 2.1 at the beginning of this dissertation, i.e. we will call transactions unlinkable if an attacker cannot distinguish between a system where a card can participate in multiple transactions and another system where a card can participate in at most one transaction. In the discussion in Sec. 3.4 comparing anonymity and unlinkability, we have seen that neither of the notions is strictly stronger. However, unlinkability is strictly stronger than per-session anonymity, meaning that if in two sessions the identity (e.g. the card number) is exposed, then they can be linked, but the converse does not hold. This explains why we have chosen unlinkability as a benchmark for privacy.

We will define unlinkability formally in Sec. 4.3.3, where we prove that the protocol we introduce in later sections satisfies unlinkability, however, let us reiterate here as well the informal meaning of unlinkability.

A real-world scenario where the card is issued and within its lifespan can participate in several protocol sessions, and an idealised scenario, where cards are disposed of after each transaction and can participate in one payment session at most (hence sessions are trivially unlinkable) should be indistinguishable to an attacker for a given payment protocol. Guaranteeing this property without compromising the aforementioned security and privacy requirements is our primary challenge.

We explain that unlinkability cannot hold in all contexts if we aim to fulfil also our functional and security requirements. As mentioned above, two sessions that are not per-session anonymous can be linked. Therefore, to achieve unlinkability, certainly, any identity unique either to the card or the cardholder must never be revealed to an attacker. We call such identities *strong*, and they include the cardholder's name, the PAN, the card's public key, and any signature on the data specific to the card.

On the other hand, even if strong identities were protected, *coarse* identities that are common to a group of cards may enable tracking of groups of cardholders. Coarse identities include the payment system, the validity date, the format of transaction data, and other implementation-specific features. Some coarse identities are inevitably exposed as a consequence of the requirements in Sec. 4.1.1, 4.1.2. For instance, the terminal needs to know which payment system the card uses to authenticate the card and needs to be able to distinguish between valid and expired cards. In certain contexts coarse identity may even play the role of the strong one, e.g. if the card exposes the application identifier (as it does at the initialisation phase of EMV) in a country where such identifier is not used, this card is linkable with a high probability in that country, since cards around do not carry such application identifier. Other coarse identities include the network traffic response times, which may reveal information about whether the card belongs to a local or foreign bank.

Coarse identities can be combined to *fingerprint* a card. Thus we are obliged to accept that unlinkability can only be achieved up to their fingerprint, that is, we can link two sessions with the same fingerprint only. However, we require that this fingerprint is minimised, thereby limiting the capability of an attacker to perform unauthorised profiling of cardholders and their behaviours.

4.2 The UTX protocol

In this section, we introduce the UTX (Unlinkable Transactions) protocol that satisfies the security and privacy requirements introduced in Sec. 4.1. We pay particular attention to minimising the fingerprint given by the coarse identities, thereby maximising unlinkability. We start by discussing the initialisation phase, then we introduce the message theory representing cryptographic primitives employed in the protocol. We then explain the key distribution between the participants. Finally, we thoroughly explain transactions that can either be offline, online, high, or low-value.

4.2.1 Application selection

Recall from the description of the Initialisation phase of EMV in Sec. 3.1.1 that the card can generally support several payment methods, colloquially called *applications*. We show schematically in Fig. 4.1 the shortened version of the *Initialisation* phase regarding the application selection. First, the terminal asks the card to send the list of supported applications, then the card provides the list and the terminal selects one (possibly with the help of the cardholder). Knowing the payment system, the terminal can select the appropriate public key to authenticate the data on the card. Notice that the list of payment applications is a coarse identity of the card even if this list consists of a single application since it can still be distinguished from other cards.

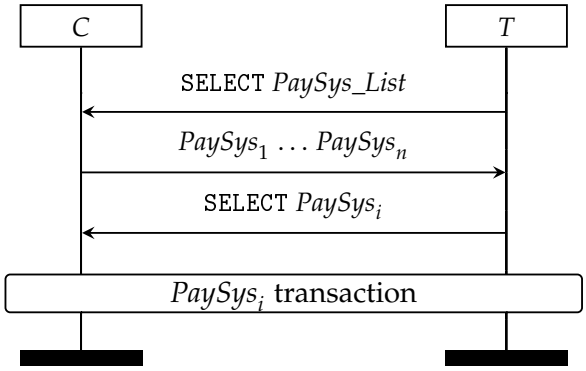


Figure 4.1: Payment System Selection.

In order to avoid a coarse identity being exposed at this point, we design the protocol such that the card presents a list comprising a single element, Unlinkable. This means that a group of payment systems agree to provide privacy-preserving payments using the name Unlinkable for the respective application. Terminals, thus, should also be upgraded to support Unlinkable in order to accept unlinkable payments before such cards are rolled out. An alternative is to allow each payment system to provide its own unlinkable application and to tolerate that the payment system becomes part of the coarse identity of the card. Our analysis covers both choices.

4.2.2 Keys required to set up Unlinkable

Here we explain who generates and holds keys and signatures involved in the UTX protocol. An authority, who is either a payment system or a delegate acting on behalf of a group of payment systems, produces signatures involved in the protocol using two types of signing keys. Firstly, a secret key s, is used to produce *certificates for banks*, which are kept by the terminal and used by the card to check that the terminal is connected to a legitimate bank. Secondly, a list of secret keys $\chi_{\texttt{MM}}$ is maintained for each new calendar month. They are used by the authority upon request from the payment system to generate *month certificates* unique to each card supporting Unlinkable for every month the card is valid. A card valid for five years

would store 61 such month certificates, that the terminal checks to be sure that the card is valid at the month of a particular purchase. The public key for checking month certificates is broadcast to terminals from the first of every month.

We take care to prohibit an attacker from learning the expiry or the issuing month, which would allow many cards to be distinguished. To do so, we introduce the following pointer mechanism. The card maintains a pointer to the most recent month certificate that has been used in response to a legitimate request by the terminal. When the terminal asks the card to show the certificate for the month, the card compares the pointer with the received month. If the received month is greater than what the pointer references, the card advances the pointer to this month and shows the respective certificate. If either the received month coincides with or is one month behind the pointer, the card simply shows the certificate for this month, and the pointer remains untouched. Otherwise, if the month requested is older than two months, the card terminates the session. A terminal cannot request a month in the future, assuming that the public keys for verification are carefully managed, such that they are never released in advance.

We allow a window of two months to allow time for offline terminals to eventually receive the most recent public key for the month. For this reason, a new card valid for 60 months is loaded with 61 month certificates with a pointer referencing the issuing month. That way, a newly issued card cannot be distinguished from cards already in circulation as it is ready to present the certificate for the month prior to the month in which it was issued. Thus, the only coarse identities revealed are whether the card is outdated or has not been used since the beginning of the month.

4.2.3 Message theory

We now introduce cryptographic primitives employed by the UTX protocol. The message theory we use to define UTX differs from the one we have used to define UBDH by just one primitive – instead of using Verheul signatures exclusively, we now also include a generic signature scheme. As usual for symbolic analysis that we conduct later in Sec. 4.3, we assume perfect cryptography and omit low-level details such as ECC domain parameters. Fig. 4.2 presents the message theory for the UTX protocol that consists of the syntax that defines messages agents can form and the equational theory E, that axiomatises cryptographic operations.

Our message theory admits operations for ECC-based cryptography, i.e. multiplication between two field elements (scalars) and multiplication between a scalar and an element of the DH group. As before, when we say that "a message is blinded with a scalar," we mean multiplication by that scalar. Next, we include a standard set of cryptographic operations such as hashing, symmetric key cryptography, n-tuples, and generic digital signatures. Finally, we introduce the already familiar Verheul signature scheme [Ver01], which is invariant under blinding of the message-signature pair with the same scalar (hence can appear as "new" in each session). We also define several constants employed in UTX.

The equational theory E captures the two types of multiplication and contains conventional destructor functions: decryption, projection, and two versions of signature verification. A digital signature is successfully verified whenever the

$M, N ::= \mathfrak{g}$	DH group generator (constant)
$\mid x$	variable
$\mid M \cdot N$	multiplication
$\mid \phi(M, N)$	scalar multiplication
$\mid \mathtt{h}(M)$	hash
$\mid \{M\}_N$	symmetric encryption
$\mid \langle M_1, \ldots, M_k \rangle$	n-tuple
$\mid \mathtt{pk}(M)$	public key
$\mid \mathtt{sig}(M, N)$	signature
$\mid \mathtt{vpk}(M)$	Verheul public key
$\mid \mathtt{vsig}(M, N)$	Verheul signature
$\mid \mathtt{check}(M, N)$	check signature
$\mid \mathtt{vcheck}(M, N)$	check Verheul signature
$\mid \mathtt{p_i}(N)$	ith projection
$\mid \mathtt{dec}(M, N)$	symmetric decryption
$\mid \bot$	empty messsage
$\mid \mathtt{ok}$	PIN success reply
$\mid \mathtt{accept}$	bank success reply
$\mid \mathtt{auth}$	transaction authorisation
$\mid \mathtt{lo},\ \mathtt{hi}$	low and high amount

$$
\begin{aligned}
&M \cdot N =_E N \cdot M \\
&(M \cdot N) \cdot K =_E M \cdot (N \cdot K) \\
&\phi(M \cdot N, K) =_E \phi(M, \phi(N, K)) \\
&\mathtt{p_i}(\langle M_1, \ldots, M_k \rangle) =_E M_i \\
&\mathtt{dec}(\{M\}_K, K) =_E M \\
&\mathtt{check}(\mathtt{sig}(M, K), \mathtt{pk}(K)) =_E M \\
&\mathtt{vcheck}(\mathtt{vsig}(M, K), \mathtt{vpk}(K)) =_E M \\
&\phi(M, \mathtt{vsig}(N, K)) =_E \mathtt{vsig}(\phi(M, N), K)
\end{aligned}
$$

Figure 4.2: UTX message theory.

message corresponds to the message extracted from the signature by applying the appropriate check function. Notice that the last equation ensures that if the function $\mathtt{vcheck}(\_, \_)$ is applied to the signature, blinded with some scalar and the matching Verheul public key, it returns the message, blinded with the same scalar.

4.2.4 Before running the protocol: the setup

Before describing the protocol, we explain how the payment system issues a card in collaboration with the issuing bank, how the acquiring bank joins the payment system, and how the terminal connects to the acquiring bank. In the next section, where we describe the transaction, we collapse the payment system, the issuing bank, and the acquiring bank into a single agent.

Issuing a card. Here we outline how a card could be manufactured involving signing authority that payment systems could share as explained in Sec. 4.2.2.

To issue a card, the payment system generates a new card's private key c, computes the card's public key $\phi(c, \mathfrak{g})$, and asks the signing authority to generate

the following list of month Verheul signatures $\{\langle \mathtt{MM}, \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{MM}})\rangle\}_{\mathtt{MM}=0}^{60}$ which it loads to the card together with $\mathtt{pk}(s)$, c, $\phi(c,\mathfrak{g})$, PAN, and PIN. Then the card is sent to the issuing bank together with $\phi(c,\mathfrak{g})$, PAN, and PIN; the bank generates and loads to the card a new master key *mk*, and finally sends the card to the user. Since no one should ever have access to c except the card, we assume the payment system never shares or stores c.

The keys used by the terminal to connect to the payment system. To allow an acquiring bank supporting the payment system to process payments in the month MM, the authority knowing s issues a certificate $\langle\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle$ to each acquiring bank, where b_t is the private key of the bank. In turn, the acquiring bank loads the terminal with both this data and a symmetric key *kbt* used for secure communication between the terminal and the bank. The terminal presents the bank's certificate at each run of the protocol. As explained in Section 4.2.2, the terminal and the bank must update the month key certificate and the month validation key regularly without being offline for more than two months.

Firstly we explain why the month MM is signed. Recall the card points to the most recent month it has seen. Hence, if this month requested by the terminal is the month pointed to by the card or the month before, it is safe to reveal that it is valid for either of these two months. The signature $\mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)$ containing the month MM is required in the situation where the next month is requested, in which case this signature serves as proof to the card that the next month has arrived. This prevents attackers from learning whether the card is valid next month and also avoids the pointer being advanced too quickly, thereby invalidating the card in the current month. Notice that $\mathtt{vpk}(\chi_{\mathtt{MM}})$ is publicly known for the past few months and could have been transferred by the terminal to the card and used by the card to check whether a request for the next month is valid. However, since checking Verheul signatures is too expensive for the card, we avoid using keys $\mathtt{vpk}(\chi_{\mathtt{MM}})$, and instead only check the certificate $\langle\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle$ against the generic $\mathtt{pk}(s)$ already present in the card which can employ a more efficient signature scheme since it does not need to support blinding.

Secondly, the bank's certificate enables the card to verify that $\phi(b_t,\mathfrak{g})$ is a public key for a legitimate bank connected to the payment system providing Unlinkable, hence it can safely use $\phi(b_t,\mathfrak{g})$ to encrypt the application cryptogram and the related data at the end of the transaction. This signature helps to avoid the situation when an attacker introduces their own public key and thereby can look inside the message encrypted for the bank to gather sensitive information including the PAN.

In principle, the signature on the month and the signature on the bank's public key could be separate, yet it is efficient to transmit them in a single message. Moreover, the bank's public key introduces certain padding to small and publically known constants MM representing months. If a bank requires multiple keys, multiple certificates could be produced.

The secure channel between the bank and the terminal modelled here as a symmetric key *kbt* could be established by other means, which is consistent with EMV as it is not specified.

4.2.5 The UTX transaction

We introduce online and offline modes of the UTX protocol in Fig. 4.3. We annotate offline and online high-value modes, for which the PIN is always asked, as `offl` and `onl` respectively. In the offline mode, the PIN is sent to the card. As the PIN must be transferred to the card, and the card cannot leave the session until the PIN is entered, high-value offline transactions are always performed as a contact payment. In online mode, the PIN is not sent to the card, instead, it is sent to the bank together with the application cryptogram. Parts of the protocol involving the PIN check are indicated by dashed lines and annotated as `offl` and `onl` indicating these two modes of operation. In Fig. 4.3 the two messages exchanged between the terminal and the bank are either executed during the transaction (online mode) or postponed to the moment when the terminal goes online to upload collected cryptograms and, optionally, to update its bank's certificate (offline mode).

Initialisation. When the card is close enough to the terminal, it is powered up, and the terminal asks which payment methods the card supports by issuing the `SELECT` command. The card supporting unlinkable payments replies with a singleton list containing only Unlinkable, as explained in Sec. 4.2.1 above. The terminal then selects this payment method and sends to the card the ephemeral public key $\phi(t, \mathfrak{g})$. The card in response sends to the terminal $\phi(a, \phi(c, \mathfrak{g}))$, which is its public key, blinded with a fresh scalar a. After that, the card and the terminal establish the symmetric session key $k_c := \mathtt{h}(\phi(a \cdot c, \phi(t, \mathfrak{g}))) =_E \mathtt{h}(\phi(t, \phi(a, \phi(c, \mathfrak{g})))) =: k_t$ which they use to encrypt all further communications. We represent that all communications between the card and the terminal are encrypted by putting the box around the messages they exchange.

A *passive eavesdropper* who only listens and intercepts messages is now locked out of the session since it has no access to the derived key. On the other hand, an *active attacker* can choose their own public key and engage in the handshake, so we must explain below how such active attacker is mitigated. The only information about the card exposed at this point is the fact that the card supports the application Unlinkable.

Validity check. After the secret key is established, the card presents evidence that it is valid. To do so, firstly, the terminal sends to the card the current month certificate $\langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle$. The card verifies this certificate against the public key $\mathtt{pk}(s)$, hence believes that this terminal is connected to a legitimate acquiring bank and that `MM`, and $\phi(b_t, \mathfrak{g})$ are authentic.

Having received this legitimate request to show the certificate corresponding to `MM`, the card updates its pointer, leaves it untouched or, aborts the transaction as described in Sec. 4.2.2. After the decision about the pointer has been made, the card blinds the appropriate month Verheul signature $\mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}})$ with the scalar a and sends to the terminal $\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}))\rangle$.

The terminal verifies this blinded message-signature pair against the current month Verheul public key $\mathtt{vpk}(\chi_{\mathtt{MM}})$ and additionally checks that the first element of the received pair coincides with the card's blinded public key used to establish

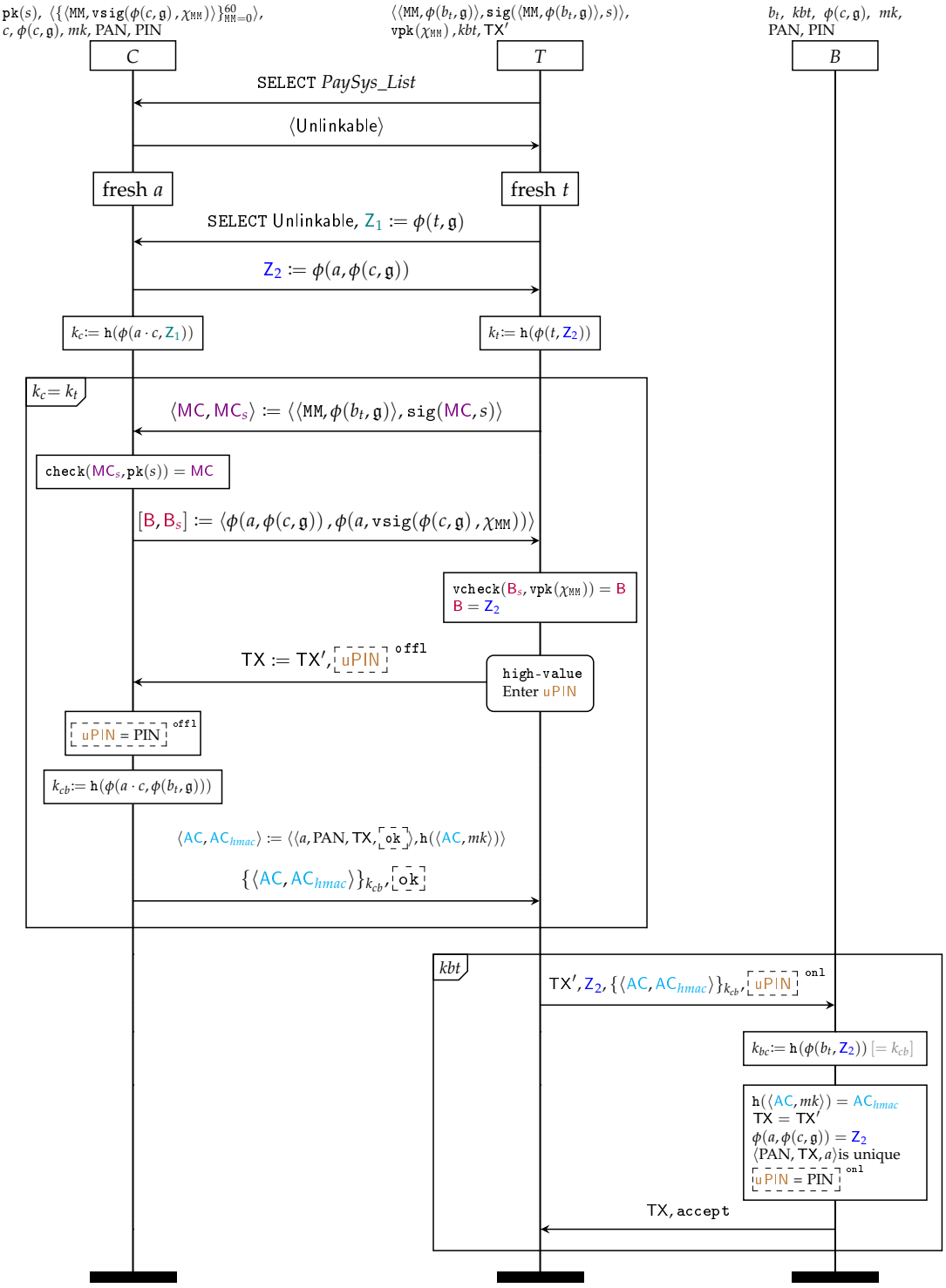


Figure 4.3: The UTX protocol.

a session key. This check ensures that the terminal is still communicating with the same card.

Since both elements of the message coming from the card at this stage are freshly blinded, as for the session key, they are distinct in each session, hence the terminal cannot use it to reidentify the card in future sessions by simply requesting the same month. At this point in the protocol, the card exposes that it is valid at the month $\texttt{MM}$ (since the key $\texttt{vpk}(\chi_{\texttt{MM}})$ fits) which is not a coarse card's identity, as all other cards that have not yet expired and support unlinkable payments, expose the same information.

Cardholder verification (high-value). In case of a high-value *offline* transaction, the terminal asks the cardholder to enter the PIN and sends the entered number $\texttt{uPIN}$ to the card together with the transaction details. If this input matches the actual card's PIN, the card includes the $\texttt{ok}$ message both in the reply to the terminal and in the cryptogram to indicate to the issuing bank that the PIN has been successfully verified on the card's side. Otherwise, the card includes the $\perp$ message in the reply and in the cryptogram, which the terminal has to send to the bank anyway to log failed attempts to enter the PIN for auditing purposes. In case of a high-value *online* transaction, the terminal also asks the cardholder to enter the PIN but instead keeps it and sends it to the acquiring bank together with the cryptogram.

Cryptogram generation. The terminal sends to the card the transaction details TX' comprising the currency, the amount, and the date; and either $\perp$ (when the transaction is low-value), or, the entered $\texttt{uPIN}$ (when the transaction is high-value offline). The card computes $k_{cb} := \texttt{h}(\phi(a \cdot c, \phi(b_t, \mathfrak{g})))$, which serves as a symmetric session key between the card and the acquiring bank for this transaction only. Then the card generates one of the cryptograms.

- $\mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX} \rangle$ if no $\texttt{uPIN}$ has been received.
- $\mathsf{AC}^{\texttt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \texttt{ok} \rangle$ if the received $\texttt{uPIN}$ is correct.
- $\mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp \rangle$ otherwise.

Finally, the card uses the master key mk that has already been shared between the card and the issuing bank to compute hash-based message authentication code of the form $\texttt{h}(\langle AC, mk \rangle)$ and replies, respectively, with one the following messages to the terminal.

- $\{\langle \mathsf{AC}, \texttt{h}(\langle \mathsf{AC}, mk \rangle) \rangle\}_{k_{cb}}$
- $\langle \{\langle \mathsf{AC}^{\texttt{ok}}, \texttt{h}\big(\langle \mathsf{AC}^{\texttt{ok}}, mk \rangle\big) \rangle\}_{k_{cb}}, \texttt{ok} \rangle$
- $\langle \{\langle \mathsf{AC}^{\perp}, \texttt{h}\Big(\langle \mathsf{AC}^{\perp}, mk \rangle\Big) \rangle\}_{k_{cb}}, \perp \rangle$

Each of these messages corresponds to the cryptograms described and contains additional information on whether the PIN was successfully verified by the card ($\texttt{ok}$ entry) or the PIN verification has failed ($\perp$ entry) since the terminal cannot open the cryptogram encrypted for the acquiring bank.

Notice that the card includes a nonce a in each of the cryptograms to make it unique per session. The fact that the same a is used for blinding the card's public key at the initialisation step allows the bank to strongly connect the cryptogram to the current session, thereby avoiding the cryptogram being replayed in other sessions. Although a trusted terminal is already assured that a valid card generated the cryptogram in the current session, it is beneficial for the bank to also check this. This is because the bank may not fully trust the terminal to be implemented correctly, in which case, if the terminal fails to authenticate the card properly as described in the validity check stage above, the terminal cannot be reimbursed for the cryptogram generated by an honest card in another session and replayed in a session with an unauthenticated device posing as a card. Therefore UTX ensures the recent aliveness of the card from the perspective of the bank, even in the presence of compromised terminals.

Message integrity is outside of the symbolic model we use to analyse the UTX protocol. In practice, however, the integrity of this last message from the card to the terminal should be guaranteed. It is especially important for high-value offline transactions, as an attacker with the stolen card may attempt to use that card without knowing the PIN by altering bits corresponding to the `ok` element of the response, therefore tricking the terminal into believing that the PIN was verified successfully. The application cryptogram will later be rejected by the bank, however, an attacker would be gone with the goods. The integrity of the message may be achieved by computing two different keys (thus avoiding potential key-cycles [AR02, Lau02]) instead of one $k_c{=}_E\ k_t$ at the initialisation step as follows. A shared key material is padded with 0, and then hashed to obtain a session symmetric key $k_c^s{:=}\ \mathtt{h}(\langle \phi(a\cdot c, \phi(t,\mathfrak{g})), 0\rangle) =_E \mathtt{h}(\langle \phi(t, \phi(a, \phi(c,\mathfrak{g}))), 0\rangle) =: k_t^s$. The same key material is padded with 1, and then hashed to obtain a key to verify the message integrity $k_c^h{:=}\ \mathtt{h}(\langle \phi(a\cdot c, \phi(t,\mathfrak{g})), 1\rangle) =_E \mathtt{h}(\langle \phi(t, \phi(a, \phi(c,\mathfrak{g}))), 1\rangle) =: k_t^h$. Then the card would respond with $\langle M, \mathtt{h}\Big(\langle M, k_c^h\rangle\Big)\rangle$, where M is one of the three responses above. The terminal could then verify the integrity of the card's response by comparing the hash of the first element of the received message with k_c^h, and the second element of the received message in a similar way the bank verifies the integrity of the cryptogram in Transaction authorisation step we describe next.

Transaction authorisation. In the final stage of the protocol, the terminal asks the bank to authorise the payment. The terminal uses the pre-established secret key *kbt* that is shared with the acquiring bank to send the following.

- The transaction details TX'.
- The blinded card's public key $\mathsf{Z}_2 := \phi(a, \phi(c, \mathfrak{g}))$.
- The encrypted cryptogram of one of the three types described above that it has received from the card.
- The user-entered PIN $\mathtt{uPIN}$ in case the transaction is high-value online, or the message $\perp$ otherwise.

Recall that B in Fig. 4.3 represents both the acquiring and the issuing banks. The acquiring bank uses its private key b_t and the received card's blinded public

key Z_2 to compute the session symmetric key with the card $k_{bc} := \mathtt{h}(\phi(b_t, \mathsf{Z}_2)) = \mathtt{h}(\phi(b_t, \phi(a, \phi(c, \mathfrak{g}))))$ and to decrypt the cryptogram. Internally to B, the acquiring bank uses the PAN from the decrypted cryptogram to forward all the information received from the terminal to the issuing bank. In turn, the issuing bank bank determines mk, $\phi(c, \mathfrak{g})$, and the PIN corresponding to the PAN received and performs the following.

- It checks that the first element of the cryptogram hashed with mk equals the second element, making sure the cryptogram is authentic.
- It checks that the transaction details TX' received from the terminal match the transaction details from the cryptogram: $\mathsf{TX}' = \mathsf{TX}$
- It checks that the blinding factor a from the cryptogram multiplied by the card's public key $\phi(c, \mathfrak{g})$ matches the blinded public key Z_2 received from the terminal: $\phi(a, \phi(c, \mathfrak{g})) = \mathsf{Z}_2$.
- It checks the transaction history of the card and ensures that the received a has not been used for an identical transaction, hence preventing a replay of the cryptogram. This replaces the transaction counter (ATC) mechanism from the EMV standard.
- If the transaction value is high, the bank checks if the `ok` tag is present in the cryptogram and proceeds with the reply, otherwise, if the `ok` tag is not present, the bank checks if the received $\mathtt{uPIN}$ matches the card's PIN: $\mathtt{uPIN} =$ PIN and proceeds with the reply.

If the above is successful, the terminal receives the reply message $\langle \mathsf{TX}, \mathtt{accept} \rangle$ encrypted with kbt.

Notice that in UTX the payment system still uses the PAN to route payments between acquiring and issuing banks, however, it is now hidden from the terminal in contrast to the current EMV standard, where it is exposed. The main changes to the infrastructure to roll out UTX are as follows. The acquiring bank requires a key for decrypting the cryptogram. The issuing bank requires to ensure itself that the nonce from the cryptogram is tied to the legitimate card-terminal session. In addition, a substantial update is needed for public key infrastructure explained in Sec 4.2.2, 4.2.4.

4.3 Unlinkability and security analysis

As in the previous chapter, where we have analysed only the key agreement phase, we will employ quasi-open bisimilarity to define and then verify the unlinkability of UTX, and we will use the ProVerif tool with its corresponding assertions mechanism to define and then verify the security properties of UTX. We focus the analysis on the core component of our protocol, modelling its key agreement and transaction authorisation steps. We omit the application selection step as it involves only constant messages that are the same for all sessions. We also restrict the analysis to the case where all cards are synchronised to execute within the same month `MM`,

leaving the modelling and verification of what happens when there is a transition from one month to the next as future work.

Validity and unlinkability. Exposing the fact that the card is valid within the current month carries two following risks. Firstly, it would be relatively easy to distinguish expired cards from most cards since they will fail to provide a certificate for the current month. However, it is natural to expect that cardholders are issued new cards before they expire, and those expired cards are destroyed, and hence we do not consider this to be a privacy concern. Secondly, cards that are not used every month (such as a card for a savings account) can be distinguished from more regularly used cards since they may be ready to present certificates to an active attacker more than a month out of date. However, we consider such cards to be outliers excluded from the analysis of unlinkability.

4.3.1 Attacker model

It is worth reflecting on the attacker model we assumed in the previous chapter. As usual, our attacker is the implicit environment that interacts with honest participants. What is special is that we assume that cardholders only enter their PIN into honest terminals. In other words, the cardholder uses terminals at reputable points of sale that observe adequate security measures, such as security cameras, i.e. an attacker that secretly interacts with the card without the cardholder's awareness cannot obtain the card's PIN. The properties of unlinkability and PIN secrecy are immediately compromised if the PIN is entered into a malicious terminal which reveals the PIN to attackers. If an attacker possesses a PIN, clearly the card can be stolen and then used for high-value purchases for which the PIN is required. While theft may be mitigated by cancelling cards, an attacker knowing the PIN may authorise high-value purchases via a relay attack, making it difficult for the cardholder to dispute the transaction, as legally a cardholder is always held liable for transactions authorised by a PIN; and hence the primary goal of the security of money in the account would be compromised. Supposing that relay attacks were mitigated, an attacker knowing the PIN may still attack unlinkability as follows. For high-value transactions, it becomes possible for a terminal that remembers the PIN to track cards by the fact that the same PIN is used. Moreover, even in a low-value contact scenario not requiring the PIN, the PIN can nonetheless be used to track specific individuals since such terminals remembering PINs can run a fake session with a high-value amount requiring the PIN to be sent from the terminal to the card in order to check if it has already seen this card before processing the legitimate low-value transaction. In contrast to the above, for low-value contactless payments, unlinkability is preserved even if the PIN is compromised.

Definitely, there are other attacker models we could have verified with respect to. An example of a weaker one, which our analysis covers, is the attacker assumed in the EMVCo proposal for key establishment [rfc12] we have discussed in the previous chapter. An example of a stronger one, which is out of our scope, is an adversary capable of side-channel attacks, e.g. by measuring the execution time of cryptographic operations or the response time from the bank.

4.3.2 Formal specification of the protocol

We have three processes that model the three roles in UTX: the terminal, the card, and the bank. We also have a top-level process in Fig. 4.7 representing the implementation of UTX that expresses how three role processes are assembled and instantiated across multiple payment sessions in full execution of the protocol.

The card. Th process below, represents the execution of a payment session by a card.

$$\nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s, \mathfrak{g}), \mathtt{vsig}_{\mathtt{MM}}, \mathrm{PAN}, mk, \mathrm{PIN})$$

The card's process C, defined in Fig. 4.4, is parametrised by the session channel ch, the card's secret key c, the system-wide public key pks used to check the bank's certificate $\mathtt{crt}$ received from the terminal, the signature $\mathtt{vsig}_{\mathtt{MM}}$ on the card's public key for the current month (considering the currently valid month only simplifies the initial analysis), the card number PAN, and the PIN.

$$\begin{aligned}
&C(ch, c, pk_s, \mathtt{vsig}_{\mathtt{MM}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad ch(z_1).\\
&\quad \nu a.\ \mathtt{let} z_2 := \phi(a, \phi(c, \mathfrak{g}))\ \mathtt{in}\\
&\quad \overline{ch}\langle z_2\rangle.\\
&\quad \mathtt{let}\, k_c := \mathtt{h}(\phi(a \cdot c, z_1))\, \mathtt{in}\\
&\quad ch(m).\\
&\quad \mathtt{let}\, \langle\langle \mathsf{MM}, y_B\rangle, \mathsf{MC}_s\rangle := \mathtt{dec}(k_c, m)\, \mathtt{in}\\
&\quad \mathtt{if}\ \mathtt{check}(\mathsf{MC}_s, pk_s) = \langle \mathsf{MM}, y_B\rangle\ \mathtt{then}\\
&\quad \overline{ch}\langle\{\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}_{\mathtt{MM}})\rangle\}_{k_c}\rangle.\\
&\quad ch(x).\\
&\quad \mathtt{let}\, \langle \mathsf{TX}, \mathtt{uPin}\rangle := \mathtt{dec}(k_c, x)\, \mathtt{in}\\
&\quad \mathtt{let}\, \mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX}\rangle \mathtt{in}\\
&\quad \mathtt{let}\, \mathsf{AC}^{\mathtt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \mathtt{ok}\rangle \mathtt{in}\\
&\quad \mathtt{let}\, \mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp\rangle \mathtt{in}\\
&\quad \mathtt{let}\, k_{cb} := \mathtt{h}(\phi(a \cdot c, y_B))\, \mathtt{in}\\
&\quad \mathtt{if}\ \mathtt{uPin} = \perp\ \mathtt{then}\\
&\qquad \overline{ch}\langle\{\{\langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk\rangle)\rangle\}_{k_{cb}}\}_{k_c}\rangle\\
&\quad \mathtt{else\,if}\ \mathtt{uPin} = \mathrm{PIN}\ \mathtt{then}\\
&\qquad \overline{ch}\langle\{\langle\{\langle \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\langle \mathsf{AC}^{\mathtt{ok}}, mk\rangle)\rangle\}_{k_{cb}}, \mathtt{ok}\rangle\}_{k_c}\rangle\\
&\quad \mathtt{else}\\
&\qquad \overline{ch}\Big\langle\{\langle\{\langle \mathsf{AC}^{\perp}, \mathtt{h}(\langle \mathsf{AC}^{\perp}, mk\rangle)\rangle\}_{k_{cb}}, \perp\rangle\}_{k_c}\Big\rangle
\end{aligned}$$

Figure 4.4: Specifications of the card's role in UTX.

First, the card establishes a key with the terminal, then checks the certificate of the terminal and sends back its own monthly certificate (comprising its public key and the corresponding Verheul signature) blinded with the scalar a used in

the shared key establishment. Using the data provided in the terminal's certificate, the card also generates k_{cb}, which is a fresh symmetric key to be used by the card to communicate securely with the bank (the terminal cannot obtain this key). Upon receiving the transaction details, the card decides as follows: if no PIN has been provided or the corresponding PIN matches its own PIN, the card accepts the transaction and replies with the corresponding cryptogram. Otherwise, the rejection cryptogram $\mathsf{AC}^{\perp}$ is generated and sent as a reply to the terminal.

The terminal. The modes in which a terminal can operate are combined in a role T defined as follows:

$$\nu ch.\overline{term}\langle ch\rangle.T(user, ch, pub_{\mathtt{MM}}, \mathsf{crt}, kbt)$$

where ch identifies the communication channel used in that session, $user$ is a secret channel name used to enter the PIN, $pub_{\mathtt{MM}}$ is the public key used for verifying the card certificate for the given month, and kbt is a shared secret key between the terminal and the bank. To incorporate various operation modes for the terminal, we have three types of processes from which the terminal process T is made of: the process for Online High-Value transactions $T_{\mathtt{onhi}}$, for Offline High-Value transactions $T_{\mathtt{onhi}}$, and for Low-Value transactions $T_{\mathtt{lo}}$, i.e. T is defined as $T_{\mathtt{onhi}} + T_{\mathtt{offhi}} + T_{\mathtt{lo}}$.

$$\begin{aligned}
&T_{\mathtt{onhi}}(user, ch, pk_{\mathtt{MM}}, \mathsf{crt}, kbt) \triangleq \\
&\quad \nu\, \mathsf{TXdata}. \\
&\quad \mathtt{let}\ \mathsf{TX} := \langle \mathsf{TXdata}, \mathtt{hi}\rangle\ \mathtt{in} \\
&\quad \nu t.\mathtt{let} z_1 := \phi(t, \mathfrak{g})\ \mathtt{in} \\
&\quad \overline{ch}\langle z_1\rangle. \\
&\quad ch(z_2). \\
&\quad \mathtt{let}\ k_t := \mathtt{h}(\phi(t, z_2))\ \mathtt{in} \\
&\quad \overline{ch}\langle\{\mathsf{crt}\}_{k_t}\rangle. \\
&\quad ch(n). \\
&\quad \mathtt{let}\ \langle \mathsf{B}, \mathsf{B}_s\rangle := \mathtt{dec}(k_t, n)\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{vcheck}(\mathsf{B}_s, pk_{\mathtt{MM}}) = \mathsf{B}\ \mathtt{then} \\
&\quad \mathtt{if}\ \mathsf{B} = z_2\ \mathtt{then} \\
&\quad user(\mathtt{uPIN}). \\
&\quad \overline{ch}\langle\{\langle \mathsf{TX}, \perp\rangle\}_{k_t}\rangle. \\
&\quad ch(y). \\
&\quad \overline{ch}\langle\{\langle \mathsf{TX}, z_2, \mathtt{dec}(k_t, y), \mathtt{uPIN}\rangle\}_{kbt}\rangle. \\
&\quad ch(r). \\
&\quad \mathtt{if}\ \mathtt{dec}(kbt, r) = \langle \mathsf{TX}, \mathtt{rtype}\rangle \\
&\quad \mathtt{if}\ \mathtt{rtype} = \mathtt{accept}\ \mathtt{then} \\
&\quad \overline{ch}\langle\mathtt{auth}\rangle
\end{aligned}$$

Figure 4.5: Specification of the online high-value terminal's role in UTX.

Initially, each terminal proceeds with the key establishment phase with the card, then it sends its certificate, and checks the received month certificate. High-Value terminals rely on the PIN received from the *user* channel to perform transaction authorisation. To represent the different types of transactions that can occur, we have constants `lo` and `hi` for low-value and, respectively, high-value transactions. The Online High-Value terminal process is given in Fig. 4.5. Since the transaction is high-value, the PIN is required, and after the initialisation, the user enters the PIN using the channel *user*. The terminal sends the transaction details to the card, receives the cryptogram as a response, and sends it together with the entered PIN to the bank. Since we are in online mode, the terminal authorises the transaction only after the confirmation from the bank has been received.

The Offline High-Value and Low-Value terminal behaviours are similar, and their specifications appear in the proof of Theorem 8 in Sec. 4.3.3 where we analyse the unlinkability of UTX. In the Offline High-Value, the user enters the PIN after the initialisation, and the terminal sends this PIN for verification to the card since the transaction is offline. The terminal accepts the transaction only if it receives the `ok` reply from the card. Regardless of the outcome, the terminal also sends the cryptogram to the bank. The Low-Value terminal does not require the PIN. Depending on the mode, online or offline, the terminal authorises a transaction, respectively, after receiving the confirmation from the bank or after receiving the application cryptogram from the card.

The bank. The process B, specified in Fig. 4.6, that connects to a terminal session identified by the shared key *kbt* is represented as follows.

$$\nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)$$

In addition to *kbt*, its parameters are the session channel *ch*, the system-wide channel *si* that is used by the payment system to access the card database, and the bank's secret key b_t. We model each entry inserted into the card database using the instruction $!\overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle$, and the corresponding entry can be read by receiving a message on the channel consisting of the pair $\langle si, \text{PAN}\rangle$ where the first component of the channel keeps the database private to the bank and the second component indicates the entry to look up.

After receiving a transaction request from a terminal, the bank derives the symmetric key with the card k_{bc}, obtains the PAN from the application cryptogram, and uses it to obtain the card's PIN, its master key *mk*, and the public key $\phi(c, \mathfrak{g})$ from the database channel *si*. The integrity of the cryptogram is then checked against the corresponding information from the database, taking into account the verification of the PIN if the transaction is high value. If all the checks are ok, the transaction is accepted, otherwise not; and in all cases, a confirmation message is sent in reply to the terminal.

$$
\begin{aligned}
&B(ch, si, kbt, b_t) \triangleq \\
&\quad ch(x). \\
&\quad \texttt{let}\ \langle \mathsf{TX}', z_2, \mathsf{EAC}, \mathsf{uPIN}\rangle := \texttt{dec}(kbt, x) \\
&\quad \texttt{let}\ k_{bc} := \texttt{h}(\phi(b_t, z_2))\ \texttt{in} \\
&\quad \texttt{let}\ \langle \mathsf{AC}, \mathsf{AC}_{hmac}\rangle = \texttt{dec}(k_{bc}, \mathsf{EAC})\texttt{in} \\
&\quad \texttt{let}\ \langle x_a, \mathsf{PAN}, \mathsf{TX}, \mathsf{pinV}\rangle = \mathsf{AC}\ \texttt{in} \\
&\quad \langle si, \mathrm{PAN}\rangle(\mathrm{PIN}, mk, pk_c). \\
&\quad \texttt{if}\ \texttt{h}(\langle \mathsf{AC}, mk\rangle) = \mathsf{AC}_{hmac}\ \texttt{then} \\
&\quad \texttt{if}\ \mathsf{TX} = \mathsf{TX'}\ \texttt{then} \\
&\quad \texttt{if}\ \phi(x_a, pk_c) = z_2 \\
&\quad \texttt{let}\ \langle \mathsf{TXdata}, \mathsf{TXtype}\rangle := \mathsf{TX'}\ \texttt{in} \\
&\quad \texttt{if}\ \mathsf{TXtype} = \texttt{lo}\ \texttt{then} \\
&\qquad \overline{ch}\langle\{\langle \mathsf{TX'}, \texttt{accept}\rangle\}_{kbt}\rangle \\
&\quad \texttt{else if}\ \mathsf{TXtype} = \texttt{hi}\ \texttt{then} \\
&\qquad \texttt{if}\ (\mathsf{pinV} = \texttt{ok}) \vee (\mathsf{uPIN} = \mathrm{PIN})\ \texttt{then} \\
&\qquad\quad \overline{ch}\langle\{\langle \mathsf{TX}', \texttt{accept}\rangle\}_{kbt}\rangle. \\
&\qquad \texttt{else} \\
&\qquad\quad \overline{ch}\langle\{\langle \mathsf{TX'}, \texttt{reject}\rangle\}_{kbt}\rangle.
\end{aligned}
$$

Figure 4.6: Specification of the bank's role in UTX.

The full protocol. To complete the specification, we present the full system in Fig. 4.7a. It operates as follows. At the start, the system-wide parameters are generated and public data that includes the system public key $\texttt{pk}(s)$ and the month public key $\texttt{vpk}(\chi_{\texttt{MM}})$ is announced on the public channel *out*. A new card is issued by the generation of the card-specific parameters PIN, *mk*, *c*, and PAN, and can participate in many sessions, hence the red replication operator "!". Notice that together with the card the system has a $\overline{user}\langle\mathrm{PIN}\rangle$ process that models the user entering PIN into a terminal on the channel *user* known only to terminals; and the process $\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle$ that models the entry into the card database that the bank can access to get the card's data. The bottom part of the figure specifies the back end of the system, i.e. the banks and the terminals. There is a system-wide secret key of the bank b_t and session-wise (hence the replicated) symmetric key between the bank and the terminal *kbt*. Notice that, as previously, we endow our attacker with the power to observe which agents are communicating using public session channels *ch*. In what follows, we reason about the protocol in this strong adversarial setting.

(a) The real protocol specification UTX_{impl}.

$$
\begin{aligned}
&\nu\, user, s, si, \chi_{\texttt{MM}}.\overline{out}\langle \texttt{pk}(s)\rangle.\overline{out}\langle \texttt{vpk}(\chi_{\texttt{MM}})\rangle.\Big(\\
&\quad !\nu \text{PIN}, mk, c, \text{PAN}.(\\
&\qquad \texttt{let crtC} := \texttt{vsig}(\phi(c,\mathfrak{g})\,,\chi_{\texttt{MM}})\,\texttt{in}\\
&\qquad\quad !\nu ch.\overline{card}\langle ch\rangle.C(ch,c,\phi(s,\mathfrak{g})\,,\texttt{crtC},\text{PAN},mk,\text{PIN})\\
&\qquad \mid !\overline{user}\langle \text{PIN}\rangle \mid !\overline{\langle si,\text{PAN}\rangle}\langle\langle \text{PIN},mk,\phi(c,\mathfrak{g})\rangle\rangle\ )\ \mid\\
&\quad \nu b_t.!\nu kbt.(\\
&\qquad \nu ch.\overline{bank}\langle ch\rangle.B(ch,si,kbt,b_t)\ \mid\\
&\qquad \texttt{let crt} := \langle\langle \texttt{MM},\phi(b_t,\mathfrak{g})\rangle,\texttt{sig}(\langle \texttt{MM},\phi(b_t,\mathfrak{g})\rangle,s)\rangle\texttt{in}\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user,ch,\texttt{vpk}(\chi_{\texttt{MM}})\,,\texttt{crt},kbt))\ \Big)
\end{aligned}
$$

(b) The ideal unlinkable protocol specification UTX_{spec}.

$$
\begin{aligned}
&\nu\, user, s, si, \chi_{\texttt{MM}}.\overline{out}\langle \texttt{pk}(s)\rangle.\overline{out}\langle \texttt{vpk}(\chi_{\texttt{MM}})\rangle.\Big(\\
&\quad !\nu \text{PIN}, mk, c, \text{PAN}.(\\
&\qquad \texttt{let crtC} := \texttt{vsig}(\phi(c,\mathfrak{g})\,,\chi_{\texttt{MM}})\,\texttt{in}\\
&\qquad\quad \nu ch.\overline{card}\langle ch\rangle.C(ch,c,\phi(s,\mathfrak{g})\,,\texttt{crtC},\text{PAN},mk,\text{PIN})\\
&\qquad \mid !\overline{user}\langle \text{PIN}\rangle \mid !\overline{\langle si,\text{PAN}\rangle}\langle\langle \text{PIN},mk,\phi(c,\mathfrak{g})\rangle\rangle\ )\ \mid\\
&\quad \nu b_t.!\nu kbt.(\\
&\qquad \nu ch.\overline{bank}\langle ch\rangle.B(ch,si,kbt,b_t)\ \mid\\
&\qquad \texttt{let crt} := \langle\langle \texttt{MM},\phi(b_t,\mathfrak{g})\rangle,\texttt{sig}(\langle \texttt{MM},\phi(b_t,\mathfrak{g})\rangle,s)\rangle\texttt{in}\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user,ch,\texttt{vpk}(\chi_{\texttt{MM}})\,,\texttt{crt},kbt))\ \Big)
\end{aligned}
$$

Figure 4.7: Specifications for the real UTX protocol and its ideal unlinkable version.

4.3.3 The proof of unlinkability of UTX

In this section, we give the formal definition of unlinkability for UTX protocol and formally prove that the UTX is unlinkable.

The formal definition of unlinkability. Recall that the core of the unlinkability scheme is the equivalence between the idealised and the real-world system. We define both in Fig. 4.7. Notice that in the system UTX_{impl} defining the real-world scenario the card with the private key c can participate in any number of sessions, while in the specification UTX_{spec} defining the idealised situation, the card can only participate in one session at most. The possibility of entering the PIN arbitrarily many times is given by the process $!\overline{user}\langle \text{PIN}\rangle$, and accessing the database in arbitrarily many bank-terminal sessions given by the process $!\overline{\langle si,\text{PAN}\rangle}\langle\langle \text{PIN},mk,\phi(c,\mathfrak{g})\rangle\rangle$, remains the same for both real and idealised worlds.

We are ready to give the unlinkability definition for the UTX protocol.

Definition 14. (unlinkability) *We say that the payments are unlinkable if* $UTX_{impl} \sim UTX_{spec}$.

There is a difference with the Def. 9 of unlinkability for key establishment given in Sec. 3.3.3 where the terminal and the bank are deliberately omitted. The reason is that the key establishment in isolation requires no shared secret between the

parties, yet to execute, for instance, a full high-value transaction, at least the PIN is required to be shared between all three parties involved in the protocol. In addition, to validate a transaction there is a secret *mk* shared between the bank and the card, meaning that, even if only transactions without the PIN are modelled, the bank and card must be explicitly modelled in a transaction and we cannot incorporate the compositionality argument from Theorem 4 in Sec. 2.2.4, however compositionality will appear in the corollary of the theorem we are ready to formulate.

Theorem 8. *The* UTX *protocol is unlinkable.*

Proof. It is worth outlining the idea of the proof first. As in the examples from Chapter 2 and Theorem 7 from the previous chapter to prove that $UTX_{\text{spec}} \sim UTX_{\text{impl}}$ we construct a relation $\mathfrak{R}$, s.t. $UTX_{\text{spec}}\,\mathfrak{R}\,UTX_{\text{impl}}$ and then check that it is a quasi-open bisimulation (Def. 5 from Sec. 2.2.2). i.e. we verify that each state can match each other's actions, that $\mathfrak{R}$ is *open*, i.e. that an attacker to distinguish between two worlds by manipulating free variables (Def. 4 from Sec. 2.2.2), and that any related states are *statically equivalent* (Def. 3 from Sec. 2.2.2). We form the relation $\mathfrak{R}$ by pairing the states based on the number of sessions *started* with terminals, cards, and banks and the respective stages of each session; we ignore the number of exhausted processes that model entering the PIN and accessing the database for card's details.

First, let us define three parameters lists $\vec{\chi} := (ch, c, s, \chi_{\texttt{MM}}, \text{PAN}, mk, \text{PIN})$[1], $\vec{\psi} := (ch, pk_{\texttt{MM}}, \mathsf{crt}, ch)$, and $\vec{\omega} := (ch, si, ch, b_t)$; and "tail" subprocesses representing different stages of the execution for each role specification. A process highlighted in blue defines the process starting from the next line, i.e. each tail subprocess defines actions left to complete the protocol. For instance, $\mathcal{C}_2(\vec{\chi}, z_1) \triangleq \nu a.\overline{ch}\langle\phi(a, \phi(c, \mathfrak{g}))\rangle.\mathcal{C}_3(\vec{\chi}, z_1, a)$. For terminal tails to be properly defined we include the output of the `auth` message when the respective terminal accepts the transaction.

[1] $\vec{\chi}$, that always comes without vector, should not be confused with the private key $\chi_{\texttt{MM}}$, that always cones with a subscript `MM`.

$C(ch, c, pk_s, \mathtt{vsig}_{\mathtt{MM}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq$

$\mathcal{C}_1(\vec{\chi})$

$ch(z_1).$

$\mathcal{C}_2(\vec{\chi}, z_1)$

$\nu a.\overline{ch}\langle\phi(a, \phi(c, \mathfrak{g}))\rangle.$

$\mathcal{C}_3(\vec{\chi}, z_1, a)$

$\mathtt{let}\ k_c := \mathtt{h}(\phi(a \cdot c, z_1))\ \mathtt{in}$

$ch(m).$

$\mathcal{C}_4(\vec{\chi}, k_c, a, m)$

$\mathtt{let}\ \langle \mathsf{MC}, \mathsf{MC}_s\rangle := \mathtt{dec}(k_c, m)\ \mathtt{in}$

$\mathtt{if}\ \mathtt{check}(\mathsf{MC}_s, pk_s) = \mathsf{MC}\ \mathtt{then}$

$\mathtt{if}\ \mathtt{p}_1(\mathsf{MC}) = \mathtt{MM}\ \mathtt{then}$

$\overline{ch}\langle\{\langle\phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}_{\mathtt{MM}})\rangle\}_{k_c}\rangle.$

$\mathcal{C}_5(\vec{\chi}, k_c, a, m)$

$ch(x).$

$\mathcal{C}_6(\vec{\chi}, k_c, a, m, x)$

$\mathtt{let}\ \langle \mathsf{TX}, \mathsf{uPin}\rangle := \mathtt{dec}(k_c, x)\ \mathtt{in}$

$\mathtt{let}\ \mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX}\rangle\ \mathtt{in}$

$\mathtt{let}\ \mathsf{AC}^{\mathtt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \mathtt{ok}\rangle\ \mathtt{in}$

$\mathtt{let}\ \mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp\rangle\ \mathtt{in}$

$\mathtt{let}\ k_{cb} := \mathtt{h}(\phi(a \cdot c, \mathtt{p}_3(\mathtt{p}_1(\mathtt{dec}(k_c, m)))))\ \mathtt{in}$

$\mathtt{if}\ \mathsf{uPin} = \perp\ \mathtt{then}$

$\overline{ch}\langle\{\{\langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk\rangle)\rangle\}_{k_{cb}}\}_{k_c}\rangle$

$\mathtt{if}\ \mathsf{uPin} = \mathrm{PIN}\ \mathtt{then}$

$\overline{ch}\langle\{\{\{\langle \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\langle \mathsf{AC}^{\mathtt{ok}}, mk\rangle)\rangle\}_{k_{cb}}, \mathtt{ok}\rangle\}_{k_c}\rangle$

$\mathtt{else}\ \overline{ch}\left\langle\{\{\langle \mathsf{AC}^{\perp}, \mathtt{h}\left(\langle \mathsf{AC}^{\perp}, mk\rangle\right)\rangle\}_{k_{cb}}, \perp\rangle\}_{k_c}\right\rangle$

$\mathcal{C}_7 \triangleq 0$

$B(ch, si, kbt, b_t) \triangleq$

$\mathcal{B}_1(\vec{\omega})$

$ch(x).$

$\mathcal{B}_2(\vec{\omega}, x)$

$\mathtt{let}\ dx := \mathtt{dec}(kbt, x)\ \mathtt{in}$

$\mathtt{let}\ \mathrm{PAN} := \mathtt{p}_2(\mathtt{p}_1(\mathtt{dec}(\phi(b_t, \mathtt{p}_2(dx)), \mathtt{p}_3(dx))))\ \mathtt{in}$

$\langle si, \mathrm{PAN}\rangle(y).$

$\mathcal{B}_3(\vec{\omega}, x, y)$

$\mathtt{let}\ \langle \mathsf{TX}', z_2, \mathsf{EAC}, \mathsf{uPIN}\rangle := \mathtt{dec}(kbt, x)$

$\mathtt{let}\ k_{bc} := \mathtt{h}(\phi(b_t, z_2))\ \mathtt{in}$

$\mathtt{let}\ \langle \mathsf{AC}, \mathsf{AC}_{hmac}\rangle = \mathtt{dec}(k_{bc}, \mathsf{EAC})\ \mathtt{in}$

$\mathtt{let}\ \langle \mathrm{PIN}, mk, pk_c\rangle := y$

$\mathtt{if}\ \mathtt{h}(\langle \mathsf{AC}, mk\rangle) = \mathsf{AC}_{hmac}\ \mathtt{then}$

$\mathtt{if}\ \mathtt{p}_3(\mathsf{AC}) = \mathsf{TX}'\ \mathtt{then}$

$\mathtt{if}\ \phi(\mathtt{p}_1(\mathsf{AC}), pk_c) = z_2$

$\mathtt{let}\ r := \langle \mathsf{TX}', \mathtt{accept}\rangle\mathtt{in}$

$\mathtt{if}\ \mathtt{p}_2(TX') = \mathtt{lo}\ \mathtt{then}$

$\overline{ch}\langle\{r\}_{kbt}\rangle$

$\mathtt{if}\ \mathtt{p}_2(\mathsf{TX}') = \mathtt{hi}\ \mathtt{then}$

$\mathtt{if}\ \mathtt{p}_4(\mathsf{AC}) = \mathtt{ok}\ \mathtt{then}$

$\overline{ch}\langle\{r\}_{kbt}\rangle$

$\mathtt{else}\ \mathtt{if}\ \mathsf{uPIN} = \mathrm{PIN}\ \mathtt{then}$

$\overline{ch}\langle\{r\}_{kbt}\rangle$

$\mathcal{B}_4 \triangleq 0$

$T_{\texttt{onhi}}(user, ch, pk_{\texttt{MM}}, \mathsf{crt}, kbt) \triangleq$
$\mathcal{TONH}_1(user, \vec{\psi})$
$\nu\,\mathsf{TXdata}.\texttt{let}\ \mathsf{TX} := \langle \mathsf{TXdata}, \texttt{hi}\rangle \texttt{in}$
$\nu t.\overline{ch}\langle\phi(t, \mathfrak{g})\rangle.$
$\mathcal{TONH}_2(user, \vec{\psi}, t, \mathsf{TX})$
$ch(z_2).$
$\mathcal{TONH}_3(user, \vec{\psi}, t, \mathsf{TX}, z_2)$
$\texttt{let}\ k_t := \texttt{h}(\phi(t, z_2))\ \texttt{in}$
$\overline{ch}\langle\{\mathsf{crt}\}_{k_t}\rangle.$
$\mathcal{TONH}_4(user, \vec{\psi}, t, \mathsf{TX}, z_2)$
$ch(n).$
$\mathcal{TONH}_5(user, \vec{\psi}, t, \mathsf{TX}, z_2, n)$
$\texttt{let}\ \langle \mathsf{B}, \mathsf{B}_s\rangle := \texttt{dec}(k_t, n)\ \texttt{in}$
$\texttt{if}\ \texttt{vcheck}(\mathsf{B}_s, pk_{\texttt{MM}}) = \mathsf{B}\ \texttt{then}$
$\texttt{if}\ \mathsf{B} = z_2\ \texttt{then}$
$user(\texttt{uPIN}).$
$\mathcal{TONH}_6(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPIN})$
$\overline{ch}\langle\{\langle \mathsf{TX}, \bot\rangle\}_{k_t}\rangle.$
$\mathcal{TONH}_7(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPIN})$
$ch(y).$
$\mathcal{TONH}_8(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPIN}, y)$
$\overline{ch}\langle\{\langle \mathsf{TX}, z_2, \texttt{dec}(k_t, y), \texttt{uPIN}\rangle\}_{kbt}\rangle.$
$\mathcal{TONH}_9(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPIN}, y)$
$ch(r).$
$\mathcal{TONH}_{10}(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPIN}, y, r)$
$\texttt{if}\ \texttt{p}_1(\texttt{dec}(kbt, r)) = \mathsf{TX}\ \texttt{then}$
$\texttt{if}\ \texttt{p}_2(\texttt{dec}(kbt, r)) = \texttt{accept}\ \texttt{then}$
$\overline{ch}\langle\texttt{auth}\rangle\mathcal{TONH}_{11} \triangleq 0$

$T_{\texttt{offhi}}(user, ch, pk_{\texttt{MM}}, \mathsf{crt}, kbt) \triangleq$
$\mathcal{TOFH}_1(user, \vec{\psi})$
$\nu\,\mathsf{TXdata}.\texttt{let}\ \mathsf{TX} := \langle \mathsf{TXdata}, \texttt{hi}\rangle \texttt{in}$
$\nu t.\overline{ch}\langle\phi(t, \mathfrak{g})\rangle.$
$\mathcal{TOFH}_2(user, \vec{\psi}, t, \mathsf{TX})$
$ch(z_2).$
$\mathcal{TOFH}_3(user, \vec{\psi}, t, \mathsf{TX}, z_2)$
$\texttt{let}\ k_t := \texttt{h}(\phi(t, z_2))\ \texttt{in}$
$\overline{ch}\langle\{\langle\mathsf{crt}\rangle\}_{k_t}\rangle.$
$\mathcal{TOFH}_4(user, \vec{\psi}, t, \mathsf{TX}, z_2)$
$ch(n).$
$\mathcal{TOFH}_5(user, \vec{\psi}, t, \mathsf{TX}, z_2, n)$
$\texttt{let}\ \langle \mathsf{B}, \mathsf{B}_s\rangle := \texttt{dec}(k_t, n)\ \texttt{in}$
$\texttt{if}\ \texttt{vcheck}(\mathsf{B}_s, pk_{\texttt{MM}}) = \mathsf{B}\ \texttt{then}$
$\texttt{if}\ \mathsf{B} = z_2\ \texttt{then}$
$user(\texttt{uPIN}).$
$\mathcal{TOFH}_6(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPin})$
$\overline{ch}\langle\{\langle \mathsf{TX}, \texttt{uPin}\rangle\}_{k_t}\rangle.$
$\mathcal{TOFH}_7(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPin})$
$ch(y).$
$\mathcal{TOFH}_8(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPin}, y)$
$\texttt{if}\ \texttt{p}_2(\texttt{dec}(k_t, y)) = \texttt{ok}\ \texttt{then}$
$\overline{ch}\langle\texttt{auth}\rangle.$
$\mathcal{TOFH}_9(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPin}, y)$
$\overline{ch}\langle\{\langle \mathsf{TX}, z_2, \mathsf{EAC}, \bot\rangle\}_{kbt}\rangle$
$\mathcal{TOFH}_{10}(user, \vec{\psi}, t, \mathsf{TX}, z_2, n, \texttt{uPin}, y)$
$\texttt{if}\ \texttt{p}_2(\texttt{dec}(k_t, y)) \neq \texttt{ok}\ \texttt{then}$
$\overline{ch}\langle\{\langle \mathsf{TX}, z_2, \mathsf{EAC}, \bot\rangle\}_{kbt}\rangle\mathcal{TOFH}_{11} \triangleq 0$

$T_{\texttt{lo}}(ch, pk_{\texttt{MM}}, \mathsf{crt}, kbt) \triangleq$
$\mathcal{TLO}_1(\vec{\psi})$
$\nu\,\mathsf{TXdata}.\texttt{let}\ \mathsf{TX} := \langle \mathsf{TXdata}, \texttt{lo}\rangle \texttt{in}$
$\nu t.\overline{ch}\langle\phi(t, \mathfrak{g})\rangle.$
$\mathcal{TLO}_2(\vec{\psi}, t, \mathsf{TX})$
$ch(z_2).$
$\mathcal{TLO}_3(\vec{\psi}, t, \mathsf{TX}, z_2)$
$\texttt{let}\ k_t := \texttt{h}(\phi(t, z_2))\ \texttt{in}$
$\overline{ch}\langle\{\mathsf{crt}\}_{k_t}\rangle.$
$\mathcal{TLO}_4(\vec{\psi}, t, \mathsf{TX}, z_2)$
$ch(n).$
$\mathcal{TLO}_5(\vec{\psi}, t, \mathsf{TX}, z_2, n)$
$\texttt{let}\ \langle \mathsf{B}, \mathsf{B}_s\rangle := \texttt{dec}(k_t, n)\ \texttt{in}$
$\texttt{if}\ \texttt{vcheck}(\mathsf{B}_s, pk_{\texttt{MM}}) = \mathsf{B}\ \texttt{then}$
$\texttt{if}\ \mathsf{B} = z_2\ \texttt{then}$
$\overline{ch}\langle\{\langle \mathsf{TX}, \bot\rangle\}_{k_t}\rangle.$
$\mathcal{TLO}_6(\vec{\psi}, t, \mathsf{TX}, z_2, n)$
$ch(y).$
$\mathcal{TLO}_7(\vec{\psi}, t, \mathsf{TX}, z_2, n, y)$
$\overline{ch}\langle\{\langle \mathsf{TX}, z_2, \texttt{dec}(k_t, y), \bot\rangle\}_{kbt}\rangle.$
$\mathcal{TLO}_8(\vec{\psi}, t, \mathsf{TX}, z_2, n, y)$
$ch(r).$
$\mathcal{TLO}_9(\vec{\psi}, t, \mathsf{TX}, z_2, n, y, r)$
$\texttt{if}\ \texttt{p}_1(\texttt{dec}(kbt, r)) = \mathsf{TX}\ \texttt{then}$
$\texttt{if}\ \texttt{p}_2(\texttt{dec}(kbt, r)) = \texttt{accept}\ \texttt{then}$
$\overline{ch}\langle\texttt{auth}\rangle\mathcal{TLO}_{10} \triangleq 0$

The idea behind forming $\mathfrak{R}$ is to pair all *reachable* states based on the number of sessions, yet ignoring the number of existing cards. If not specified otherwise, below we talk about *started* sessions, i.e. the ones with the announced channel, hence we define the following sets of sessions: $\mathcal{D} := \{1 \dots D\}$ for cards, $\mathcal{FG} := \{1 \dots F+G\}$ for terminals, and $\mathcal{FM} := \{1 \dots F+M\}$ for the bank, where F is the number of bank-terminal sessions with a shared secret key *kbt*. These reachable states are defined by partitions of the session sets, where the element of the partition defines all sessions at a certain stage. We consider the following partitions. $A := \{\alpha_1 \dots \alpha_7\}$ of $\mathcal{D}$, $\Gamma := \{\gamma_1^{\mathtt{on}} \dots \gamma_{11}^{\mathtt{on}}, \gamma_1^{\mathtt{of}} \dots \gamma_1^{\mathtt{of}} 1, \gamma_1^{\mathtt{lo}} \dots \gamma_{10}^{\mathtt{lo}}\}$ of $\mathcal{FG}$, and $B := \{\beta_1 \dots \beta_4\}$ of $\mathcal{FM}$. Here, e.g. α_2 defines all sessions where the actions left are defined by the process of the form $C_2(\cdot)$, or, similarly, $\gamma_5^{\mathtt{of}}$ defines all offline high-value terminal sessions where the actions left are defined by $\mathcal{TOFH}_5(\cdot)$.

We also define the list of global parameters $\vec{e} := (user, s, si, \chi_{\mathtt{MM}})$, introduce the shorthand for different transaction types $\mathsf{TX}_i := \langle \mathsf{TXdata}_i, \mathtt{lo} \rangle$ or $\langle \mathsf{TXdata}_i, \mathtt{hi} \rangle$, and define the numbers $E := \bigcup_{i=3}^{7} \alpha_i$ and $L := \bigcup_{i=2}^{11} \gamma_i^{\mathtt{on}} \cup \bigcup_{i=2}^{11} \gamma_i^{\mathtt{of}} \cup \bigcup_{i=2}^{10} \gamma_i^{\mathtt{lo}}$ standing for the number of session where fresh blinding factor a_i or fresh ephemeral private key t_i has already been generated.

The following processes are also required to make the definition of $\mathfrak{R}$ less bulky.

$$
\begin{aligned}
PC_{\text{spec}} \triangleq \nu \text{PIN}, mk, c, \text{PAN}.(& \\
& \nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s, \mathfrak{g}), \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}), \text{PAN}, mk, \text{PIN}) \mid \\
& !\overline{user}\langle \text{PIN}\rangle \mid \\
& !\overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle)
\end{aligned}
$$

$$
\begin{aligned}
PC_{\text{impl}} \triangleq \nu \text{PIN}, mk, c, \text{PAN}.!(& \\
& \nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s, \mathfrak{g}), \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}), \text{PAN}, mk, \text{PIN}) \mid \\
& \overline{user}\langle \text{PIN}\rangle \mid \\
& \overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle)
\end{aligned}
$$

$$
\begin{aligned}
PBT \triangleq \nu b_t.!\nu kbt.(& \\
& \nu ch.\overline{bank}\langle ch\rangle.B(ch, si, ch, b_t) \mid \\
& \mathtt{let}\ \mathtt{crt} := \langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle\ \mathtt{in} \\
& \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{onhi}}(user, ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathtt{crt}, kbt) + \\
& \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{offhi}}(user, ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathtt{crt}, kbt) + \\
& \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{lo}}(ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathtt{crt}, kbt))
\end{aligned}
$$

We define the processes corresponding to different stages of the bank's and card's executions, and the processes corresponding to the user entering the PIN, and the bank looking up for the card's details. In what follows $k_i^c(a,x) \coloneqq \mathtt{h}(\phi(a \cdot c_i, x))$. Also, let K be the total number of card sessions for which the channel is not yet announced in the spec world, H be the number of cards in the impl world, the list $\vec{K} \coloneqq \{K_1, \ldots, K_H\}$ be s.t. K_h defines the number of sessions with the card h for which the channel is not yet announced in the impl world, $\Lambda \coloneqq \{\lambda_1 \ldots \lambda_H\}$ be the partition of $\mathcal{D}$, where λ_h is the set of sessions with the card h in the impl world, and finally $D_h \coloneqq |\lambda_h|$. To keep track of inputs in the card, bank, and terminal sessions we use matricies $\underset{D\times 3}{X}$, $\underset{G\times 1}{Y}$ and $\underset{F\times 4}{Z}$ respectively; the element (i,j) defines jth input in the ith session.

$$B_i = \begin{cases} vch.\overline{bank}\langle ch\rangle.B(ch, si, kbt_i, b_t) & \text{if } i \in \mathcal{FG} \setminus \mathcal{FM} \\ \mathcal{B}_1(\vec{\omega}_i) & \text{if } i \in \beta_1 \\ \mathcal{B}_2(\vec{\omega}_i, Y_1^i\sigma) & \text{if } i \in \beta_2 \\ \mathcal{B}_3(\vec{\omega}_i, Y_1^i\sigma, \mathrm{DB}) & \text{if } i \in \beta_3 \\ \mathcal{B}_4 & \text{if } i \in \beta_4 \end{cases}$$

$$C_i = \begin{cases} vch.\overline{card}\langle ch\rangle. \\ C(ch, c_i, \phi(s, \mathfrak{g}), \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}), \mathrm{PAN}_i, mk_i, \mathrm{PIN}_i) & \text{if } i \notin \mathcal{D}, i \le K \\ \mathcal{C}_1(\vec{\chi}_i) & \text{if } i \in \alpha_1 \\ \mathcal{C}_2(\vec{\chi}_i, X_i^1\sigma) & \text{if } i \in \alpha_2 \\ \mathcal{C}_3(\vec{\chi}_i, X_i^1\sigma, a_i) & \text{if } i \in \alpha_3 \\ \mathcal{C}_4(\vec{\chi}_i, k_i^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma) & \text{if } i \in \alpha_4 \\ \mathcal{C}_5(\vec{\chi}_i, k_i^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma) & \text{if } i \in \alpha_5 \\ \mathcal{C}_6(\vec{\chi}_i, k_i^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma, X_i^3\sigma) & \text{if } i \in \alpha_6 \\ \mathcal{C}_7 & \text{if } i \in \alpha_7 \end{cases}$$

$$C_i^j = \begin{cases} vch.\overline{card}\langle ch\rangle.C(ch, c_j, \phi(s, \mathfrak{g}), \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}), \mathrm{PAN}_j, mk_j, \mathrm{PIN}_j) & \text{if } i \notin \mathcal{D}, i \le K_j \\ \mathcal{C}_1(\vec{\chi}_j) & \text{if } i \in \alpha_1 \cap \lambda_j \\ \mathcal{C}_2(\vec{\chi}_j, X_i^1\sigma) & \text{if } i \in \alpha_2 \cap \lambda_j \\ \mathcal{C}_3(\vec{\chi}_j, X_i^1\sigma, a_i) & \text{if } i \in \alpha_3 \cap \lambda_j \\ \mathcal{C}_4(\vec{\chi}_j, k_j^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma) & \text{if } i \in \alpha_4 \cap \lambda_j \\ \mathcal{C}_5(\vec{\chi}_j, k_j^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma) & \text{if } i \in \alpha_5 \cap \lambda_j \\ \mathcal{C}_6(\vec{\chi}_j, k_j^c(a_i, X_i^1\sigma), a_i, X_i^2\sigma, X_i^3\sigma) & \text{if } i \in \alpha_6 \cap \lambda_j \\ \mathcal{C}_7 & \text{if } i \in \alpha_7 \cap \lambda_j \end{cases}$$

$$U_i^j = \begin{cases} 0 & \text{if } j \le H \text{ and } \exists l \in \bigcup_{t=6}^{11} \gamma_t^{\mathtt{on}} \text{ or } l \in \bigcup_{t=6}^{9} \gamma_t^{\mathtt{of}} \text{ s.t. } \mathrm{PIN}_j \text{ is consumed in session } l \\ \overline{user}\langle \mathrm{PIN}_j \rangle & \text{if } j \le H \text{ and else} \end{cases}$$

$$DB_i^j = \begin{cases} 0 & \text{if } j \le H \text{ and } \exists l \in \beta_3 \cup \beta_4, \text{ s.t. parameters of the card } j \text{ are consumed in the session } l \\ \overline{\langle si, \mathrm{PAN}_j\rangle}\big\langle\langle \mathrm{PIN}_j, mk_j, \phi(c_j, \mathfrak{g})\rangle\big\rangle & \text{if } j \le H \text{ and else} \end{cases}$$

We define the processes corresponding to different stages of the terminal's execution.

$$T_i = \begin{cases} \mathcal{TONH}_1(user, \vec{\psi}_i) & \text{if } i \in \gamma_1^{\mathtt{on}} \\ \mathcal{TONH}_2(user, \vec{\psi}_i, t_i, \mathrm{TX}_i) & \text{if } i \in \gamma_2^{\mathtt{on}} \\ \mathcal{TONH}_3(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_3^{\mathtt{on}} \\ \mathcal{TONH}_4(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_4^{\mathtt{on}} \\ \mathcal{TONH}_5(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma) & \text{if } i \in \gamma_5^{\mathtt{on}} \\ \mathcal{TONH}_6(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}) & \text{if } i \in \gamma_6^{\mathtt{on}} \\ \mathcal{TONH}_7(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}) & \text{if } i \in \gamma_7^{\mathtt{on}} \\ \mathcal{TONH}_8(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma) & \text{if } i \in \gamma_8^{\mathtt{on}} \\ \mathcal{TONH}_9(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma) & \text{if } i \in \gamma_9^{\mathtt{on}} \\ \mathcal{TONH}_{10}(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma, \mathrm{R}) & \text{if } i \in \gamma_{10}^{\mathtt{on}} \\ \mathcal{TONH}_{11} & \text{if } i \in \gamma_{11}^{\mathtt{on}} \\ \mathcal{TOFH}_1(user, \vec{\psi}_i) & \text{if } i \in \gamma_1^{\mathtt{of}} \\ \mathcal{TOFH}_2(user, \vec{\psi}_i, t_i, \mathrm{TX}_i) & \text{if } i \in \gamma_2^{\mathtt{of}} \\ \mathcal{TONH}_3(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_3^{\mathtt{of}} \\ \mathcal{TONH}_4(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_4^{\mathtt{of}} \\ \mathcal{TONH}_5(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma) & \text{if } i \in \gamma_5^{\mathtt{of}} \\ \mathcal{TONH}_6(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}) & \text{if } i \in \gamma_6^{\mathtt{of}} \\ \mathcal{TONH}_7(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}) & \text{if } i \in \gamma_7^{\mathtt{of}} \\ \mathcal{TONH}_8(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma) & \text{if } i \in \gamma_8^{\mathtt{of}} \\ \mathcal{TONH}_9(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma) & \text{if } i \in \gamma_9^{\mathtt{of}} \\ \mathcal{TONH}_{10}(user, \vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, \mathrm{uPIN}, Z_i^3\sigma) & \text{if } i \in \gamma_{10}^{\mathtt{of}} \\ \mathcal{TONH}_{11} & \text{if } i \in \gamma_{11}^{\mathtt{of}} \\ \mathcal{TLO}_1(\vec{\psi}_i) & \text{if } i \in \gamma_1^{\mathtt{lo}} \\ \mathcal{TLO}_2(\vec{\psi}_i, t_i, \mathrm{TX}_i) & \text{if } i \in \gamma_2^{\mathtt{lo}} \\ \mathcal{TLO}_3(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_3^{\mathtt{lo}} \\ \mathcal{TLO}_4(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma) & \text{if } i \in \gamma_4^{\mathtt{lo}} \\ \mathcal{TLO}_5(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma) & \text{if } i \in \gamma_5^{\mathtt{lo}} \\ \mathcal{TLO}_6(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma) & \text{if } i \in \gamma_6^{\mathtt{lo}} \\ \mathcal{TLO}_7(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, Z_i^3\sigma) & \text{if } i \in \gamma_7^{\mathtt{lo}} \\ \mathcal{TLO}_8(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, Z_i^3\sigma) & \text{if } i \in \gamma_8^{\mathtt{lo}} \\ \mathcal{TLO}_9(\vec{\psi}_i, t_i, \mathrm{TX}_i, Z_i^1\sigma, Z_i^2\sigma, Z_i^3\sigma, \mathrm{R}) & \text{if } i \in \gamma_9^{\mathtt{lo}} \\ \mathcal{TLO}_{10} & \text{if } i \in \gamma_{10}^{\mathtt{lo}} \end{cases}$$

Or, otherwise

$$T_i = \begin{array}{l} \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{onhi}}(user, ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathsf{crt}, kbt_i) + \\ \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{offhi}}(user, ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathsf{crt}, kbt_i) + \\ \nu ch.\overline{term}\langle ch\rangle.T_{\mathtt{lo}}(ch, \mathtt{vpk}(\chi_{\mathtt{MM}}), \mathsf{crt}, kbt)_i \end{array} \quad \text{if } i \in \mathcal{FM} \setminus \mathcal{FG}$$

Now we define the generic state $(K, F, A, \Gamma, B)_{\text{spec}}(X, Y, Z)$ in the spec world.

$$
\begin{aligned}
&(K, F, A, \Gamma, B)_{\text{spec}}(X, Y, Z) \triangleq \\
&\quad \nu\, \vec{e}, \mathrm{PIN}_{1\ldots D+K}, mk_{1\ldots D+K}, c_{1\ldots D+K}, \mathrm{PAN}_{1\ldots D+K}, \\
&\quad \dot{ch}_{1\ldots D}, a_{1\ldots E}, b_t, ch_{1\ldots F+G+M}, \ddot{ch}_{1\ldots F+G}, \\
&\quad \dddot{ch}_{1\ldots F+M}, t_{1\ldots L}, \mathsf{TX}_{1\ldots L}.(\sigma \mid \\
&\qquad C_1 \mid \ldots \mid 0 \mid !\overline{user}\langle \mathrm{PIN}_1 \rangle \mid \\
&\qquad\quad \ldots \mid 0 \mid !\overline{\langle si, \mathrm{PAN}_1 \rangle}\langle\langle \mathrm{PIN}_1, mk_1, \phi(c_1, \mathfrak{g}) \rangle\rangle) \mid \\
&\qquad \ldots \\
&\qquad C_i \mid \ldots \mid 0 \mid !\overline{user}\langle \mathrm{PIN}_i \rangle \mid \\
&\qquad\quad \ldots \mid 0 \mid !\overline{\langle si, \mathrm{PAN}_i \rangle}\langle\langle \mathrm{PIN}_i, mk_i, \phi(c_i, \mathfrak{g}) \rangle\rangle) \mid \\
&\qquad \ldots \\
&\qquad C_{D+K} \mid \ldots \mid 0 \mid !\overline{user}\langle \mathrm{PIN}_{D+K} \rangle \mid \\
&\qquad\quad \ldots \mid 0 \mid !\overline{\langle si, \mathrm{PAN}_{D+K} \rangle}\langle\langle \mathrm{PIN}_{D+K}, mk_{D+K}, \phi(c_{D+K}, \mathfrak{g}) \rangle\rangle) \mid \\
&\qquad !PC_{\text{spec}} \mid \\
&\qquad B_1 \mid T_1 \mid \\
&\qquad \ldots \mid \\
&\qquad B_j \mid T_j \mid \\
&\qquad \ldots \mid \\
&\qquad B_{F+G+M} \mid T_{F+G+M} \mid !PBT)
\end{aligned}
$$

$(\vec{K}, F, A, \Gamma, B, \Lambda)_{\text{impl}}(X, Y, Z)$ defines the generic state in impl scenario.

$$
\begin{aligned}
&(\vec{K}, F, A, \Gamma, B, \Lambda)_{\text{impl}}(X, Y, Z) \triangleq \\
&\quad \nu\vec{\epsilon}, \mathrm{PIN}_{1\ldots H}, mk_{1\ldots H}, c_{1\ldots H}, \mathrm{PAN}_{1\ldots H}, \dot{ch}_{1\ldots D}, \\
&\quad a_{1\ldots E}, b_t, ch_{1\ldots F+G+M}, \ddot{ch}_{1\ldots F+G}, \dddot{ch}_{1\ldots F+M} \\
&\quad t_{1\ldots L}, \mathsf{TX}_{1\ldots L}.(\ \theta \mid \\
&\qquad C_1^1 \mid U_1^1 \mid DB_1^1 \mid \\
&\qquad \dot{C}_{i_1}^1 \mid U_{i_1}^1 \mid DB_{i_1}^1 \mid \\
&\qquad \dot{C}_{D_1+K_1}^1 \mid U_{D_1+K_1}^1 \mid DB_{D_1+K_1}^1 \mid \\
&\qquad !(\nu ch.\overline{card}\langle ch\rangle.C(ch, c_j, \phi(s, \mathfrak{g})\,, \mathtt{vsig}(\phi(c, \mathfrak{g})\,, \chi_{\mathtt{MM}})\,, \mathrm{PAN}_j, mk_j, \mathrm{PIN}_j) \mid \\
&\qquad \overline{user}\langle \mathrm{PIN}_1\rangle \mid DB(si, \mathrm{PAN}_1, mk_1, \mathrm{PIN}_1)) \mid \\
&\qquad \dot{C}_{D_{h-1}+K_{h-1}+1}^h \mid U_{D_{h-1}+K_{h-1}+1}^h \mid DB_{D_{h-1}+K_{h-1}+1}^h \mid \\
&\qquad \dot{C}_{i_h}^h \mid U_{i_h}^h \mid DB_{i_h}^h \mid \\
&\qquad \dot{C}_{D_{h-1}+K_{h-1}+D_h+K_h}^h \mid U_{D_{h-1}+K_{h-1}+D_h+K_h}^h \mid \\
&\qquad\quad DB_{D_{h-1}+K_{h-1}+D_h+K_h}^h \mid \\
&\qquad !(\nu ch.\overline{card}\langle ch\rangle.C(ch, c_h, \phi(s, \mathfrak{g})\,, \mathtt{vsig}(\phi(c, \mathfrak{g})\,, \chi_{\mathtt{MM}})\,, \mathrm{PAN}_h, mk_h, \mathrm{PIN}_h) \mid \\
&\qquad \overline{user}\langle \mathrm{PIN}_h\rangle \mid DB(si, \mathrm{PAN}_h, mk_h, \mathrm{PIN}_h)) \mid \\
&\qquad \dot{C}_{D_{H-1}+K_{H-1}+1}^H \mid U_{D_{H-1}+K_{H-1}+1}^H \mid DB_{D_{H-1}+K_{H-1}+1}^H \mid \\
&\qquad \dot{C}_{i_H}^H \mid U_{i_H}^H \mid DB_{i_H}^H \mid \\
&\qquad \dot{C}_{D_{H-1}+K_{H-1}+D_H+K_H}^H \mid U_{D_{H-1}+K_{H-1}+D_H+K_H}^H \mid \\
&\qquad\quad DB_{D_{H-1}+K_{H-1}+D_H+K_H}^H \mid \\
&\qquad !(\nu ch.\overline{card}\langle ch\rangle.C(ch, c_H, \phi(s, \mathfrak{g}), \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}), \mathrm{PAN}_H, mk_H, \mathrm{PIN}_H) \mid \\
&\qquad \overline{user}\langle \mathrm{PIN}_H\rangle \mid DB(si, \mathrm{PAN}_H, mk_H, \mathrm{PIN}_H)) \mid \\
&\qquad !PC_{\text{impl}} \mid \\
&\qquad B_1 \mid T_1 \mid \\
&\qquad \dot{B}_j \mid T_j \mid \\
&\qquad B_{F+G+M} \mid T_{F+G+M} \mid !PBT)
\end{aligned}
$$

Using the notation introduced above we are ready to define the relation $\mathfrak{R}$ as the least symmetric open relation satisfying the conditions in Fig. 4.10. Notice that the generic impl state is additionally parametrised by the partition Λ which elements track all sessions with a particular card (similarly to the partition Ω in the proof of Theorem 7).

$$
\begin{array}{rcl}
UTX_{\text{spec}} & \mathfrak{R} & UTX_{\text{impl}} \\
\\
UTX^1_{\text{spec}} \triangleq & & UTX^1_{\text{impl}} \triangleq \\
\nu\vec{\epsilon}.(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\} \mid & \mathfrak{R} & \nu\vec{\epsilon}.(\left\{{}^{\mathtt{pk}(s)}/_{pk_s}\right\} \mid \\
\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{MM}})\rangle.(!PC_{\text{spec}} \mid \nu b_t.!PBT)) & & \overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{MM}})\rangle.(!PC_{\text{impl}} \mid \nu b_t.!PBT)) \\
\\
UTX^2_{\text{spec}} \triangleq \nu\vec{\epsilon}.(\sigma_0 \mid !PC_{\text{spec}} \mid \nu b_t.!PBT) & \mathfrak{R} & \nu\vec{\epsilon}.(\sigma_0 \mid !PC_{\text{impl}} \mid \nu b_t.!PBT) \triangleq UTX^2_{\text{impl}} \\
\\
(K, F, A, \Gamma, B)_{\text{spec}}(X, Y, Z) & \mathfrak{R} & (\vec{K}, F, A, \Gamma, B, \Lambda)_{\text{impl}}(X, Y, Z)
\end{array}
$$

Figure 4.10: Defining conditions for the bisimulation relation $\mathfrak{R}$.

To conclude the definition of $\mathfrak{R}$ we should clarify which messages are available to an attacker at a given state, i.e. to define the substitutions σ_0, σ and θ. The definition of σ_0 is straightforward.

$$\sigma_0 = \left\{{}^{\mathtt{pk}(s),\mathtt{vpk}(\chi_{\mathtt{MM}})}/_{pk_s,pk_{\mathtt{MM}}}\right\}$$

To define of σ and θ we introduce the following shorthand for messages.

$ecert(\hat{\mathbf{t}}, x) = \{\langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle\}_{\mathtt{h}(\phi(\hat{\mathbf{t}},x))}$

$emcert(a, c, x) = \{\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}))\rangle\}_{\mathtt{h}(\phi(a\cdot c,x))}$

$etx(t, tx, x) = \{\langle tx, \perp\rangle\}_{\mathtt{h}(\phi(t,x))}$

$etxpin(t, tx, x, \text{uPIN}) = \{\langle tx, \text{uPIN}\rangle\}_{\mathtt{h}(\phi(t,x))}$

Let $\mathsf{AC} = \langle a, \text{PAN}, \mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(a\cdot c, x)), z))\rangle$, $\mathsf{AC}^{\mathtt{ok}} = \langle a, \text{PAN}, \mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(a\cdot c, x)), z)), \mathtt{ok}\rangle$

$\mathsf{AC}^{\perp} = \langle a, \text{PAN}, \mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(a\cdot c, x)), z)), \perp\rangle$, $\widehat{\mathbf{kbt}} = \mathtt{p}_3(\mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(a\cdot c, x)), y)))$ in

$eaclo(a, c, mk, \text{PAN}, x, y, z) = \{\{\langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk\rangle)\rangle\}_{\mathtt{h}(\phi(a\cdot c, \widehat{\mathbf{kbt}}))}\}_{\mathtt{h}(\phi(a\cdot c,x))}$

$eachi(a, c, mk, \text{PAN}, x, y, z) = \{\langle\{\langle \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\langle \mathsf{AC}^{\mathtt{ok}}, mk\rangle)\rangle\}_{\mathtt{h}(\phi(a\cdot c, \widehat{\mathbf{kbt}}))}\rangle, \mathtt{ok}\}_{\mathtt{h}(\phi(a\cdot c,x))}$

$eacfail(a, c, mk, \text{PAN}, x, y, z) = \{\langle\{\langle \mathsf{AC}^{\perp}, \mathtt{h}\Big(\langle \mathsf{AC}^{\perp}, mk\rangle\Big)\rangle\}_{\mathtt{h}(\phi(a\cdot c, \widehat{\mathbf{kbt}}))}\rangle, \perp\}_{\mathtt{h}(\phi(a\cdot c,x))}$

To present σ and θ in the next two pages we introduce the index function ind : $\{\sigma, \theta\} \to \mathcal{D}$ defined for a substitution $\rho \in \{\sigma, \theta\}$ as follows: $ind(\sigma) = i$, $ind(\theta) = j$.

The aliases for the messages output in session i, and available to the attacker are as follows. Terminal's, card's, and bank's channels are labelled as cht_i, chc_i, and chb_i respectively. Terminal's messages are labelled as ua_i, ub_i, uc_i, ud_i, and ue_i. Card's messages as va_i, vb_i, vc_i. Bank's reply as wa_i.

We start with a natural freshness conditions, i.e. that message labels from $\mathrm{dom}(\sigma)$, $\mathrm{dom}(\theta)$ cannot refer to neither bound nor free names.

for any $i, k \in \mathcal{D} \cup \mathcal{F} \cup \mathcal{G}, l \in \{1 \dots 3\}, m \in \{1 \dots 4\}$
$pk_s, pk_{\mathtt{MM}}, cht_i, chc_i, chb_i, ua_i, va_i, ub_i, vb_i, uc_i, vc_i, ud_i, wa_i, ue_i, X_i^l, Y_i^1, Z_i^m$ #
$FV \cup \{user, s, si, \chi_{\mathtt{MM}}, b_t, \mathrm{PIN}_k, mk_k, c_k, \dot{ch}_k, a_k, ch_k, \ddot{ch}_k, t_k, \mathsf{TX}_k\}$
where $FV = \{card, term, \mathtt{ok}, \bot, \mathtt{accept}, \mathtt{auth}, \mathtt{lo}, \mathtt{hi}\}$

$\mathrm{fv}(X_i^1)$ # $\{va_i, vb_i, vc_i\}$
$\mathrm{fv}(X_i^2)$ # $\{vb_i, vc_i\}$
$\mathrm{fv}(X_i^3)$ # $\{vc_i\}$

$\mathrm{fv}(Z_i^1)$ # $\{ub_i, uc_i, ud_i, ue_i\}$
$\mathrm{fv}(Z_i^2)$ # $\{uc_i, ud_i, ue_i\}$
$\mathrm{fv}(Z_i^3)$ # $\{ud_i, ue_i\}$
$\mathrm{fv}(Z_i^4)$ # $\{ue_i\}$

$\mathrm{fv}(Y_i^1)$ # $\{wa_i\}$

$cht_i\rho = \ddot{ch}_i$ if $i \in \mathcal{FG}$

$chc_i\rho = \dot{ch}_i$ if $i \in \mathcal{D}$

$chb_i\rho = \dddot{ch}_i$ if $i \in \mathcal{FM}$

$ua_i\rho = \phi(t_i, \mathfrak{g})$ if $i \in \bigcup_{l=2}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=2}^{11} \gamma_l^{\mathtt{of}}$ or $\bigcup_{l=2}^{10} \gamma_l^{\mathtt{lo}}$

$va_i\rho = \phi\left(a_i, \phi\left(c_{ind(\rho)}, \mathfrak{g}\right)\right)$ if $i \in \bigcup_{l=3}^{7} \alpha_l$ (and, if $\rho = \theta$, $i \in \lambda_j$)

$ub_i\rho = ecert(t_i, Z_1^i\rho)$ if $i \in \bigcup_{l=4}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=4}^{11} \gamma_l^{\mathtt{of}}$ or $\bigcup_{l=4}^{10} \gamma_l^{\mathtt{lo}}$

$vb_i\rho = emcert(a_i, c_{ind(\rho)}, X_1^i\rho)$ if $i \in \bigcup_{l=5}^{7} \alpha_l$ (and, if $\rho = \theta$, $i \in \lambda_j$) and
$\mathtt{check}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right), \mathtt{pk}(s)\right) = \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)$ and
$\mathtt{p}_1\left(\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)\right) = \mathtt{MM}$

$uc_i\rho = etxpin(t_i, tx_i, Z_1^i\rho, \text{uPIN})$ if $i \in \bigcup_{l=7}^{11} \gamma_l^{\texttt{of}}$ and
$\texttt{vcheck}\big(\texttt{p}_2\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big), \texttt{vpk}(\chi_{\texttt{MM}})\big) = \texttt{p}_1\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big)$ and
$\texttt{p}_1\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big) = Z_1^i\rho$

$= etx(t_i, tx_i, Z_1^i\rho)$ if $i \in \bigcup_{l=7}^{11} \gamma_l^{\texttt{on}}$ or $\bigcup_{l=6}^{10} \gamma_l^{\texttt{lo}}$ and
$\texttt{vcheck}\big(\texttt{p}_2\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big), pk_{\texttt{MM}}\big) = \texttt{p}_1\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big)$ and
$\texttt{p}_1\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\rho\big)\big), Z_i^2\rho\big)\big) = Z_1^i\rho$

$vc_i\rho = eaclo(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho)$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\texttt{p}_2\Big(\texttt{dec}\Big(\texttt{h}\Big(\phi\Big(a_i \cdot c_{ind(\rho)}, X_i^1\rho\Big)\Big), X_i^3\rho\Big)\Big) = \perp$

$= eachi(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho)$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\texttt{p}_2\Big(\texttt{dec}\Big(\texttt{h}\Big(\phi\Big(a_i \cdot c_{ind(\rho)}, X_i^1\rho\Big)\Big), X_i^3\rho\Big)\Big) = \text{PIN}_i$

$= eacfail(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho)$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and else

$ud_i\rho = \{\langle \mathsf{TX}_i, Z_1^i\rho, \texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_1^i\rho\big)\big), Z_3^i\rho\big), \text{uPIN}\rangle\}_{kbt_i}$ if $i \in \bigcup_{l=9}^{11} \gamma_l^{\texttt{on}}$

$= \{\langle \mathsf{TX}_i, Z_1^i\rho, \texttt{p}_1\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_1^i\rho\big)\big), Z_3^i\rho\big)\big), \perp\rangle\}_{kbt_i}$ if $i \in \gamma_{11}^{\texttt{of}}$

$= \{\langle \mathsf{TX}_i, Z_1^i\rho, \texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_1^i\rho\big)\big), Z_3^i\rho\big), \perp\rangle\}_{kbt_i}$ if $i \in \bigcup_{l=8}^{10} \gamma_l^{\texttt{lo}}$

$wa_i\rho = \{\langle \texttt{p}_1(dy_i), \texttt{accept}\rangle\}_{kbt_i}$ let $dy_i = \texttt{dec}\big(kbt_i, Y_i^1\rho\big)$ and $\langle \text{PIN}_j, mk_j, \phi\big(c_j, \mathfrak{g}\big)\rangle = \text{DB}$
if $i \in \beta_4$ and $\exists j$, s.t. $j \in \alpha_7$ and
$\texttt{h}\big(\langle \texttt{p}_1(\texttt{dec}(\texttt{h}(\phi(b_t, \texttt{p}_2(dy_i))), \texttt{p}_3(dy_i))), mk_j\rangle\big) = \texttt{p}_2(\texttt{dec}(\texttt{h}(\phi(b_t, \texttt{p}_2(dy_i))), \texttt{p}_3(dy_i)))$ and
$\texttt{p}_3(\texttt{p}_1(\texttt{dec}(\texttt{h}(\phi(b_t, \texttt{p}_2(dy_i))), \texttt{p}_3(dy_i)))) = \texttt{p}_1(dy_i)$ and
$\phi\big(\texttt{p}_1(\texttt{p}_1(\texttt{dec}(\texttt{h}(\phi(b_t, \texttt{p}_2(dy_i))), \texttt{p}_3(dy_i)))), \phi\big(c_j, \mathfrak{g}\big)\big) = \texttt{p}_2(dy_i)$ and
$\big(\texttt{p}_2(\texttt{p}_1(dy_i)) = \texttt{lo}$ or $\texttt{p}_2(\texttt{p}_1(dy_i)) = \texttt{hi}$ and $\texttt{p}_4(\texttt{p}_1(\texttt{dec}(\texttt{h}(\phi(b_t, \texttt{p}_2(dy_i))), \texttt{p}_3(dy_i)))) = \texttt{ok}$ or else if $\texttt{p}_4(dy_i) = \text{PIN}_j\big)$

$ue_i\rho = \texttt{auth}$ if $i \in \gamma_9^{\texttt{of}} \cup \gamma_{11}^{\texttt{of}}$ and $\texttt{p}_2\big(\texttt{dec}\big(\texttt{h}\big(\phi\big(t_i, Z_i^1\big)\big), Z_i^3\big)\big) = \texttt{ok}$ or
if $i \in \gamma_{11}^{\texttt{on}}$ or $i \in \gamma_{10}^{\texttt{lo}}$ and $\texttt{p}_1\big(\texttt{dec}\big(kbt_i, Z_4^i\big)\big) = \mathsf{TX}_i$ and $\texttt{p}_2\big(\texttt{dec}\big(kbt_i, Z_4^i\big)\big) = \texttt{accept}$

Bisimulation. Now, when the relation is defined, we can start consider all possible moves each side can make. Since we have defined $\mathfrak{R}$ as a symmetric relation, we consider only the cases when the spec process starts first.

Case 1. $\overline{out}(pk_s)$, $UP_{\text{spec}}\ \mathfrak{R}\ UP_{\text{impl}}$.

The process UP_{spec} can make the transition $\overline{out}(pk_s)$ to the state UP^1_{spec}. There is a state UP^1_{impl} to which the process UP_{impl} can make the transition $\overline{out}(pk_s)$. By the definition of $\mathfrak{R}$ we have $UP^1_{\text{spec}}\ \mathfrak{R}\ UP^1_{\text{impl}}$.

Case 2. $\overline{out}(pk_{\texttt{MM}})$, $UP^1_{\text{spec}}\ \mathfrak{R}\ UP^1_{\text{impl}}$.

Identical to Case 1.

Case 3. $\overline{card}(chc_{D+1})$.

From now on, we will only track the effect of the transition on the parameters defining the state, hence the parameters not affected by the transition are omitted.

The spec process either creates a new card and outputs the channel, hence transits to $\alpha_1 \cup \{D+1\}$ or outputs a channel for the waiting card evolving to $K-1, \alpha_1 \cup \{D+1\}$. The impl process can match by either creating a new card and announcing the channel $\alpha_1 \cup \{D+1\}, \Lambda \cup \{D+1\}$, starting new session for the existing card h making the transition to $\alpha_1 \cup \{D+1\}, \lambda_h \cup \{D+1\}$ or outputting the channel for the waiting card h evolving to $K_h - 1, \alpha_1 \cup \{D+1\}, \lambda_h \cup \{D+1\}$. In either case the resulting states are related by $\mathfrak{R}$.

Case 4. $\overline{term}(cht_{F+G+1})$.

Either the spec process starts a new `on`, `of` or `lo` terminal session by also creating a symmetric bank-terminal symmetric key transiting to respectively $\gamma_1^{\texttt{on}} \cup \{F+G+1\}$, $\gamma_1^{\texttt{of}} \cup \{F+G+1\}$ or $\gamma_1^{\texttt{lo}} \cup \{F+G+1\}$. Or the spec process starts a new `on`, `of` or `lo` terminal session for the existing bank-terminal key transiting to respectively $F+1, \gamma_1^{\texttt{on}} \cup \{F+G+1\}$, $F+1, \gamma_1^{\texttt{of}} \cup \{F+G+1\}$ or $F+1, \gamma_1^{\texttt{lo}} \cup \{F+G+1\}$. The impl process can always match and the resulting states are related by $\mathfrak{R}$.

Case 5. $\overline{bank}(chb_{F+G+1})$.

Identical to Case 4. Notice that in cases 3-5, as new card terminal or bank sessions started, the input matrices X, Y and Z also grow by one row to accommodate future inputs.

Case 6. $\overline{cht_i}(ua_i)$, $i \in \gamma_1^{\texttt{on}}$, $i \in \gamma_1^{\texttt{of}}$ or $i \in \gamma_1^{\texttt{lo}}$.

The spec process moves to the respective state $\gamma_1^{\texttt{on}} \setminus \{i\}, \gamma_2^{\texttt{on}} \cup \{i\}$, $\gamma_1^{\texttt{of}} \setminus \{i\}, \gamma_2^{\texttt{of}} \cup \{i\}$ or $\gamma_1^{\texttt{lo}} \setminus \{i\}, \gamma_2^{\texttt{lo}} \cup \{i\}$. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 7. $chc_i\ X_i^1$, $i \in \alpha_1$.

The spec process moves to the respective state $\alpha_1 \setminus \{i\}, \alpha_2 \cup \{i\}$ where the $(i,1)$ element of the matrix X is replaced by X_i^1. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 8. $\overline{chc_i}(va_i)$, $i \in \alpha_2$.

The spec process moves to the respective state $\alpha_2 \setminus \{i\}, \alpha_3 \cup \{i\}$. The process on the right can always match, it is parametrised by the same partition, hence the resulting states are related by $\mathfrak{R}$.

Case 9. $cht_i\ Z_i^1$, $i \in \gamma_2^{\texttt{on}}$, $i \in \gamma_2^{\texttt{of}}$ or $i \in \gamma_2^{\texttt{lo}}$.

The spec process moves to the respective state $\gamma_2^{\texttt{on}} \setminus \{i\}, \gamma_3^{\texttt{on}} \cup \{i\}$, $\gamma_2^{\texttt{of}} \setminus \{i\}, \gamma_3^{\texttt{of}} \cup \{i\}$ or $\gamma_2^{\texttt{lo}} \setminus \{i\}, \gamma_3^{\texttt{lo}} \cup \{i\}$ where the $(i,1)$ element of the matrix Z is replaced by Z_i^1.

The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 10. $\overline{cht_i}(ub_i)$, $i \in \gamma_3^{\mathtt{on}}$, $i \in \gamma_3^{\mathtt{of}}$ or $i \in \gamma_3^{\mathtt{lo}}$.

The spec process moves to the respective state $\gamma_3^{\mathtt{on}} \setminus \{i\}, \gamma_4^{\mathtt{on}} \cup \{i\}$, $\gamma_3^{\mathtt{of}} \setminus \{i\}, \gamma_4^{\mathtt{of}} \cup \{i\}$ or $\gamma_3^{\mathtt{lo}} \setminus \{i\}, \gamma_4^{\mathtt{lo}} \cup \{i\}$. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 11. $chc_i\, X_i^2$, $i \in \alpha_3$.

The spec process moves to the state $\alpha_3 \setminus \{i\}, \alpha_4 \cup \{i\}$ where the $(i,2)$ element of the matrix X is replaced by X_i^2. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 12. $\overline{chc_i}(vb_i)$, $i \in \alpha_4$ and the conditions for $vb_i\sigma$ in the definition of σ in Fig. **??** are satisfied given the inputs (X, Y, Z).

Let $\alpha_4 \setminus \{i\}, \alpha_5 \cup \{i\}$ define the resulting state in which the spec process can make the transition. Consider the second condition. We can rewrite this condition as $X_i^2\sigma = \{\langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle\}_{\mathtt{h}(\phi(a_i \cdot c_i, X_i^1\sigma))}$, where M_1 and M_2 are arbitrary messages. As $\mathtt{MM}$ is a global constant and M_1 and M_2 are arbitrary attacker's inputs, the term $\langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle$ can always be produced. Since at the point of input of X_i^2 the only message on the network that refers to (some multiple) of a_i is $\phi(a_i, \phi(c_i, \mathfrak{g}))$ available through the alias va_i, the key is either of the form $\mathtt{h}(va_i)$ or $\mathtt{h}(\phi(\hat{t}, va_i))$ for some $\hat{t}$ before σ is applied. In either case we then conclude that $X_i^2\theta = \{\langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle\}_{\mathtt{h}(\phi(a_i \cdot c_h, X_i^1\theta))}$ holds since $va_i\theta = \phi(a_i, \phi(c_h, \mathfrak{g}))$ for some card h and we have established that the second condition holds also in the impl case. The case when the spec process stats first is similar. Now notice, that from the second equation it follows that $\mathtt{dec}(\mathtt{h}(\phi(a_i \cdot c_i, X_1^i\sigma)), X_i^2\sigma) = \mathtt{dec}(\mathtt{h}(\phi(a_i \cdot c_h, X_1^i\sigma)), X_i^2\theta) = \langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle$, and the first condition becomes as follows $\mathtt{check}(\mathtt{pk}(s), \mathtt{p}_2(\langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle)) = \mathtt{p}_1(\langle\langle \mathtt{MM}, M_1\rangle, M_2\rangle)$ which is independent of σ and θ and trivially holds in the impl case. We conclude that the impl process can always match by transiting to the state parametrised by $\alpha_4 \setminus \{i\}, \alpha_5 \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$.

Case 13. $cht_i\, Z_2^i$, $i \in \gamma_4^{\mathtt{on}}$, $i \in \gamma_4^{\mathtt{of}}$ or $i \in \gamma_4^{\mathtt{lo}}$

The spec process moves to the state $\gamma_4^{\mathtt{on}} \setminus \{i\}, \gamma_5^{\mathtt{on}} \cup \{i\}$, $\gamma_4^{\mathtt{of}} \setminus \{i\}, \gamma_5^{\mathtt{of}} \cup \{i\}$ or $\gamma_4^{\mathtt{lo}} \setminus \{i\}, \gamma_5^{\mathtt{lo}} \cup \{i\}$ and the element $(i,2)$ of the matrix Z is replaced by Z_i^2. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 14. τ, $i \in \gamma_5^{\mathtt{on}}$ or $i \in \gamma_5^{\mathtt{of}}$ and the conditions for $uc_i\sigma$ in the definition of σ are satisfied given the inputs (X, Y, Z).

Let $\gamma_5^{\mathtt{on}} \setminus \{i\}, \gamma_6^{\mathtt{on}} \cup \{i\}$ or $\gamma_5^{\mathtt{of}} \setminus \{i\}, \gamma_6^{\mathtt{of}} \cup \{i\}$ define the resulting state in which the spec process can make the transition. Similarly to Case 12 consider the second condition. We can rewrite it as $Z_i^2\sigma = \{\langle Z_1^i\sigma, M\rangle\}_{\mathtt{h}(\phi(t_i, Z_i^1\sigma))}$ for some M. Since at the point of input of X_i^2 the only message on the network that refers to a multiple of t_i is $\phi(t_i, \mathfrak{g})$ available through the alias ua_i, the key is either of the form $\mathtt{h}(ua_i)$ or $\mathtt{h}(\phi(\hat{t}, ua_i))$ for some $\hat{t}$ before σ is applied. In either case we conclude that $Z_i^2\theta = \{\langle Z_1^i\theta, M\rangle\}_{\mathtt{h}(\phi(t_i, Z_i^1\theta))}$ holds since $ua_i\theta = \phi(t_i, \mathfrak{g})$, so we have established that the second condition holds also in the impl case. The case when the spec process stats first is similar. Now notice that if follows from the second equation that $\mathtt{dec}(\mathtt{h}(\phi(t_i, Z_i^1\sigma)), Z_i^2\sigma) = \langle Z_i^1\sigma, M\rangle$ and the first condition becomes

$\mathtt{vcheck}(M, \mathtt{vpk}(\chi_{\mathtt{MM}})) = Z_i^1\sigma$, i.e. the input $Z_i^1\sigma$ is signed with the signing key $\chi_{\mathtt{MM}}$, hence can only be the multiple of some $\phi(c_l, \mathfrak{g})$ for some l. The only message on the network of this form is $\phi(a_l, \phi(c_l, \mathfrak{g}))$ available through the alias va_l, hence the input Z_i^1 is either va_l or $\phi(\hat{s}, va_l)$ which, under θ also give signed inputs in the impl case. We conclude that the impl process can always match by transiting to the state parametrised by $\gamma_5^{\mathtt{on}} \setminus \{i\}, \gamma_6^{\mathtt{on}} \cup \{i\}$ or $\gamma_5^{\mathtt{of}} \setminus \{i\}, \gamma_6^{\mathtt{of}} \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$. Notice that there is no PIN check at this point and either "right" or "wrong" PINs are always available, hence whenever the user enters a PIN on the spec/impl side, the same type of PIN can always be entered on the impl/spec side respectively.

Case 15. $\overline{cht_i}(uc_i)$, $i \in \gamma_6^{\mathtt{on}}$, $i \in \gamma_6^{\mathtt{of}}$, or $i \in \gamma_5^{\mathtt{lo}}$ *and* the conditions for $uc_i\sigma$ in the definition of σ are satisfied given the inputs (X, Y, Z).

The spec process moves to the respective state $\gamma_6^{\mathtt{on}} \setminus \{i\}, \gamma_7^{\mathtt{on}} \cup \{i\}$, $\gamma_6^{\mathtt{of}} \setminus \{i\}, \gamma_7^{\mathtt{of}} \cup \{i\}$ or $\gamma_5^{\mathtt{lo}} \setminus \{i\}, \gamma_6^{\mathtt{lo}} \cup \{i\}$. Notice that the `lo` case is similar to Case 14. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 16. $chc_i\ X_i^3$, $i \in \alpha_5$.

The spec process moves to the state $\alpha_5 \setminus \{i\}, \alpha_6 \cup \{i\}$ where the $(i, 3)$ element of the matrix X is replaced by X_i^3. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 17. $\overline{chc_i}(vc_i)$, $i \in \alpha_6$ and the conditions for $vc_i\sigma$ in the definition of σ are satisfied given the inputs (X, Y, Z).

Let $\alpha_6 \setminus \{i\}, \alpha_7 \cup \{i\}$ define the resulting state in which the spec process can make the transition. Consider either the case when either the $\perp$ or PIN_i is the element of the received input. We can rewrite these guards as follows $X_i^3\sigma = \{\langle M, N\rangle\}_{\mathtt{h}(\phi(a_i \cdot c_i, X_i^1\sigma))}$ where M is arbitrary, $N \in \{\perp, \mathrm{PIN}_i\}$ and apply the argument from Case 12. The `else` branch is identical to Case 19.4. We conclude that the impl process can always match by transiting to the state parametrised by $\alpha_6 \setminus \{i\}, \alpha_7 \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$.

Case 18. $cht_i\ Z_3^i$, $i \in \gamma_7^{\mathtt{on}}$, $i \in \gamma_7^{\mathtt{of}}$ or $i \in \gamma_6^{\mathtt{lo}}$.

The spec process moves to the state $\gamma_7^{\mathtt{on}} \setminus \{i\}, \gamma_8^{\mathtt{on}} \cup \{i\}$ or $\gamma_7^{\mathtt{of}} \setminus \{i\}, \gamma_8^{\mathtt{of}} \cup \{i\}$ or $\gamma_6^{\mathtt{lo}} \setminus \{i\}, \gamma_7^{\mathtt{lo}} \cup \{i\}$ and the element $(i, 3)$ of the matrix Z is replaced by Z_i^3. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 19.1. $\overline{cht_i}(ud_i)$, $i \in \gamma_8^{\mathtt{on}}$.

The spec process moves to the state $\gamma_8^{\mathtt{on}} \setminus \{i\}, \gamma_9^{\mathtt{on}} \cup \{i\}$ as online terminal simply adds the entered PIN and sends the cryptogram to the bank. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 19.2."Right" PIN. $\overline{cht_i}(ue_i)$, $i \in \gamma_8^{\mathtt{of}}$ and the condition for $ue_i\sigma$ that the card replied affirmatively in the definition of σ is satisfied given the inputs (X, Y, Z).

Let $\gamma_8^{\mathtt{of}} \setminus \{i\}, \gamma_9^{\mathtt{of}} \cup \{i\}$ define the resulting state in which the spec process can make the transition. The check that the condition holds in the impl case for the substitution θ is identical to Case 12. We conclude that the impl process can always match by transiting to the state parametrised by $\gamma_8^{\mathtt{of}} \setminus \{i\}, \gamma_9^{\mathtt{of}} \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$.

Case 19.3. "Right" PIN. $\overline{cht_i}(ud_i)$, $i \in \gamma_9^{\mathtt{of}}$.

Let $\gamma_9^{\mathtt{of}} \setminus \{i\}, \gamma_{11}^{\mathtt{of}} \cup \{i\}$ define the resulting state in which the spec process can make the transition as offline terminal simply forwards the cryptogram to the bank. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 19.4. "Wrong" PIN. $\overline{cht_i}(ud_i)$, $i \in \gamma_1^{\mathtt{of}}0$ and the condition for $ue_i\sigma$ that the card replied affirmatively in the definition of σ does *not hold* given the inputs (X, Y, Z).

Let $\gamma_1^{\mathtt{of}}0 \setminus \{i\}, \gamma_{11}^{\mathtt{of}} \cup \{i\}$ define the resulting state in which the spec process can make the transition. By the Def. 2 in [HMY21] the inequality holds whenever there is no unifying the left and the right part substitution ρ that cannot refer private values in its domain or range. Consider the opposite for impl, i.e. that there is a unifying substitution ρ (in case $D + K > H$ w.l.o.g. we assume that ρ does not refer PIN_i, mk_i, c_i, PAN_i, $i \in \{H+1, \ldots, D+K\}$), i.e. $\mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(t_i\rho, Z_i^1\theta\rho\big)\big), Z_i^3\theta\rho\big)\big) = \mathtt{ok}\rho$ or, $\mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(t_i, Z_i^1\rho\theta\big)\big), Z_i^3\rho\theta\big)\big) = \mathtt{ok}\rho$. Then applying the argument from Case 12 we conclude that ρ also unifies the condition for the spec state, which contradicts the initial condition. We conclude that the impl process can always match by transiting to the state parametrised by $\gamma_1^{\mathtt{of}}0 \setminus \{i\}, \gamma_{11}^{\mathtt{of}} \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$.

Case 19.5. $\overline{cht_i}(ud_i)$, $i \in \gamma_7^{\mathtt{lo}}$.

Identical to Case 19.1, the resulting state where the spec process can make the transition is $\gamma_7^{\mathtt{lo}} \setminus \{i\}, \gamma_8^{\mathtt{of}} \cup \{i\}$. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 20. $chb_i\, Y_1^i$, $i \in \beta_1$.

The spec process moves to the state $\beta_1 \setminus \{i\}, \beta_2 \cup \{i\}$ and the element $(i, 1)$ of the matrix Y is replaced by Y_i^1. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 21. τ, $i \in \beta_2$.

Let $\beta_2 \setminus \{i\}, \beta_3 \cup \{i\}$ define the resulting state in which the spec process can make the transition. Notice that the database process $\overline{\langle si, \mathrm{PAN}_j\rangle}\big\langle\big\langle \mathrm{PIN}_j, mk_j, \phi\big(c_j, \mathfrak{g}\big)\big\rangle\big\rangle$ can only be accessed in case that there is a terminal in session $i \in \gamma_9^{on} \cup \gamma_{11}^{of} \cup \gamma_8^{lo}$ (and using kbt_i) and a corresponding card's session j where the cryptogram containing legitimate PAN_j has been sent to the terminal. Also, since the card's data retrieved by the card from the database is obtained privately using the shared secret si, and the legitimate PAN_j it is always "right" in contrast to the PIN entered to the terminal (the "wrong" PIN can be entered). Hence the subsequent integrity checks for the cryptogram always using the correct data for the received cryptogram. The impl process can always match by transiting to the state parametrised by $\beta_2 \setminus \{i\}, \beta_3 \cup \{i\}$ and the resulting states are related by $\mathfrak{R}$.

Case 22. $\overline{chb_i}(wa_i)$, $i \in \beta_3$ and the conditions for $wa_i\sigma$ hold given the inputs (X, Y, Z).

Let $\beta_3 \setminus \{i\}, \beta_4 \cup \{i\}$ define the resulting state in which the spec process can make the transition. Since the communication between the terminal and the bank is private as discussed in Case 20, the conditions for $wa_i\theta$ also hold since they depend only on whether the input $Y_i^1\theta$ can be successfully decrypted. Also notice that since high-value terminals had infinite supply of both right and wrong PINs,

the respective transactions are either accepted or declined (by not passing the PIN guard) simultaneously by spec and impl processes. The impl process can always match by transiting to the state parametrised by $\overline{chb_i}(wa_i)$, $i \in \beta_3$ and the resulting states are related by $\mathfrak{R}$.

Case 23. $cht_i\, Z_i^4$, $i \in \gamma_9^{\mathtt{on}}$ or $i \in \gamma_8^{\mathtt{lo}}$.

The spec process moves to the state $\gamma_9^{\mathtt{on}} \setminus \{i\}, \gamma_{10}^{\mathtt{on}} \cup \{i\}$ or $\gamma_8^{\mathtt{of}} \setminus \{i\}, \gamma_9^{\mathtt{of}} \cup \{i\}$ and the element $(i, 4)$ of the matrix Z is replaced by Z_i^4. The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Case 24. $\overline{cht_i}(ue_i)$, $i \in \gamma_{10}^{\mathtt{on}}$ or $i \in \gamma_9^{\mathtt{lo}}$ and the conditions for $ue_i\sigma$ hold given the inputs (X, Y, Z).

Identical to Case 22 since the communication between the bank and the terminal is private. The resulting state where the spec process can make the transition is $\gamma_{10}^{\mathtt{on}} \setminus \{i\}, \gamma_{11}^{\mathtt{on}} \cup \{i\}$ or $\gamma_9^{\mathtt{lo}} \setminus \{i\}, \gamma_{10}^{\mathtt{lo}} \cup \{i\}$ The impl process can always match by transiting to the state parametrised identically, hence the resulting states are related by $\mathfrak{R}$.

Openness. Intuitively, the relation is open if an attacker with the power to manipulate free variables by applying a substitution φ that cannot refer to the variables bound by ν, and the message aliases from the $\mathrm{dom}(\sigma) = \mathrm{dom}(\theta)$, and with the power to extend the environment by declaring some free variables private (and out messages referring to them on the network) cannot reach the pair of states that are not related as formally stated in Def. 4 from Sec. 2.2.2.

The relation $\mathfrak{R}$ is open by definition, and it is straightforward that manipulating free variables that include *out*, *card*, *term*, *bank*, `MM`, `auth`, $\perp$, `ok`, `lo`, `hi`, and, possibly, free variables that inputs may contain, introduce neither new transitions not considered above nor affects static equivalence (see below) as these variables distributed symmetrically in the related states.

Static equivalence. We proceed by proving that any two states related by $\mathfrak{R}$ are statically equivalent. Static equivalence trivially holds when frames are both empty or both are σ_0, hence we proceed with a general case in a separate Lemma 2 below. □

To prove static equivalence, we follow the steps of Lemma 1 in Sec. 3.5.2, i.e. we firstly identify building blocks for messages available to an attacker in a unique form up to multiplication and then conduct the proof by induction on the structure of $N\sigma$ from the Def. 3 of static equivalence. Since the message theory in Fig. 4.2 for UTX differs from the one in the previous chapter for UBDH by the presence of a generic signature scheme, we must update the definitions for E-weak normal form and E-normalisation of the frame. Both notions are defined and presented in full below.

Definition 15. *(E-weak normal form) The E-weak normal $M\downarrow$ of a message term M is defined inductively on the structure of M:*

- *$M = \mathfrak{g}$ or M is a variable, then $M\downarrow = M$.*
- *$M = M_1 \cdot M_2$, then $M\downarrow = M_1\downarrow \cdot M_2\downarrow$.*

- $M = \phi(M_1, M_2)$, *then* $M\lfloor = \phi(M_1\lfloor, M_2\lfloor)$ *if* $M_2\lfloor$ *is* ϕ*-atomic. Otherwise* $M\lfloor = \phi(M_1\lfloor \cdot M_2'\lfloor, M_2''\lfloor)$, *where* $M_2 =_E \phi(M_2', M_2'')$ *and* $M_2''\lfloor$ *is* ϕ*-atomic.*
- $M = \{M_1\}_{M_2}$, *then* $M\lfloor = \{M_1\lfloor\}_{M_2\lfloor}$.
- $M = \langle M_1, \ldots, M_n\rangle$, *then* $M\lfloor = \langle M_1\lfloor, \ldots, M_2\lfloor\rangle$.
- $M = \mathtt{h}(M_1)$, $M = \mathtt{pk}(M_1)$, *or* $M = \mathtt{vpk}(M_1)$ *then* $M\lfloor = \mathtt{h}(M_1\lfloor)$, $M\lfloor = \mathtt{pk}(M_1\lfloor)$, *or* $M\lfloor = \mathtt{vpk}(M_1\lfloor)$.
- $M = \mathtt{sig}(M_1, M_2)$, *then* $M\lfloor = \mathtt{sig}(M_1\lfloor, M_2\lfloor)$.
- $M = \mathtt{vsig}(M_1, M_2)$, *then* $M\lfloor = \mathtt{vsig}(M_1\lfloor, M_2\lfloor)$ *if* $M_1\lfloor$ *is* ϕ*-atomic. Otherwise* $M\lfloor = \phi(M_1'\lfloor, \mathtt{vsig}(M_1''\lfloor, M_2\lfloor))$, *where* $M_1 = \phi(M_1', M_1'')$ *and* $M_1''\lfloor$ *is* ϕ*-atomic.*
- $M = \mathtt{check}(\mathtt{sig}(M_1, M_2), \mathtt{pk}(M_2))$, *then* $M\lfloor = M_1\lfloor$.
- $M = \mathtt{vcheck}(\mathtt{vsig}(M_1, M_2), \mathtt{vpk}(M_2))$, *then* $M\lfloor = M_1\lfloor$.
- $M = \mathtt{p}_k(\langle M_1, \ldots, M_n\rangle)$ *then* $M\lfloor = M_k\lfloor$.
- $M = \mathtt{dec}(\{M_1\}_{M_2}, M_2)$, *then* $M\lfloor = M_1\lfloor$.
- *Otherwise* $M\lfloor = M$.

The normalisation of a frame $\nu\vec{z}.(\rho \mid \_)$ with respect to the equational theory E capturing the saturation of the range of ρ with weak normal forms of messages that have recipes under $\nu\vec{z}.(\rho \mid \_)$ and is defined by the following procedure.

1. $u\rho = M$ for any $u \in \mathrm{dom}(\rho)$ is replaced by $u\rho = M\lfloor$.
2. If $u\rho = K_1 \cdot K_2$ and there is a recipe M_1 for an immediate m-factor K_1, then $M_1\rho$ is added to the normalisation. If there is a recipe M_2 for an immediate m-factor K_2, then $M_2\rho$ is also added to the normalisation.
3. If $u\rho = \langle K_1, \ldots, K_n\rangle$, then $u\rho$ is replaced by $\mathtt{p_i}(u\rho) = K_i$, $1 \leq i \leq n$.
4. If $u\rho = \{K_1\}_{K_2}$ and there is a recipe M_2 for K_2, then $u\rho$ is replaced by $\mathtt{dec}(u, M_2)\,\rho = K_1$.
5. If $u\rho = \mathtt{sig}(N_1, N_2)$ and there is a recipe M_2 for N_2, then $u\rho$ is replaced by $\mathtt{check}(u, \mathtt{pk}(M_2))\,\rho = N_1$.
6. If $u\rho = \mathtt{sig}(N_1, N_2)$ and there is a recipe M_2 for $\mathtt{pk}(N_2)$, then $\mathtt{check}(u, M_2)\,\rho = N_1$ is added to the normalisation.
7. If $u\rho = \mathtt{vsig}(N_1, N_2)$ and there is a recipe M_2 for N_2, then $u\rho$ is replaced by $\mathtt{vcheck}(u, \mathtt{pk}(M_2))\,\rho = N_1$.
8. If $u\rho = \mathtt{vsig}(N_1, N_2)$ and there is a recipe M_2 for $\mathtt{vpk}(N_2)$, then $\mathtt{vcheck}(u, M_2)\,\rho = N_1$ is added to the normalisation.

Now we can conclude the proof of Theorem 8 with the following.

Lemma 2. *$(K, F, A, \Gamma, B)_{spec}(X, Y, Z)$ is statically eq. to $(\vec{K}, F, A, \Gamma, B, \Lambda)_{impl}(X, Y, Z)$.*

Proof. Firstly we define the normalisations $\sigma\downarrow_E^{\vec{x}}$ and $\theta\downarrow_E^{\vec{y}}$ of the frames $\nu\vec{x}.(\sigma \mid \_)$ and $\nu\vec{y}.(\theta \mid \_)$, where $\vec{x}$ and $\vec{y}$ define the sets of bound names in $(K, F, A, \Gamma, B)_{\text{spec}}(X, Y, Z)$ and $(\vec{K}, F, A, \Gamma, B, \Lambda)_{\text{impl}}(X, Y, Z)$ respectively. To define normalisations we, again, use the index function $ind : \{\sigma, \theta\} \to \mathcal{D}$, s.t. $ind(\sigma) = i$, $ind(\theta) = j$. We also denote the attacker's input in session i as A_i^1, B_i^1.

$cht_i\rho = \ddot{ch}_i$	if $i \in \mathcal{FG}$
$chc_i\rho = \dot{ch}_i$	if $i \in \mathcal{D}$
$chb_i\rho = \dddot{ch}_i$	if $i \in \mathcal{FM}$
$ua_i\rho = \phi(t_i, \mathfrak{g})$	if $i \in \bigcup_{l=2}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=2}^{11} \gamma_l^{\mathtt{of}}$ or $\bigcup_{l=2}^{10} \gamma_l^{\mathtt{lo}}$
$va_i\rho = \phi\left(a_i \cdot c_{ind(\rho)}, \mathfrak{g}\right)$	if $i \in \bigcup_{l=3}^{7} \alpha_l$ (and, if $\rho = \theta$, $i \in \lambda_j$)
$\mathtt{p}_1\left(\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(B_i^1, ua_i\right)\right), ub_i\right)\right)\right)\rho = \mathtt{MM}$	if $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $i \in \bigcup_{l=4}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=4}^{11} \gamma_l^{\mathtt{of}}$ or $\bigcup_{l=4}^{10} \gamma_l^{\mathtt{lo}}$
$\mathtt{p}_2\left(\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(B_i^1, ua_i\right)\right), ub_i\right)\right)\right)\rho = \phi(b_t, \mathfrak{g})$	
$\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(B_i^1, ua_i\right)\right), ub_i\right)\right)\rho = \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)$	
$\mathtt{p}_1\left(\mathtt{check}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(B_i^1, ua_i\right)\right), ub_i\right)\right), pk_s\right)\right)\rho = \mathtt{MM}$	
$\mathtt{p}_2\left(\mathtt{check}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(B_i^1, ua_i\right)\right), ub_i\right)\right), pk_s\right)\right)\rho = \phi(b_t, \mathfrak{g})$	
$ub_i\rho = ecert(t_i, Z_1^i\rho) = \{-\}_-$	if $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $i \in \bigcup_{l=4}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=4}^{11} \gamma_l^{\mathtt{of}}$ or $\bigcup_{l=4}^{10} \gamma_l^{\mathtt{lo}}$
$\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(A_1^i, va_i\right)\right), vb_i\right)\right)\rho = \phi\left(a_i \cdot c_{ind(\rho)}, \mathfrak{g}\right)$	if $X_i^1 = \phi(A_i^1, \mathfrak{g})$ and if $i \in \bigcup_{l=5}^{7} \alpha_l$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\mathtt{check}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right), \mathtt{pk}(s)\right) = \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)$ and $\mathtt{p}_1\left(\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)\right) = \mathtt{MM}$
$\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(A_1^i, va_i\right)\right), vb_i\right)\right)\rho = \phi\left(a_i \cdot c_{ind(\rho)}, \mathtt{vsig}(\mathfrak{g}, \chi_{\mathtt{MM}})\right)$	
$\mathtt{vcheck}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(A_1^i, va_i\right)\right), vb_i\right)\right), pk_{\mathtt{MM}}\right)\rho = \phi\left(a_i \cdot c_{ind(\rho)}, \mathfrak{g}\right)$	

$vb_i\rho = emcert(a_i, c_{ind(\rho)}, X_1^i\rho) = \{-\}_-$ if $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$ and if $i \in \bigcup_{l=5}^{7} \alpha_l$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\mathtt{check}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right), \mathtt{pk}(s)\right) = \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)$ and $\mathtt{p}_1\left(\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^2\rho\right)\right)\right) = \mathtt{MM}$

$uc_i\rho = etxpin(t_i, tx_i, Z_1^i\rho, \text{uPIN}) = \{-\}_-$ if $i \in \bigcup_{l=7}^{11} \gamma_l^{\mathtt{of}}$ and $\mathtt{vcheck}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right), \mathtt{vpk}(\chi_{\mathtt{MM}})\right) = \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right)$ and $\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right) = Z_1^i\rho$

$= etx(t_i, tx_i, Z_1^i\rho) = \{-\}_-$ if $i \in \bigcup_{l=7}^{11} \gamma_l^{\mathtt{on}}$ or $\bigcup_{l=6}^{10} \gamma_l^{\mathtt{lo}}$ and $\mathtt{vcheck}\left(\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right), pk_{\mathtt{MM}}\right) = \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right)$ and $\mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\rho\right)\right), Z_i^2\rho\right)\right) = Z_1^i\rho$

$vc_i\rho = eaclo(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho) = \{-\}_-$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^3\rho\right)\right) = \perp$

$= eachi(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho) = \{-\}_-$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and $\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(a_i \cdot c_{ind(\rho)}, X_i^1\rho\right)\right), X_i^3\rho\right)\right) = \text{PIN}_i$

$= eacfail(a_i, c_{ind(\rho)}, mk_i, \text{PAN}_i, X_1^i\rho, X_2^i\rho, X_3^i\rho) = \{-\}_-$ if $i \in \alpha_7$ (and, if $\rho = \theta$, $i \in \lambda_j$) and else

$ud_i\rho = \{\langle \mathsf{TX}_i, Z_1^i\rho, \mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_1^i\rho\right)\right), Z_3^i\rho\right), \text{uPIN}\rangle\}_{kbt_i}$ if $i \in \bigcup_{l=9}^{11} \gamma_l^{\mathtt{on}}$

$= \{\langle \mathsf{TX}_i, Z_1^i\rho, \mathtt{p}_1\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_1^i\rho\right)\right), Z_3^i\rho\right)\right), \perp\rangle\}_{kbt_i}$ if $i \in \gamma_{11}^{\mathtt{of}}$

$= \{\langle \mathsf{TX}_i, Z_1^i\rho, \mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_1^i\rho\right)\right), Z_3^i\rho\right), \perp\rangle\}_{kbt_i}$ if $i \in \bigcup_{l=8}^{10} \gamma_l^{\mathtt{lo}}$

$wa_i\rho = \{\langle \mathtt{p}_1(dy_i), \mathtt{accept}\rangle\}_{kbt_i}$ let $dy_i = \mathtt{dec}\left(kbt_i, Y_i^1\rho\right)$ and $\langle \text{PIN}_j, mk_j, \phi(c_j, \mathfrak{g})\rangle = \text{DB}$ if $i \in \beta_4$ and $\exists j$, s.t. $j \in \alpha_7$ and $\mathtt{h}\left(\langle \mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(b_t, \mathtt{p}_2(dy_i))), \mathtt{p}_3(dy_i))), mk_j\rangle\right) = \mathtt{p}_2(\mathtt{dec}(\mathtt{h}(\phi(b_t, \mathtt{p}_2(dy_i))), \mathtt{p}_3(dy_i)))$ and $\mathtt{p}_3(\mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(b_t, \mathtt{p}_2(dy_i))), \mathtt{p}_3(dy_i)))) = \mathtt{p}_1(dy_i)$ and $\phi\left(\mathtt{p}_1(\mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(b_t, \mathtt{p}_2(dy_i))), \mathtt{p}_3(dy_i)))), \phi\left(c_j, \mathfrak{g}\right)\right) = \mathtt{p}_2(dy_i)$ and $\left(\mathtt{p}_2(\mathtt{p}_1(dy_i)) = \mathtt{lo}\right.$ or $\mathtt{p}_2(\mathtt{p}_1(dy_i)) = \mathtt{hi}$ and $\left.\mathtt{p}_4(\mathtt{p}_1(\mathtt{dec}(\mathtt{h}(\phi(b_t, \mathtt{p}_2(dy_i))), \mathtt{p}_3(dy_i)))) = \mathtt{ok}\right.$ or else if $\mathtt{p}_4(dy_i) = \text{PIN}_j)$

$ue_i\rho = \mathtt{auth}$ if $i \in \gamma_9^{\mathtt{of}} \cup \gamma_{11}^{\mathtt{of}}$ and $\mathtt{p}_2\left(\mathtt{dec}\left(\mathtt{h}\left(\phi\left(t_i, Z_i^1\right)\right), Z_i^3\right)\right) = \mathtt{ok}$ or if $i \in \gamma_{11}^{\mathtt{on}}$ or $i \in \gamma_{10}^{\mathtt{lo}}$ and $\mathtt{p}_1\left(\mathtt{dec}\left(kbt_i, Z_4^i\right)\right) = \mathsf{TX}_i$ and $\mathtt{p}_2\left(\mathtt{dec}\left(kbt_i, Z_4^i\right)\right) = \mathtt{accept}$

Notice that $\vec{y} \supset \vec{x}$, hence it is enough to prove that for all messages M and N, s.t. $\vec{x} \# M, N$, we have $M\sigma =_E N\sigma$ iff $M\theta =_E N\theta$. As promised, we conduct the proof of static equivalence by induction on the structure of the weak normal form of $N\sigma$. Below we always start from the equation in the frame $\nu\vec{x}.(\sigma \mid \_)$, since the converse case is similar. In what follows M_i, N_i are recipes, i.e. are fresh for $\vec{x}$.

Case 1. $N\sigma =_E \mathfrak{g}$.

Case 1.1. $N = \mathfrak{g}$. If M is a recipe for $\mathfrak{g}$, then $M = \mathfrak{g}$, since there is no non-trivial recipe for $\mathfrak{g}$ under $\sigma\downarrow_E^{\vec{x}}$ and we have $\mathfrak{g}\sigma =_E \mathfrak{g}\sigma$ iff $\mathfrak{g}\theta =_E \mathfrak{g}\theta$ as required.

Case 1.2. $N \neq \mathfrak{g}$. There is nothing to prove, since there is no non-trivial recipe for $\mathfrak{g}$ under $\sigma\downarrow_E^{\vec{x}}$.

Case 2. $N\sigma =_E z$, z is a variable.

Case 2.1. $N = z$. If M is a recipe for z, then $M = z$, since there is no non-trivial recipe for z under $\sigma\downarrow_E^{\vec{x}}$ and we have $z\sigma =_E z\sigma$ if and only if $z\theta =_E z\theta$ as required.

Case 2.2. $N\sigma =_E ch_i$, where $ch_i \in \{\dot{ch}_i, \ddot{ch}_i, \dddot{ch}_i, \}$. Since N is fresh for $\vec{x}$, $N \in \{chc_i, cht_i, chb_i\}$, and in either case there is a unique recipe $M \in \{chc_i, cht_i, chb_i\}$ for ch_i under $\sigma\downarrow_E^{\vec{x}}$, and we have $M\sigma =_E N\sigma$ iff $M\theta =_E N\theta$ as required.

Case 3. $N\sigma =_E K_1 \cdot K_2$.

Notice that all message terms in the range of $\sigma\downarrow_E^{\vec{x}}$ are m-atomic, hence no message is an immediate m-factor of another message. Therefore $N\sigma$ is generated by m-factors which have a recipe under $\sigma\downarrow_E^{\vec{x}}$.

Case 3.1. $N = N_1^{\epsilon_1} \cdot \ldots \cdot N_k^{\epsilon_k}$, and we have $N\sigma = N_1^{\epsilon_1}\sigma \cdot \ldots \cdot N_k^{\epsilon_k}\sigma$. By the induction hypothesis suppose that for all recipes M_i for an m-factor $N_i\sigma$ of $N\sigma$, we have $M_i\sigma =_E N_i\sigma$ iff $M_i\theta =_E N_i\theta$ for $i \in \{1, \ldots, k\}$. By applying multiplication, we have $M_1^{\epsilon_1}\theta \cdot \ldots \cdot M_k^{\epsilon_k}\theta = (M_1^{\epsilon_1} \cdot \ldots \cdot M_k^{\epsilon_k})\theta =_E (N_1^{\epsilon_1} \cdot \ldots \cdot N_k^{\epsilon_k})\theta = N_1^{\epsilon_1}\theta \cdot \ldots \cdot N_k^{\epsilon_k}\theta$ as required, and $N_i\theta$ is an m-factor of $N\theta$.

Case 4. $N\sigma =_E \phi(K_1, K_2)$.

We have several recipes of the form $\phi(\cdot,\cdot)$ in the domain of $\sigma\downarrow_E^{\vec{x}}$.

$$
\begin{aligned}
&V_1 := ua_i, V_2 := va_i\\
&V_3 := \mathtt{p}_2\big(\mathtt{p}_1\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(B_i^1, ua_i\big)\big), ub_i\big)\big)\big)\\
&V_4 := \mathtt{p}_2\big(\mathtt{check}\big(\mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(B_i^1, ua_i\big)\big), ub_i\big)\big), pk_s\big)\big)\\
&V_5 := \mathtt{p}_1\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(A_1^i, va_i\big)\big), vb_i\big)\big)\\
&V_6 := \mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(A_1^i, va_i\big)\big), vb_i\big)\big)\\
&V_7 := \mathtt{vcheck}\big(\mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi\big(A_1^i, va_i\big)\big), vb_i\big)\big), pk_{\mathtt{MM}}\big)
\end{aligned}
$$

Case 4.1. $N\sigma =_E \phi(t_i, \mathfrak{g})$. Since N is fresh for $\vec{x}$, $N = V_1$. Let M be a recipe for $\phi(t_i, \mathfrak{g})$, then $M = V_1$ and we have $V_1\sigma =_E V_1\sigma$ iff $V_1\theta =_E V_1\theta$ as required.

Case 4.2. $N\sigma =_E \phi(a_i \cdot c_i, \mathfrak{g})$ and $X_i^1 = \phi\big(A_i^1, \mathfrak{g}\big)$. Since N is fresh for $\vec{x}$, $N \in \{V_2, V_5, V_7\}$. Let M be a recipe for $\phi(a_i \cdot c_i, \mathfrak{g})$, then $M \in \{V_2, V_5, V_7\}$ and we have $M\sigma =_E N\sigma$ iff $M\theta =_E N\theta$ for any N and M as required. If $X_i^1 \neq \phi\big(A_i^1, \mathfrak{g}\big)$, $N = V_2$, there is only one recipe $M_1 = V_1$ and the argument is the same.

Case 4.3. $N\sigma =_E \phi(b_t, \mathfrak{g})$ and $Z_i^1 = \phi\big(B_i^1, \mathfrak{g}\big)$. Since N is fresh for $\vec{x}$, $N \in \{V_3, V_4\}$. Let M be a recipe for $\phi(b_t, \mathfrak{g})$, then $M \in \{V_3, V_4\}$ and the argument is identical to Case 4.1. If $Z_i^1 \neq \phi\big(B_i^1, \mathfrak{g}\big)$ there is no recipe for $\phi(b_t, \mathfrak{g})$ and there is nothing to prove.

Case 4.4. $N\sigma =_E \phi(a_i \cdot c_i, \mathtt{vsig}(\mathfrak{g}, \chi_{\mathtt{MM}}))$ when $X_i^1 = \phi\big(A_i^1, \mathfrak{g}\big)$. Identical to Case 4.1, where $N = M = V_6$, and there is nothing to prove if $X_i^1 \neq \phi\big(A_i^1, \mathfrak{g}\big)$.

Case 4.5. $N = \phi(N_1, N_2)$, $N_2 \in \{V_1, \ldots V_7\}$ for $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$. By the induction hypothesis suppose that for all recipes M_1 for $N_1\sigma$, we have $M_1\sigma =_E N_1\sigma$ iff $M_1\theta =_E N_1\theta$, then by multiplying N_2 by M_1 we get $\phi(M_1\theta, N_2\theta) = \phi(M_1, N_2)\,\theta =_E \phi(N_1, N_2)\,\theta = \phi(N_1\theta, N_2\theta)$ for any N_2 as required. In case $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$ we have $N_2 \in \{\ldots \widehat{V}_3, \widehat{V}_4 \ldots\}$; in case $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$ we have $N_2 \in \{\ldots \widehat{V}_5, \widehat{V}_6, \widehat{V}_7\}$; and in case $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$ we have $N_2 \in \{V_1, V_2\}$, and the argument is the same.

Case 4.6. $N = \mathtt{sig}(\ldots \mathtt{sig}(N_1, N_2) \ldots, N_k)$, $N_1 \in \{V_1, \ldots, V_7\}$ for $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$. By the induction hypothesis suppose that for all recipes M_i for $N_i\sigma$ we have $M_i\sigma =_E N_i\sigma$ iff $M_i\theta =_E N_i\theta$ for any $i \in \{2, \ldots, k\}$. By applying the $\mathtt{vsig}(\cdot,\cdot)$ function to N_1, we have

$$\begin{aligned}
&\mathtt{vsig}(\ldots \mathtt{vsig}(N_1, M_2) \ldots, M_k)\theta = \\
&\mathtt{vsig}(\ldots \mathtt{vsig}(N_1\theta, M_2\theta) \ldots, M_k\theta) =_E \\
&\mathtt{vsig}(\ldots \mathtt{vsig}(N_1\theta, N_2\theta) \ldots, N_k\theta) = \\
&\mathtt{vsig}(\ldots \mathtt{vsig}(N_1, N_2) \ldots, N_k)\theta
\end{aligned}$$

as required. In case $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$ we have $N_1 \in \{\ldots \widehat{V}_3, \widehat{V}_4 \ldots\}$; in case $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$ we have $N_1 \in \{\ldots \widehat{V}_5, \widehat{V}_6, \widehat{V}_7\}$; and in case $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$ we have $N_1 \in \{V_1, V_2\}$, and the argument is the same.

Case 4.7. $N = \phi(N_1, N_2)$. Identical to Case 3.1, where $\epsilon_1 = \epsilon_2 = 1$, $k = 2$.

Case 5. $N\sigma =_E \langle K_1, K_2\rangle$.

The range of $\sigma|_E^{\vec{x}}$ contains no pair, hence the only option is $N = \langle N_1, N_2\rangle$, which is identical to Case 4.7.

Case 6. $N\sigma =_E \mathtt{h}(K_1)$. Identical to Case 3.1, where $\epsilon_1 = 1$, $k = 1$.

Case 7. $N\sigma =_E \mathtt{pk}(K_1)$.

Case 7.1. $N\sigma =_E \mathtt{pk}(s)$. Then $N = pk_s$, since N is fresh for $\vec{x}$. There is a unique recipe $M = pk_s$ for $\mathtt{pk}(s)$ and we have $pk_s\sigma =_E pk_s\sigma$ if and only if $pk_s\theta =_E pk_s\theta$ as required.

Case 7.2. $N = \mathtt{pk}(N_1)$. Identical to Case 6.

Case 8. $N\sigma =_E \mathtt{vpk}(K_1)$. Identical to Case 7, since there is a unique recipe $pk_{\mathtt{MM}}$ in the range of $\sigma|_E^{\vec{x}}$.

Case 9. $N\sigma =_E \mathtt{vsig}(K_1, K_2)$. Identical to Case 5.

Case 10. $N\sigma =_E \mathtt{sig}(K_1, K_2)$. Identical to Case 7, since there is a unique recipe $\mathtt{p}_2\big(\mathtt{dec}\big(\mathtt{h}\big(\phi(B_i^1, ua_i)\big), ub_i\big)\big)$ in the range of $\sigma|_E^{\vec{x}}$ if $Z_i^1 = \phi(B_i^1, \mathfrak{g})$, and there is nothing to prove if $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$.

Case 11. $N\sigma =_E \{K_1\}_{K_2}$.

Cases 11.1-11.4 are identical to Case 2.2, however we list all possibilities for the sake of completeness. Let $ENK = \{uc_i\sigma, vc_i\sigma, ud_i\sigma, wa_i\sigma\}$.

Case 11.1. $N\sigma =_E enk$, where $enk \in ENK$, and $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$.

Case 11.2. $N\sigma =_E enk$, where $enk \in ENK \cup \{ub_i\sigma\}$, and $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 = \phi(A_i^1, \mathfrak{g})$.

Case 11.2. $N\sigma =_E enk$, where $enk \in ENK \cup \{vb_i\sigma\}$, and $Z_i^1 = \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$.

Case 11.4. $N\sigma =_E enk$, where $enk \in ENK \cup \{ub_i\sigma, vb_i\sigma\}$, and $Z_i^1 \neq \phi(B_i^1, \mathfrak{g})$ and $X_i^1 \neq \phi(A_i^1, \mathfrak{g})$.

Case 11.5. $N = \{N_1\}_{N_2}$. identical to Case 5.
Case 12. $N\sigma =_E \mathtt{MM}$.
Case 12.1. $N = \mathtt{MM}$. Identical to Case 1.1.
Case 12.2. $N \neq \mathtt{MM}$, and $N\sigma =_E \mathtt{MM}$. Identical to Case 2.2, when $Z_i^1 = \phi(B_i^1, \mathfrak{g})$.
Case 13. $N\sigma \in \{\bot, \mathtt{ok}, \mathtt{accept}, \mathtt{auth}, \mathtt{lo}, \mathtt{hi}\}$. Identical to Case 1. □

Concluding remarks on the proof method. To prove that the UTX protocol is unlinkable, we have followed the steps identified in Sec. 3.5.2 after the proof of Lemma 1 with two major differences. Firstly, we could not withdraw any party from the protocol and apply the compositionality right in the definition since parties now have shared secrets. This has led us to a significant increase in the work because we have had to consider all possible moves the process specifying the terminals and the banks could do. Secondly, the change in message theory has forced us to update the definitions for E-weak normal form and the respective frame normalisation. Beyond this, yet again, we identify constructing a quasi-open bisimulation serving as a proof certificate as the most creative step in the proof. Below, however, we demonstrate how compositionality can still play its role in the context of UTX.

4.3.4 Further results obtained by compositionality

In this subsection we demonstrate how the Theorem 3, stating that quasi-open bisimilarity is a congruence relation, could be employed to show that the UTX protocol is unlinkable in the presence of multiple signing authorities. We also hypothesise that the verification effort can be significantly reduced by dropping terminals entirely in case all transactions are low-value, i.e. not requiring the PIN.

Unlinkability in the face of coarse identities. To permit the situation where multiple signing authorities exist at the same time it is enough to put the replication in front of UTX_{spec} and UTX_{impl} from Fig. 4.7. The following corollary, then, is straightforward.

Corollary 1. $!UTX_{impl} \sim !UTX_{spec}$.

Proof. By Theorem 8 we have $UTX_{\text{impl}} \sim UTX_{\text{spec}}$. By Theorem 3 quasi-open bisimilarity is a congruence relation, i.e. it holds in any context. Consider the context $\mathcal{O}(\cdot) := !(\cdot)$ which, after either UTX_{impl} or UTX_{spec} being put in $\mathcal{O}$ gives a system with multiple signing authorities, i.e. we obtain $!UTX_{\text{impl}} \sim !UTX_{\text{spec}}$. □

Thereby we permit a coarse identity, a signing authority, to exist in the system, as represented by building multiple authorities into the specification $!UTX_{\text{spec}}$, without compromising unlinkability. This justifies the observation in Sec. 4.1.3 where we have pointed out that unlinkability can only be achieved up to the fingerprint comprising the coarse identities of the card being revealed. In particular, Corollary 1 concerns the case where multiple payment systems might not agree to provide a common application for unlinkable payments as discussed in Sec. 4.2.1, and therefore these different payment systems form a coarse identity of the card. Corollary 1 ensures that even in this scenario UTX remains unlinkable.

Unlinkability in case all payments are low-value. Here we expand on the point made at the end of Sec. 4.3.1 that for low-value contactless payments, unlinkability is preserved even if the PIN, the card's strong identity, is compromised. The key observation is that low-value terminals do not require the PIN, i.e. no input on the private channel *user* is expected.

To model the situation when all payments are low-value, we drop high-value terminals $T_{\texttt{onhi}}$ and $T_{\texttt{offhi}}$ from the picture and simplify how banks communicate with honest terminals, i.e. Instead of session-specific bank-terminal symmetric key *kbt*, as in Fig. 4.7, we use one global shared symmetric key. Fig. 4.11 contains the real-world and the idealised-unlinkable versions of system with low-value payments only. Let us call such reduced version of the protocol UTXL (UTX Low).

(a) The real protocol specification $UTXL_{\text{impl}}$.

$$
\begin{aligned}
&\nu\, s, si, \chi_{\texttt{MM}}, b_t, kbt.\overline{out_s}\langle \texttt{pk}(s)\rangle.\Big(\\
&\quad !\nu \text{PIN}, mk, c, \text{PAN}.\big(\\
&\qquad \overline{opin}\langle \text{PIN}\rangle.\\
&\qquad \texttt{let crtC} := \texttt{vsig}(\phi(c,\mathfrak{g})\,, \chi_{\texttt{MM}})\,\texttt{in}\\
&\qquad !\nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s,\mathfrak{g})\,, \texttt{crtC}, \text{PAN}, mk, \text{PIN}) \mid\\
&\qquad !\overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big) \mid\\
&\qquad !\nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t) \mid\\
&\qquad \texttt{let crt} := \langle\langle \texttt{MM}, \phi(b_t,\mathfrak{g})\rangle, \texttt{sig}(\langle \texttt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle\texttt{in}\\
&\qquad \overline{out_v}\langle \texttt{vpk}(\chi_{\texttt{MM}})\rangle.\\
&\qquad \overline{out_c}\langle \texttt{crt}\rangle.\\
&\qquad !\nu ch.\overline{term}\langle ch\rangle.T_{\texttt{lo}}(ch, \texttt{vpk}(\chi_{\texttt{MM}})\,, \texttt{crt}, kbt)\ \Big)
\end{aligned}
$$

(b) The ideal unlinkable protocol specification $UTXL_{\text{spec}}$.

$$
\begin{aligned}
&\nu\, s, si, \chi_{\texttt{MM}}, b_t, kbt.\overline{out_s}\langle \texttt{pk}(s)\rangle.\Big(\\
&\quad !\nu \text{PIN}, mk, c, \text{PAN}.\big(\\
&\qquad \overline{opin}\langle \text{PIN}\rangle.\\
&\qquad \texttt{let crtC} := \texttt{vsig}(\phi(c,\mathfrak{g})\,, \chi_{\texttt{MM}})\,\texttt{in}\\
&\qquad \nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s,\mathfrak{g})\,, \texttt{crtC}, \text{PAN}, mk, \text{PIN}) \mid\\
&\qquad !\overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big) \mid\\
&\qquad !\nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t) \mid\\
&\qquad \texttt{let crt} := \langle\langle \texttt{MM}, \phi(b_t,\mathfrak{g})\rangle, \texttt{sig}(\langle \texttt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle\texttt{in}\\
&\qquad \overline{out_v}\langle \texttt{vpk}(\chi_{\texttt{MM}})\rangle.\\
&\qquad \overline{out_c}\langle \texttt{crt}\rangle.\\
&\qquad !\nu ch.\overline{term}\langle ch\rangle.T_{\texttt{lo}}(ch, \texttt{vpk}(\chi_{\texttt{MM}})\,, \texttt{crt}, kbt)\ \Big)
\end{aligned}
$$

Figure 4.11: Specifications for the real UTXL protocol and its ideal unlinkable version.

Besides the differences with the full system in Fig. 4.7 emphasised above, in the system with low-value transactions only we explicitly output PINs on the public channel *opin*, and the public information that an attacker may use to construct a low-value accepting terminal – the public key to verify the card, and the bank's certificate – in the channels out_v, out_c next to the terminal's process ($(\overline{out_v}\langle \mathtt{vpk}(\chi_{\mathtt{MM}})\rangle, \overline{out_c}\langle \mathsf{crt}\rangle)$). To verify the unlinkability of UTXL, we should show that $UTXL_{\text{impl}} \sim UTXL_{\text{spec}}$, however in that case we could reduce the amount of work needed for verification using compositionality and verify a strictly smaller system.

Let us consider in Fig. 4.12 the respective real-world and idealised subsystems of UTXL called SUTXL (Small UTXL) comprising only cards and banks.

(a) The real protocol specification $SUTXL_{\text{impl}}$.

$$
\begin{aligned}
&\nu\, s, si, \chi_{\mathtt{MM}}, b_t.\\
&\mathtt{let}\ \mathsf{crt} := \langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\overline{out_v}\langle \mathtt{vpk}(\chi_{\mathtt{MM}})\rangle.\\
&\overline{out_c}\langle \mathsf{crt}\rangle.\\
&\overline{out_s}\langle \mathtt{pk}(s)\rangle.\Big(\\
&\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
&\quad \overline{opin}\langle \mathrm{PIN}\rangle.\\
&\quad \mathtt{let}\ \mathsf{crtC} := \mathtt{vsig}(\phi(c, \mathfrak{g})\,, \chi_{\mathtt{MM}})\ \mathtt{in}\\
&\quad !\nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s, \mathfrak{g})\,, \mathsf{crtC}, \mathrm{PAN}, mk, \mathrm{PIN}) \mid\\
&\quad !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle\ \big) \mid\\
&\quad !\nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \Big)
\end{aligned}
$$

(b) The ideal unlinkable protocol specification $SUTXL_{\text{spec}}$.

$$
\begin{aligned}
&\nu\, s, si, \chi_{\mathtt{MM}}, b_t.\\
&\mathtt{let}\ \mathsf{crt} := \langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\overline{out_v}\langle \mathtt{vpk}(\chi_{\mathtt{MM}})\rangle.\\
&\overline{out_c}\langle \mathsf{crt}\rangle.\\
&\overline{out_s}\langle \mathtt{pk}(s)\rangle.\Big(\\
&\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
&\quad \overline{opin}\langle \mathrm{PIN}\rangle.\\
&\quad \mathtt{let}\ \mathsf{crtC} := \mathtt{vsig}(\phi(c, \mathfrak{g})\,, \chi_{\mathtt{MM}})\ \mathtt{in}\\
&\quad \nu ch.\overline{card}\langle ch\rangle.C(ch, c, \phi(s, \mathfrak{g})\,, \mathsf{crtC}, \mathrm{PAN}, mk, \mathrm{PIN}) \mid\\
&\quad !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle\ \big) \mid\\
&\quad !\nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \Big)
\end{aligned}
$$

Figure 4.12: Subsystem specifications for SUTXL.

In SUTXL we not only assume that all transactions are low-value and executed

with any unauthorised device constructed using public information, but we also allow the bank to process any transactions received, as the variable *kbt* in SUTXL specification is not bound (in contrast to UTXL). To justify that it is enough to verify that $SUTXL_{\text{impl}} \sim SUTXL_{\text{spec}}$, we provide the following lemma.

Lemma 3. *If* $SUTXL_{impl} \sim SUTXL_{spec}$, *then* $UTXL_{impl} \sim UTXL_{spec}$.

Proof. Consider the following context.

$$\begin{aligned}\mathcal{L}\{\cdot\} \triangleq\ & \nu\, out_v, out_c, kbt.\, (\{\cdot\} \mid \\ & out_v(pk_{\texttt{MM}}).out_c(\texttt{crt}). \\ & \overline{out'_v}\langle pk_{\texttt{MM}}\rangle.\overline{out'_c}\langle\texttt{crt}\rangle. \\ & !vch.\overline{term}\langle ch\rangle.T_{\texttt{lo}}(ch, pk_{\texttt{MM}}, \texttt{crt}, kbt))\end{aligned}$$

Similarly to the proof of Theorem 4, after we plug in the context $\mathcal{L}\{\cdot\}$ either $SUTXL_{\text{impl}}$ or $SUTXL_{\text{spec}}$, it will take two τ transitions and the application of the substitution $\left\{{}^{out_v,out_c}/_{out'_v,out'_c}\right\}$ (since quasi-open bisimilarity is closed under substitutions) to obtain the initial bigger system ($UTXL_{\text{impl}}$ and $UTXL_{\text{spec}}$ respectively). Therefore the context above leads to the correct representation of the full system, and we can verify against the reduced definition, dropping low-value terminals completely. □

To conclude that the system where all payments are low-value, it is still left to verify that the subsystem represented by the SUTXL protocol in Fig. 4.12 is unlinkable. We formulate this claim separately as a hypothesis since we are leaving the proof for future work, however we expect such proof to be quite close to the proof of the Theorem 8 since it considers a more general case.

Hypothesis 1. SUTXL(UTXL) *is unlinkable, i.e.* $SUTXL_{impl} \sim SUTXL_{spec}$.

4.3.5 On future unlinkability proofs

As mentioned at the beginning of this section, we restrict our analysis to the case where all cards are synchronised to execute within the same month `MM`. However, we would like to expand on the model that admits a transition from one month to the next. In this small section, we call such an enhanced model UTXMM.

To reflect such behaviour of cards, we require each card to respond to two months at any time and, whenever the new month is asked, to invalidate the oldest of two months. Notice that this requires a card to carry the state, i.e. to "remember" that it should respond only to the recent month and never return responding to the older months if asked. Below we show how we can employ recursion to model such behaviour.

Without loss of generality, we restrict the model to three months `M1`, `M2`, `M3`, and populate the world with two types of cards – responding to `M1`, `M2` or to `M2`, `M3`.

Notice that a card can advance its pointer to the next month only in the real-world system, where it can participate in multiple transactions. In contrast, in the idealised scenario, where cards are disposable, no change in the state of a given card is required. This requires us to have two different role specifications for cards.

In Fig. 4.13, we give the specification for the card's role in the real-world system.

$$
\begin{aligned}
&Crw^{rec}(c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad \nu ch.\overline{card}\langle ch\rangle. \\
&\quad ch(z_1). \\
&\quad \nu a.\ \mathtt{let} z_2 := \phi(a, \phi(c, \mathfrak{g}))\ \mathtt{in} \\
&\quad \overline{ch}\langle z_2\rangle. \\
&\quad \mathtt{let}\, k_c := \mathtt{h}(\phi(a \cdot c, z_1))\, \mathtt{in} \\
&\quad ch(m). \\
&\quad \mathtt{let}\, \langle\langle \mathsf{MM}, y_B\rangle, \mathsf{MC}_s\rangle := \mathtt{dec}(k_c, m)\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{check}(\mathsf{MC}_s, pk_s) = \langle \mathsf{MM}, y_B\rangle \\
&\quad \mathtt{if}\ \mathsf{MM} = \mathtt{M1}\ \mathtt{then} \\
&\qquad Cont^{rec}(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else\ if}\ \mathsf{MM} = \mathtt{M2}\ \mathtt{then} \\
&\qquad Cont^{rec}(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else\ if}\ \mathsf{MM} = \mathtt{M3}\ \mathtt{then} \\
&\qquad \nu \chi_{\mathtt{M4}}.\mathtt{let}\ \mathsf{crtC4} := \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{M4}})\ \mathtt{in} \\
&\qquad Cont^{rec}(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathsf{crtC4}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else} \\
&\qquad Crw^{rec}(c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN})
\end{aligned}
$$

$$
\begin{aligned}
&Cont^{rec}(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad \overline{ch}\langle\{\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}_{\mathtt{MM}})\rangle\}_{k_c}\rangle. \\
&\quad ch(x). \\
&\quad \mathtt{let}\, \langle \mathsf{TX}, \mathtt{uPin}\rangle := \mathtt{dec}(k_c, x)\, \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX}\rangle\, \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC}^{\mathtt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \mathtt{ok}\rangle\, \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp\rangle\, \mathtt{in} \\
&\quad \mathtt{let}\, k_{cb} := \mathtt{h}(\phi(a \cdot c, y_B))\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{uPin} = \perp\ \mathtt{then} \\
&\qquad \overline{ch}\langle\{\{\langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk\rangle)\rangle\}_{k_{cb}}\}_{k_c}\rangle. \\
&\qquad Crw^{rec}(c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else\,if}\ \mathtt{uPin} = \mathrm{PIN}\ \mathtt{then} \\
&\qquad \overline{ch}\langle\{\{\mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\mathsf{AC}^{\mathtt{ok}}, mk)\}_{k_{cb}}, \mathtt{ok}\}_{k_c}\rangle. \\
&\qquad Crw^{rec}(c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else} \\
&\qquad \overline{ch}\left\langle\{\{\mathsf{AC}^{\perp}, \mathtt{h}(\mathsf{AC}^{\perp}, mk)\}_{k_{cb}}, \perp\}_{k_c}\right\rangle. \\
&\qquad Crw^{rec}(c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN})
\end{aligned}
$$

Figure 4.13: The real-world specification of the card's role in UTXMM.

The specification of the card's behaviour in the real-world scenario is split into two parts. In the initial part, represented by the process Crw^{rec}, the card decides if it needs to advance the pointer to the next month, and the rest, represented by $Cont^{rec}$. The card is initially set up to respond for months `M1`, `M2`. Whenever one of the two is asked, the process $Cont^{rec}$ at the end of the transaction refers to Crw^{rec} with the same parameters, but if the month asked is `M3`, the process $Cont^{rec}$ at the end of the run calls the process Crw^{rec} with a "shifted" list of month signatures: $\texttt{vsig}_{\texttt{M2}}, \texttt{vsig}_{\texttt{M3}}, \texttt{crtC4}$. The possibility to complete the session responding to the month `M1` is now lost for the card c – it simply aborts the protocol by restarting the session (the else branch in the last line of Crw^{rec}).

The card's role in the idealised world is specified in Fig. 4.14. There is no recursion in the card's role in contrast to the real-world spec. The card simply continues the run replying to any month asked, and then is getting disposed of.

$$
\begin{aligned}
&Cid(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad ch(z_1). \\
&\quad \nu a.\ \mathtt{let}\, z_2 := \phi(a, \phi(c, \mathfrak{g}))\ \mathtt{in} \\
&\quad \overline{ch}\langle z_2 \rangle. \\
&\quad \mathtt{let}\, k_c := \mathtt{h}(\phi(a \cdot c, z_1))\ \mathtt{in} \\
&\quad ch(m). \\
&\quad \mathtt{let}\, \langle \langle \mathsf{MM}, y_B \rangle, \mathsf{MC}_s \rangle := \mathtt{dec}(k_c, m)\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{check}(\mathsf{MC}_s, pk_s) = \langle \mathsf{MM}, y_B \rangle \\
&\quad \mathtt{if}\ \mathsf{MM} = \mathtt{M1}\ \mathtt{then} \\
&\qquad Cont(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else\ if}\ \mathsf{MM} = \mathtt{M2}\ \mathtt{then} \\
&\qquad Cont(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
&\quad \mathtt{else\ if}\ \mathsf{MM} = \mathtt{M3}\ \mathtt{then} \\
&\qquad \nu \chi_{\mathtt{M4}}.\mathtt{let}\, \mathsf{crtC4} := \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{M4}})\ \mathtt{in} \\
&\qquad Cont(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathsf{crtC4}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
\\
&Cont(ch, c, pk_s, \mathtt{vsig}_{\mathtt{M1}}, \mathtt{vsig}_{\mathtt{M2}}, \mathtt{vsig}_{\mathtt{M3}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad \overline{ch}\langle \{ \langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}_{\mathtt{MM}}) \rangle \}_{k_c} \rangle. \\
&\quad ch(x). \\
&\quad \mathtt{let}\, \langle \mathsf{TX}, \mathtt{uPin} \rangle := \mathtt{dec}(k_c, x)\ \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX} \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC}^{\mathtt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \mathtt{ok} \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\, \mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\, k_{cb} := \mathtt{h}(\phi(a \cdot c, y_B))\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{uPin} = \perp\ \mathtt{then} \\
&\qquad \overline{ch}\langle \{ \{ \langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk \rangle) \rangle \}_{k_{cb}} \}_{k_c} \rangle \\
&\quad \mathtt{else\ if}\ \mathtt{uPin} = \mathrm{PIN}\ \mathtt{then} \\
&\qquad \overline{ch}\langle \{ \{ \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\mathsf{AC}^{\mathtt{ok}}, mk) \}_{k_{cb}}, \mathtt{ok} \}_{k_c} \rangle \\
&\quad \mathtt{else} \\
&\qquad \overline{ch}\Big\langle \{ \{ \mathsf{AC}^{\perp}, \mathtt{h}(\mathsf{AC}^{\perp}, mk) \}_{k_{cb}}, \perp \}_{k_c} \Big\rangle
\end{aligned}
$$

Figure 4.14: The ideal-world specification of the card's role in UTXMM.

Finally, we define the spec and imp worlds of UTXMM in Fig. 4.15.

(a) The real protocol specification $UTXMM_{\text{impl}}$.

$$
\begin{aligned}
&\nu\, user, s, si, \chi_{\mathtt{M1}}, \chi_{\mathtt{M2}}, \chi_{\mathtt{M3}}.\overline{out}\langle \mathtt{pk}(s)\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M1}})\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M2}})\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M3}})\rangle.\Big(\\
&\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
&\qquad \mathtt{let}\ \mathsf{crtC1} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M1}})\ \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crtC2} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M2}})\ \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crtC3} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M3}})\ \mathtt{in}\\
&\qquad\quad Crw^{rec}(c, \phi(s,\mathfrak{g}), \mathsf{crtC1}, \mathsf{crtC2}, \mathsf{crtC3}, \mathrm{PAN}, mk, \mathrm{PIN}) +\\
&\qquad\quad \nu\chi_{\mathtt{M4}}.\mathtt{let}\ \mathsf{crtC4} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M4}})\ \mathtt{in}\\
&\qquad\quad Crw^{rec}(c, \phi(s,\mathfrak{g}), \mathsf{crtC2}, \mathsf{crtC3}, \mathsf{crtC4}, \mathrm{PAN}, mk, \mathrm{PIN})\\
&\qquad \mid !\overline{user}\langle \mathrm{PIN}\rangle \mid !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big)\ \mid\\
&\quad \nu b_t.!\nu kbt.\big(\\
&\qquad \nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \mid\\
&\qquad \mathtt{let}\ \mathsf{crt1} := \langle\langle \mathtt{M1}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M1}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crt2} := \langle\langle \mathtt{M2}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M2}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crt3} := \langle\langle \mathtt{M3}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M3}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M1}}), \mathsf{crt1}, kbt) +\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M2}}), \mathsf{crt2}, kbt) +\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M3}}), \mathsf{crt3}, kbt)\big)\ \Big)
\end{aligned}
$$

(b) The ideal unlinkable protocol specification $UTXMM_{\text{spec}}$.

$$
\begin{aligned}
&\nu\, user, s, si, \chi_{\mathtt{M1}}, \chi_{\mathtt{M2}}, \chi_{\mathtt{M3}}.\overline{out}\langle \mathtt{pk}(s)\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M1}})\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M2}})\rangle.\overline{out}\langle \mathtt{vpk}(\chi_{\mathtt{M3}})\rangle.\Big(\\
&\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
&\qquad \mathtt{let}\ \mathsf{crtC1} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M1}})\ \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crtC2} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M2}})\ \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crtC3} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M3}})\ \mathtt{in}\\
&\qquad\quad \nu ch.\overline{card}\langle ch\rangle.Cid(ch, c, \phi(s,\mathfrak{g}), \mathsf{crtC1}, \mathsf{crtC2}, \mathsf{crtC3}, \mathrm{PAN}, mk, \mathrm{PIN}) +\\
&\qquad\quad \nu\chi_{\mathtt{M4}}.\chi\mathtt{let}\ \mathsf{crtC4} := \mathtt{vsig}(\phi(c,\mathfrak{g}), \chi_{\mathtt{M4}})\ \mathtt{in}\\
&\qquad\quad \nu ch.\overline{card}\langle ch\rangle.Cid(ch, c, \phi(s,\mathfrak{g}), \mathsf{crtC2}, \mathsf{crtC3}, \mathsf{crtC4}, \mathrm{PAN}, mk, \mathrm{PIN})\\
&\qquad \mid !\overline{user}\langle \mathrm{PIN}\rangle \mid !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big)\ \mid\\
&\quad \nu b_t.!\nu kbt.\big(\\
&\qquad \nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \mid\\
&\qquad \mathtt{let}\ \mathsf{crt1} := \langle\langle \mathtt{M1}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M1}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crt2} := \langle\langle \mathtt{M2}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M2}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \mathtt{let}\ \mathsf{crt3} := \langle\langle \mathtt{M3}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{M3}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in}\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M1}}), \mathsf{crt1}, kbt) +\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M2}}), \mathsf{crt2}, kbt) +\\
&\qquad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathtt{vpk}(\chi_{\mathtt{M3}}), \mathsf{crt3}, kbt)\big)\ \Big)
\end{aligned}
$$

Figure 4.15: Specifications for the real UTXMM protocol and its ideal unlinkable version.

We would like to highlight two crucial differences with the respective specification of UTX presented previously in Fig. 4.7. Firstly, since we consider two types of cards that respond either to M1, M2 or to M2, M3 we are taking care of populating the system with both types. Right at the start, there are cards with the pointer already advanced, represented by the second branch in the choice in the card's part of the specification, i.e. $\nu\chi_{\mathtt{M4}}.\mathtt{let}\,\mathtt{crtC4} := \ldots$. Secondly, the replication in the $UTXMM_{\text{impl}}$ is now implicit since the process Crw^{rec} is recursive.

We expect an enhanced model of the UTX protocol described in this section to be also unlinkable. However, the full proof would require at least the same amount of work as the proof of Theorem 8 plus some additional complications of handling the recursion since the evidence that UTXMM is unlinkable is left as future work, we formulate the following.

Hypothesis 2. UTXLMM *is unlinkable, i.e.* $UTXMM_{impl} \sim UTXMM_{spec}$.

4.3.6 Authentication and secrecy in UTX

Our security definition relies on correspondence assertions between events of the ProVerif tool [B+01, Bla09] and the authentication property of injective agreement, which we have already explained and used in Sec. 3.5.3.

Recall that injective agreement [Low97] between parties X and Y ensures the following two conditions are satisfied.

- *agreement* When, e.g., X thinks it has authenticated Y, then Y executed the protocol exchanging the same messages as X.
- *injectivity* Each run of X corresponds to a unique run of Y.

To verify the security requirements identified in Sec. 4.1.2 we establish that the UTX protocol satisfies the injective correspondences between events listed in Fig. 4.16, 4.17. The events are parametrised by the messages the card, the terminal and the bank exchange, i.e., z_1, z_2 stand for the ephemeral terminal's key and the blinded card's public key, *ecert*, *emcert*, *etx*, *eac* represent the messages the card exchanges with the terminal and *req*, *resp* the message the terminal exchanges with the bank. In Fig. 4.18, 4.19, 4.18 we present the exact locations where we place these events in each of the role specifications.

The terminal agrees with the card

$$\mathtt{TCommitWithC}(z_1, z_2, ecert, emcert, etx, eac) \Rightarrow \mathtt{CRunning}(z_1, z_2, ecert, emcert, etx, eac)$$

The terminal agrees wih the bank and the card

$$\mathtt{TCommitWithBC}(req, resp, z_1, z_2, ecert, emcert, etx, eac) \Rightarrow (\mathtt{BRunningWithT}(req, resp) \wedge \mathtt{CRunning}(z_1, z_2, ecert, emcert, etx, eac))$$

The bank agrees with the terminal and the card

$$\mathtt{BCommitWithTC}(req) \Rightarrow (\mathtt{TRunningWithBC}(req, z_1, z_2, ecert, emcert, etx, eac) \wedge \mathtt{CRunning}(z_1, z_2, ecert, emcert, etx, eac))$$

Figure 4.16: Correspondence assertions for injective agreement in UTX.

The first assertion in Fig. 4.16 is straightforward – whenever the terminal thinks it has executed the session with the card, they have exchanged the same messages, thereby agreeing on crucial data such as the derived keys, transaction details, the cryptogram, etc. The second assertion ensures that when the terminal has completed a session after receiving the response from the bank, the bank and the card have the same information as the terminal has. Similarly, the third assertion verifies that when the bank received the message from the terminal, then the terminal and the card completed the session in which they agreed on all messages they exchanged.

Furthermore we check that the additional functional properties presented in Fig. 4.17 are satisfied in UTX.

Transaction security
$\texttt{TAccept}(kbt, \mathsf{TX}) \Rightarrow \neg\texttt{BReject}(kbt, \mathsf{TX})$

Bank agrees with the card on the encrypted cryptogram
$\texttt{BCommitWithC}(\mathsf{EAC}) \Rightarrow \texttt{CRunningWithB}(\mathsf{EAC})$

Figure 4.17: Additional functional properties of UTX.

The natural property of *transaction security* ensures that if a terminal accepts a transaction after a payment session with the card (and therefore, the merchant delivered the goods), then it should be the case that the issuing bank always accepts that transaction (and therefore the merchant gets effectively paid). In addition, we check that the bank and the card agree on a single piece of data the card intends to send to the bank, the encrypted cryptogram.

Our ProVerif specification of the UTX protocol slightly differs from the one presented in Fig. 4.7a. in two aspects and is presented below. Firstly, we drop session channels and use common open public channels *card*, *bank* and *term* in role specifications. Secondly, we allow all three types of terminals to run in parallel since the non-deterministic choice $+$ operator is not supported in ProVerif.

$$
\begin{aligned}
&\nu\, user, s, si, \chi_{\texttt{MM}}.\overline{out}\langle \texttt{pk}(s)\rangle.\overline{out}\langle \texttt{vpk}(\chi_{\texttt{MM}})\rangle.\Big(\\
&\quad !\nu \text{PIN}, mk, c, \text{PAN}.\big(\\
&\qquad \texttt{let}\ \mathsf{crtC} := \texttt{vsig}(\phi(c, \mathfrak{g})\,, \chi_{\texttt{MM}})\ \texttt{in}\\
&\qquad\quad !C(card, c, \phi(s, \mathfrak{g})\,, \mathsf{crtC}, \text{PAN}, mk, \text{PIN})\\
&\qquad \mid !\overline{user}\langle \text{PIN}\rangle \mid !\overline{\langle si, \text{PAN}\rangle}\langle\langle \text{PIN}, mk, \phi(c, \mathfrak{g})\rangle\rangle\ \big)\ \mid\\
&\quad \nu b_t.!\nu kbt.\big(\\
&\qquad B(bank, si, kbt, b_t)\ \mid\\
&\qquad \texttt{let}\ \mathsf{crt} := \langle\langle \texttt{MM}, \phi(b_t, \mathfrak{g})\rangle, \texttt{sig}(\langle \texttt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle\texttt{in}\\
&\qquad T_{\texttt{onhi}}(user, term, \texttt{vpk}(\chi_{\texttt{MM}})\,, \mathsf{crt}, kbt)\ \mid\\
&\qquad T_{\texttt{offhi}}(user, term, \texttt{vpk}(\chi_{\texttt{MM}})\,, \mathsf{crt}, kbt)\ \mid\\
&\qquad T_{\texttt{lo}}(term, \texttt{vpk}(\chi_{\texttt{MM}})\,, \mathsf{crt}, kbt)\big)\ \Big)
\end{aligned}
$$

Below we revisit role specifications in Fig. 4.4, 4.5, 4.6 and demonstrate the exact locations where we insert the events involved in injective correspondences from Fig. 4.16, 4.17.

$$
\begin{aligned}
&C(ch, c, pk_s, \mathtt{vsig}_{\mathtt{MM}}, \mathrm{PAN}, mk, \mathrm{PIN}) \triangleq \\
&\quad ch(z_1). \\
&\quad va.\ \mathtt{let} z_2 := \phi(a, \phi(c, \mathfrak{g}))\ \mathtt{in} \\
&\quad \overline{ch}\langle z_2 \rangle. \\
&\quad \mathtt{let}\ k_c := \mathtt{h}(\phi(a \cdot c, z_1))\ \mathtt{in} \\
&\quad ch(m). \\
&\quad \mathtt{let}\ \langle \langle \mathsf{MM}, y_B \rangle, \mathsf{MC}_s \rangle := \mathtt{dec}(k_c, m)\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{check}(\mathsf{MC}_s, pk_s) = \langle \mathsf{MM}, y_B \rangle\ \mathtt{then} \\
&\quad \mathtt{let}\ emcert = \{\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}_{\mathtt{MM}}) \rangle\}_{k_c}\ \mathtt{in} \\
&\quad \overline{ch}\langle emcert \rangle. \\
&\quad ch(x). \\
&\quad \mathtt{let}\ \langle \mathsf{TX}, \mathtt{uPin} \rangle := \mathtt{dec}(k_c, x)\ \mathtt{in} \\
&\quad \mathtt{let}\ \mathsf{AC} := \langle a, \mathrm{PAN}, \mathsf{TX} \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\ \mathsf{AC}^{\mathtt{ok}} := \langle a, \mathrm{PAN}, \mathsf{TX}, \mathtt{ok} \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\ \mathsf{AC}^{\perp} := \langle a, \mathrm{PAN}, \mathsf{TX}, \perp \rangle\ \mathtt{in} \\
&\quad \mathtt{let}\ k_{cb} := \mathtt{h}(\phi(a \cdot c, y_B))\ \mathtt{in} \\
&\quad \mathtt{if}\ \mathtt{uPin} = \perp\ \mathtt{then} \\
&\qquad \mathtt{let}\ eac := \{\{\langle \mathsf{AC}, \mathtt{h}(\langle \mathsf{AC}, mk \rangle) \rangle\}_{k_{cb}}\}_{k_c}\ \mathtt{in} \\
&\qquad \mathtt{ev{:}CRunningWithB}\ (eac) \\
&\qquad \mathtt{ev{:}CRunning}\ (z_1, z_2, m, emcert, x, eac) \\
&\qquad \overline{ch}\langle eac \rangle \\
&\quad \mathtt{else\ if}\ \mathtt{uPin} = \mathrm{PIN}\ \mathtt{then} \\
&\qquad \mathtt{let}\ eac := \{\langle \{\langle \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\langle \mathsf{AC}^{\mathtt{ok}}, mk \rangle) \rangle\}_{k_{cb}}, \mathtt{ok} \rangle\}_{k_c}\ \mathtt{in} \\
&\qquad \mathtt{ev{:}CRunningWithB}\ (\{\langle \mathsf{AC}^{\mathtt{ok}}, \mathtt{h}(\langle \mathsf{AC}^{\mathtt{ok}}, mk \rangle) \rangle\}_{k_{cb}}) \\
&\qquad \mathtt{ev{:}CRunning}\ (z_1, z_2, m, emcert, x, eac) \\
&\qquad \overline{ch}\langle eac \rangle \\
&\quad \mathtt{else} \\
&\qquad \mathtt{let}\ eac := \{\langle \{\langle \mathsf{AC}^{\perp}, \mathtt{h}(\langle \mathsf{AC}^{\perp}, mk \rangle) \rangle\}_{k_{cb}}, \perp \rangle\}_{k_c}\ \mathtt{in} \\
&\qquad \mathtt{ev{:}CRunningWithB}\ \left(\{\langle \mathsf{AC}^{\perp}, \mathtt{h}(\langle \mathsf{AC}^{\perp}, mk \rangle) \rangle\}_{k_{cb}}\right) \\
&\qquad \mathtt{ev{:}CRunning}\ (z_1, z_2, m, emcert, x, eac) \\
&\qquad \overline{ch}\langle eac \rangle
\end{aligned}
$$

Figure 4.18: Events in the card's role.

$T_{\texttt{onhi}}(user, ch, pk_{\texttt{MM}}, \mathsf{crt}, kbt) \triangleq$

$\nu\, \mathsf{TXdata}.$
$\texttt{let}\ \mathsf{TX} := \langle \mathsf{TXdata}, \texttt{hi} \rangle\ \texttt{in}$
$\nu t.\texttt{let} z_1 := \phi(t, \mathfrak{g})\ \texttt{in}$
$\overline{ch}\langle z_1 \rangle.$
$ch(z_2).$
$\texttt{let}\ k_t := \texttt{h}(\phi(t, z_2))\ \texttt{in}$
$\overline{ch}\langle \{\mathsf{crt}\}_{k_t} \rangle.$
$ch(n).$
$\texttt{let}\ \langle \mathsf{B}, \mathsf{B}_s \rangle := \texttt{dec}(k_t, n)\ \texttt{in}$
$\texttt{if}\ \texttt{vcheck}(\mathsf{B}_s, pk_{\texttt{MM}}) = \mathsf{B}\ \texttt{then}$
$\texttt{if}\ \mathsf{B} = z_2\ \texttt{then}$
$user(\mathsf{uPIN}).$
$\overline{ch}\langle \{\langle \mathsf{TX}, \perp \rangle\}_{k_t} \rangle.$
$ch(y).$
$\texttt{ev:TCommitWithC}\ (z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, \{\langle \mathsf{TX}, \perp \rangle\}_{k_t}, y)$
$\texttt{let}\ req = \{\langle \mathsf{TX}, z_2, \texttt{dec}(k_t, y), \mathsf{uPIN} \rangle\}_{kbt}\ \texttt{in}$
$\texttt{ev:TRunningWithBC}\ (req, z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, \{\langle \mathsf{TX}, \perp \rangle\}_{k_t}, y)$
$\overline{ch}\langle req \rangle.$
$ch(r).$
$\texttt{if}\ \texttt{dec}(kbt, r) = \langle \mathsf{TX}, \texttt{rtype} \rangle\ \texttt{then}$
$\quad \texttt{ev:TCommitWithBC}\ (req, r, z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, \{\langle \mathsf{TX}, \perp \rangle\}_{k_t}, y)$
$\quad \texttt{if}\ \texttt{rtype} = \texttt{accept}\ \texttt{then}$
$\quad\quad \texttt{ev:TAccept}\ (kbt, \mathsf{TX})$
$\quad\quad \overline{ch}\langle \texttt{auth} \rangle$

Figure 4.19: Events in the terminal's role.

$B(ch, si, kbt, b_t) \triangleq$
$ch(x).$
$\texttt{let}\ \langle \mathsf{TX}', z_2, \mathsf{EAC}, \mathsf{uPIN}\rangle := \texttt{dec}(kbt, x)$
$\texttt{let}\ k_{bc} := \texttt{h}(\phi(b_t, z_2))\ \texttt{in}$
$\texttt{let}\ \langle \mathsf{AC}, \mathsf{AC}_{hmac}\rangle = \texttt{dec}(k_{bc}, \mathsf{EAC})\texttt{in}$
$\texttt{let}\ \langle x_a, \mathsf{PAN}, \mathsf{TX}, \mathsf{pinV}\rangle = \mathsf{AC}\ \texttt{in}$
$\langle si, \mathrm{PAN}\rangle(\mathrm{PIN}, mk, pk_c).$
$\texttt{if}\ \texttt{h}(\langle \mathsf{AC}, mk\rangle) = \mathsf{AC}_{hmac}\ \texttt{then}$
$\texttt{if}\ \mathsf{TX} = \mathsf{TX}'\ \texttt{then}$
$\texttt{if}\ \phi(x_a, pk_c) = z_2$
$\texttt{let}\ \langle \mathsf{TXdata}, \mathsf{TXtype}\rangle := \mathsf{TX}'\ \texttt{in}$
$\texttt{if}\ \mathsf{TXtype} = \texttt{lo}\ \texttt{then}$
$\texttt{ev:BCommitWithC}\ (\mathsf{EAC})$
$\texttt{ev:BRunningWithT}\ (x, \{\langle \mathsf{TX}', \texttt{accept}\rangle\}_{kbt})$
$\texttt{ev:BCommitWithTC}\ (x)$
$\overline{ch}\langle\{\langle \mathsf{TX}', \texttt{accept}\rangle\}_{kbt}\rangle$
$\texttt{else if}\ \mathsf{TXtype} = \texttt{hi}\ \texttt{then}$
$\texttt{if}\ (\mathsf{pinV} = \texttt{ok}) \vee (\mathsf{uPIN} = \mathrm{PIN})\ \texttt{then}$
$\texttt{ev:BCommitWithC}\ (\mathsf{EAC})$
$\texttt{ev:BRunningWithT}\ (x, \{\langle \mathsf{TX}', \texttt{accept}\rangle\}_{kbt})$
$\texttt{ev:BCommitWithTC}\ (x)$
$\overline{ch}\langle\{\langle \mathsf{TX}', \texttt{accept}\rangle\}_{kbt}\rangle.$
$\texttt{else}$
$\texttt{ev:BReject}\ (kbt, \mathsf{TX}')$
$\texttt{ev:BCommitWithC}\ (\mathsf{EAC})$
$\texttt{ev:BRunningWithT}\ (x, \{\langle \mathsf{TX}', \texttt{reject}\rangle\}_{kbt})$
$\texttt{ev:BCommitWithTC}\ (x)$
$\overline{ch}\langle\{\langle \mathsf{TX}', \texttt{reject}\rangle\}_{kbt}\rangle.$

Figure 4.20: Events in the bank's's role.

The respective ProVerif file can be found in the repository [repb]. The code contains the full specification of the UTX protocol and of the correspondences, ensuring injective agreement and the additional functional properties from Fig. 4.17. It also contains queries corresponding to the expected secrecy of the private data that includes the master key between the card and the bank, of the PIN, and of the cryptogram and the reachability queries attesting that honest participants can successfully complete the protocol. All properties are successfully verified within 80 minutes.

4.3.7 Compromised scenarios

Finally, using ProVerif, we analyse two scenarios where the terminal accepts a potentially fake card. The code verifying these additional scenarios described below is provided in the directory *compromised* of the repository [repb].

Security under compromised terminals. Firstly, we support the point made at the *Cryptogram generation* part of Section 4.2.5 that even if a terminal neglects to perform the checks required to authenticate the card, the bank is still ensured that a valid card is executing a transaction. To model that, we remove the Verheul signature verification in the terminal's process. In that case, the first property that the terminal authenticates the card fails as expected, while others are preserved.

Security under compromised $\chi_{\mathtt{MM}}$. Secondly, we consider that the key $\chi_{\mathtt{MM}}$ is leaked, allowing attackers to manufacture cards passing the terminal's check by producing valid Verheul signatures. The verification outcome in this case is similar – the terminal-card agreement fails, making offline transactions insecure, while online transactions are still safe, i.e. the injective agreement involving the bank holds. Therefore, the payment system should notify terminal owners to stop accepting offline payments if $\chi_{\mathtt{MM}}$ has been compromised.

4.3.8 The estimate of the runtime performance

Concluding the analysis, we give a rough estimate of the runtime performance of the UTX protocol focusing on the card operations. Indeed since the terminal is a more powerful device than the card we expect its contribution to the runtime to be minuscule. We make our assessment based on the estimations reported in [MDHM18, DRHM17b] for the Multos Card ML3 supporting ECC scalar multiplication. The table below summarises the amount of time for individual operations performed by the card. As we expect the equality check and forming n-tuples operations to be negligible, we omit them in our calculation. Overall the numbers add up to 700ms per on-card computation per session. We expect that further optimisation and using more recent smart card platforms would lower this number within the current 500 *ms* recommendation [emv21].

Operation	$\phi(\cdot,\cdot)$	$\mathtt{h}(\cdot)$	$\mathtt{dec}(\cdot,\cdot)$	$\{\cdot\}.$	$\mathtt{check}(\cdot,\cdot)$
# of ops	6	3	2	3	1
ms per op	61	11	13	13	228

The numbers from the third line correspond to the 256-bit security level for ϕ and `check` operations, which are evaluated using the Barreto-Naehrig pairing-friendly curve since Verheul signatures are pairing-based, and ECDSA, respectively. To the best of our knowledge, there is no credible source for 256-bit security assessment for the rest, hence we use the available benchmarks – `dec` and $\{\ \}$ are evaluated using 128-bit key AES in CBC mode on 128-bit message, and, finally, `h` has been tested using SHA-256 on 128-bit message.

4.4 Summary and future work

In this chapter, we have identified in Sec. 4.1 the requirements for an unlinkable smartcard-based payments protocol and have demonstrated that at least one protocol satisfying these requirements (as established in Sec. 4.3) exists – the UTX protocol presented in Fig. 4.3.

We provide precisely three modes which agents should implement to process UTX payments. The modes of payment should be standardised and be common to all cards supporting UTX. This avoids cards being distinguished by implementation differences. This is in contrast to the current EMV standard, which, as we explain in Chapter 3 has many different modes of operation contained in over 2000 pages split into several books, as the particular *implementation* of the protocol serves as a coarse identity of the card. Moreover, having a concise, coherent, and linear presentation can improve the reliability of the system. Our message sequence chart in Fig. 4.3 and the applied π-calculus specification of UTX in Fig. 4.4, 4.5, 4.6 go some way towards this aim.

We admit that the set of functional requirements we have identified comprise the essential minimum for a useful unlinkable smartcard-based payment protocol. There is a variety of "nice to have" and even "essential" in one's view features that some EMV cards nowadays support. To incorporate such features in the UTX protocol while preserving its security and privacy properties may give rise to certain challenges.

An example of a feature that is straightforward to include is a dynamic limit *LIM*, which defines a threshold above which high-value transactions require a PIN. Such limit can be updated without reissuing the card by including it in the certificate that the terminal presents $\langle\langle \mathtt{MM}, LIM, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, LIM, \phi(b_t, \mathfrak{g})\rangle, s)\rangle$.

An example of a feature that would require a non-incremental update to UTX is introducing relay protection [RCN+22] which would mitigate the situation where a high-value online transaction is compromised via relay attack if the PIN is exposed as we mention in Sec. 4.3.1.

Relay protection would also be essential in combination with another "essential" feature, PIN tries counter, that should limit the number of incorrect attempts to enter the PIN. An active attacker near the honest terminal waiting to process an online high-value transaction can relay messages to the card and can enter the PIN incorrectly enough times to exceed the limit thereby blocking the card from any online transactions. More specifically, this attack forces the card to obtain a coarse identity – the fact that it cannot participate in online transactions. Then to identify such cards an attacker should yet again relay communication between the card

and an online terminal – transactions would be declined with an explicit reason of the PIN tries exceeded. Relay protection would mitigate against this scenario as it would make it impossible to enter the PIN remotely since the user should be physically close and, thereby, aware of someone entering the PIN. Notice that in case of offline high-value purchases an active attacker could not exceed PIN tries, as asking the PIN offline we always assume a contact transaction, i.e. the user is already close.

Finally, a proof of concept implementation is needed to clarify that UTX is fast enough to be usable contactlessly and verify the estimate presented in Section 4.3.8.

Chapter 5

Conclusion

In the age of automated data collection, privacy is essential when it comes to fundamental rights such as freedom, security, and dignity. It is only a matter of time before private data will be harvested with the purpose of manipulation and surveillance if there is a technical capacity to do so. Billions of people are at threat of being actively tracked by an unauthorised party just by the fact that they carry a payment card. As we explain in the introduction to Chapter 3 tracking the movements of the cardholder is enabled by the protocol the majority of card use, the EMV protocol. In this dissertation, we have addressed this by offering the UTX (Fig. 4.3) protocol, a smartcard-based payment protocol that satisfies the initial functional and security requirements of the current EMV (Sec. 4.1.1, 4.1.2) and at the same time provides strong privacy guarantees to cardholders (Sec. 4.1.3), which include anti-tracking. We are confident making such claims since we have proven that UTX delivers what it intends to deliver using formal methods. We have used a state-of-the-art ProVerif tool to specify and verify the security properties of UTX (Sec. 4.3.6), and applied the methodology described in Chapter 2 to reason about privacy in UTX (Sec. 4.3.3).

Our UTX protocol is not the first attempt to introduce privacy in card payments, in fact the developers of EMV themselves were aware of privacy issues at least since 2013, when they proposed to introduce encryption in the next generation of EMV by running the BDH key agreement (Fig. 3.14), which we have investigated (Sec. 3.3), as the first phase of the transaction. We have demonstrated, that BDH would not protect the cardholder against tracking since the card running BDH exposes its strong form of identity to active attackers (Sec. 3.3.2). We have demonstrated how to use a generic signature scheme that respects blinding to improve the BDH protocol (Sec. 3.5.1), thereby achieving UBDH (Fig. 3.17), that protects the cardholder against tracking (Theorem 7) without compromising the initial authentication requirement of BDH (Sec. 3.5.3). Moreover it serves as a basis to UTX. To support our solution, we have pointed out that at least one existing signature scheme meets our requirements, namely Verheul signatures.

The UTX protocol fills the vacuum left by the withdrawal of the next generation of EMV in October 2019, when the developers officially announced they were abandoning their efforts to enhance privacy [emv19] and, in fact, strengthens the initial security requirements of EMV (Sec. 4.1.2). In particular, we have requested that the application cryptogram is secret and can only be processed by a legitimate acquiring bank. This requirement is addressed in UTX by using the certified bank's public key that the card obtains at the beginning of each transaction and uses to encrypt the cryptogram (Sec. 4.2.4, *Cryptogram generation* in Sec. 4.2.5). We have also

summarised how we have utilised ProVerif to prove that UTX satisfies all security requirements we have identified (Fig. 4.16, 4.17).

We have chosen unlinkability as our privacy requirement and have highlighted that the fingerprint of the card, comprising coarse identities of a card, that permits groups of cards to be tracked, should be minimised (Sec. 4.1.3). Since strong identities compromise unlinkability, we have hidden any strong identity of the card by utilising signatures satisfying blinding condition (Fig. 3.16) to make the validity signature distinct in every session (*Validity check* in Sec. 4.2.5), and by encrypting the cryptogram that contains the card number PAN to hide it from the terminal (*Cryptogram generation* in Sec. 4.2.5). We have minimised the card's fingerprint by introducing certificates that reveal that the card is valid for the current and previous months without revealing the expiry date (*Keys required to set up* Unlinkable, *Validity check* in Sec. 4.2.5). If payment systems agree on a common certification authority, we may reduce the card's fingerprint further by introducing the Unlinkable application (Sec. 4.2.1). We then have proven that these measures indeed achieve unlinkability in UTX (Theorem 8).

Finally, we have justified the use of quasi-open bisimilarity (Def. 5) to define the unlinkability of BDH/UBDH and UTX protocols (Def. 9, 14) by the following pros.

- Quasi-open bisimilarity captures a realistic attacker which is capable of observing not only the exchanged data, but also making dynamic decisions to manipulate the execution and, thereby evaluating the behaviour of the system (Sec. 2.2.1).
- The verification outcome is always decisive, thanks to the completeness of the modal logic intuitionistic $\mathcal{FM}$ (Sec. 2.2.3) characterising quasi-open bisimilarity (Theorem 2).
- It is possible to reduce the amount of work needed for verification, thanks to quasi-open bisimilarity being congruence (Theorem 3, 4).

In the following revision of the research questions we highlight how exactly we have utilised the pros above to reason about privacy in studied protocols

Research Question 1

Can we identify the requirements for equivalence notion suitable for modelling indistinguishability properties of security protocols?

At the start of Chapter 2, we have made several observations about a useful equivalence notion.

- Since we target the effective verification of indistinguishability/privacy properties, we have demanded that the equivalence failure always leads to an executable attack (Fig. 2.2). With the private server example (Fig. 2.11) we have demonstrated that the equivalence that does not have such property may lead to incorrect verification results.

- Since there are real-world situations when a protocol should satisfy a natural privacy requirement when there is no shared secret between the parties, we have demanded that the target equivalence should enable compositionality (Fig. 2.3). While being a requirement, compositionality has a nice bonus as it also supports effective verification – it might be possible to reduce the amount of work, as we have illustrated by the example of authentication to an untrusted device (Sec. 2.2.4) and the discussion about the BDH protocol (Sec. 3.3).
- Since we do not want to underestimate the attacker's capabilities, as the verification result would be, in this case, useless, we have demanded the target equivalence be some form of bisimilarity that explicitly endows an attacker with the ability to make decisions dynamically. We have supported this point with the story about the BAC protocol in the introduction – this protocol is only correct if an attacker can follow the pre-defined sequence of actions.

Thus we conclude that we have identified natural requirements for the equivalence notion and can start answering the follow-up question.

Research Question 2

Can we identify a canonical equivalence notion satisfying the demands identified?

We have picked quasi-open bisimilarity (Def. 5), the coarsest (Theorem 5) equivalence relation from the region identified in Fig.2.4. We have explained precisely the capabilities of an attacker it captures (Sec. 2.2.1), among which the most innovative is the ability to consider systematically different ways to "stage" an attack by instantiating and grounding free variables. We have drawn special attention to the notion of proof certificates (Sec. 2.2.6) serving the evidence that can be checked independently when there is either an attack (hence the certificate is a modal logic formula that one side of the relation fails to satisfy) or there is no attack (hence the certificate is a quasi-open bisimulation between the idealised specification and the real-world implementation) on the protocol.

Research Question 3

Can we reason effectively about security protocols using the equivalence identified?

In definitions of unlinkability of BDH/UBDH and UTX protocols (Def. 9, 14), we have employed the fact that quasi-open bisimilarity is a congruence relation which has helped to simplify verification – in the first case, we have withdrawn terminals from the picture as they share no secret with cards if we consider the key agreement in isolation; in the second case, we have automatically extended our analysis to the case where each payment system provides its own unlinkable payments (Corollary 1), and demonstrated the it is enough to verify a subsystem comprising only cards and terminals in case no no transaction requires the PIN (Lemma 3).

We have also demonstrated by the respective proofs of unlinkability of UBDH and UTX (Theorems 7, 8) that the verification can be broken down into two major phases – preliminary phase comprising ingenious steps and the mechanical phase comprising checking the conditions for quasi-open bisimulation. We would like to stop here on the "hard" preliminary phase.

The first ingenious step is to construct a quasi-open bisimulation (Fig. 3.18, 4.10) for which we have several informal hints.

- We pair states based on the number of started sessions for each role.
- We define all possible "tail" subprocesses of processes specifying the roles in the protocol.
- We use these subprocesses to define partitions of the set of sessions corresponding to different phases of the protocol execution and use these partitions to parametrise the related states.

The second ingenious step is message theory-dependent and consists of defining frame normalisations (Def. 13, 15). The main hint is the following.

- Find all ways to apply destructors to messages in the range of a frame, thereby finding all recipes for messages that cannot be simplified further.

We have demonstrated that it is possible to construct frame normalisations to then prove static equivalence in the presence of a sophisticated equational theory (Lemmas 1, 2). However, the limitations of our approach it is yet to be discovered, which we consider the first step for future work.

In conclusion we should say that we still expect stakeholders to take the possibility of making payments unlinkable seriously. Not only because awareness of privacy issues is growing, and supporting legislation, such as GDPR, is emerging, but also because methods, such as ours, which enable the verification of privacy properties for the unbounded case, are improving. The UTX protocol, proven to satisfy the identified security and privacy requirements, demonstrates the feasibility of privacy-preserving smartcard-based payments.

The ultimate research goal for future work is the automation of our approach since handwritten proofs are not only error-prone, they cannot be independently checked by a non-professional. The nature of the proofs we have provided identifies what we think is feasible for a future tool – a tool that consumes a man-made proof certificate and verifies it. In a world where the lesson in the protocol design that we give in Chapter 4 is learned, the developers of the protocol produce such a certificate, and its inclusion in the documentation is a necessary step for further implementation. We view this dissertation as a step towards this idealised situation.

Bibliography

[ABF17] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The applied pi calculus: Mobile values, new names, and secure communication. *Journal of the ACM (JACM)*, 65(1):1–41, 2017. doi:10.1145/3127586.

[AC06] Martín Abadi and Véronique Cortier. Deciding knowledge in security protocols under equational theories. *Theoretical Computer Science*, 367(1-2):2–32, 2006. doi:10.1016/j.tcs.2006.08.032.

[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)*, pages 107–121. IEEE, 2010. doi:10.1109/CSF.2010.15.

[AF01] Martín Abadi and Cédric Fournet. Mobile values, new names, and secure communication. In *Symposium on Principles of Programming Languages (POPL)*, pages 104–115. ACM SIGPLAN-SIGACT, 2001. doi:10.1145/360204.360213.

[AF04] Martín Abadi and Cédric Fournet. Private authentication. *Theoretical Computer Science*, 322(3):427–476, 2004. doi:10.1016/j.tcs.2003.12.023.

[AFN17] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. Intruder deduction problem for locally stable theories with normal forms and inverses. *Theoretical Computer Science*, 672:64–100, 2017. doi:10.1016/j.tcs.2017.01.027.

[AHT17] Ki Yung Ahn, Ross Horne, and Alwen Tiu. A characterisation of open bisimilarity using an intuitionistic modal logic. In *International Conference on Concurrency Theory (CONCUR)*, pages 7:1–7:17, 2017. doi:10.4230/LIPIcs.CONCUR.2017.7.

[AHT21] Ki Yung Ahn, Ross Horne, and Alwen Tiu. A characterisation of open bisimilarity using an intuitionistic modal logic. *Logical Methods in Computer Science*, 17:2:1–2:40, 2021. doi:10.46298/lmcs-17(3:2)2021.

[AR02] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of cryptology*, 15(2):103–127, 2002. doi:10.1007/s00145-001-0014-7.

[B+01] Bruno Blanchet et al. An efficient cryptographic protocol verifier based on prolog rules. In *Computer Security Foundations Symposium (CSF)*, pages 82–96. IEEE, 2001. doi:10.1109/CSFW.2001.930138.

[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. doi:10.1016/j.jlap.2007.06.002.

[BC14] Paolo Baldan and Silvia Crafa. A logic for true concurrency. *Journal of the ACM*, 61(4):24:1–24:36, 2014. doi:10.1145/2629638.

[BCD14] Sergiu Bursuc, Hubert Comon-Lundh, and Stéphanie Delaune. Deducibility constraints and blind signatures. *Information and Computation*, 238:106–127, 2014. doi:10.1016/j.ic.2014.07.006.

[BCDD20] Ioana Boureanu, Tom Chothia, Alexandre Debant, and Stéphanie Delaune. Security analysis and implementation of relay-resistant contactless payments. In *Conference on Computer and Communications Security (CCS)*, pages 879–898. ACM SIGSAC, 2020. doi:10.1145/3372297.3417235.

[BCM+14] Mike Bond, Omar Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. Chip and skim: Cloning EMV cards with the pre-play attack. In *Symposium on Security and Privacy (S&P)*, pages 49–64. IEEE, 2014. doi:10.1109/SP.2014.11.

[BCM+15] Mike Bond, Marios O. Choudary, Steven J. Murdoch, Sergei Skorobogatov, and Ross Anderson. Be prepared: The EMV preplay attack. In *Symposium on Security and Privacy (S&P)*, pages 56–64. IEEE, 2015. doi:10.1109/MSP.2015.24.

[BDM20] David Baelde, Stéphanie Delaune, and Solène MOREAU. A method for proving unlinkability of stateful protocols. In *Computer Security Foundations Symposium (CSF)*. IEEE, 2020. doi:10.3233/JCS-171070.

[BHJ+10] Lejla Batina, Jaap-Henk Hoepman, Bart Jacobs, Wojciech Mostowski, and Pim Vullers. Developing efficient blinded attribute certificates on smart cards via pairings. In *International Conference on Smart Card Research and Advanced Applications (CARDIS)*, pages 209–222. Springer, 2010. doi:10.1007/978-3-642-12510-2_15.

[bioa] Idex biometrics. URL: https://www.idexbiometrics.com.

[biob] Mastercard biometric card. URL: https://www.mastercard.us/en-us/business/overview/safety-and-security/authentication-services/biometrics/biometrics-card.html.

[Bla04] Bruno Blanchet. Automatic proof of strong secrecy for security protocols. In *Symposium on Security and Privacy (S&P)*, pages 86–100. IEEE, 2004. doi:10.1109/SECPRI.2004.1301317.

[Bla09] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009. doi:10.3233/JCS-2009-0339.

[Bla16] Bruno Blanchet. Modeling and verifying security protocols with the applied pi-calculus and proverif. *Foundations and Trends in Privacy and Security*, 1(1-2):1–135, 2016. doi:10.1561/3300000004.

[BN99] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge university press, 1999.

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography*, pages 319–331. Springer Berlin Heidelberg, 2006. doi:10.1007/11693383_22.

[BNP01] Michele Boreale, Rocco De Nicola, and Rosario Pugliese. Proof techniques for cryptographic processes. *SIAM Journal of Computing*, 31(3):947–986, 2001. doi:10.1137/S0097539700377864.

[BST21a] David Basin, Ralf Sasse, and Jorge Toro-Pozo. Card brand mixup attack: Bypassing the PIN in non-Visa cards by using them for Visa transactions. In *USENIX Security Symposium*, pages 179–194. USENIX Association, 2021. URL: https://www.usenix.org/conference/usenixsecurity21/presentation/basin.

[BST21b] David A. Basin, Ralf Sasse, and Jorge Toro-Pozo. The EMV standard: Break, Fix, Verify. In *Symposium on Security and Privacy (S&P)*, pages 1766–1781. IEEE, 2021. doi:10.1109/SP40001.2021.00037.

[BSWW13] Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. An analysis of the EMV channel establishment protocol. In *Conference on Computer and Communications Security (CCS)*, pages 373–386. ACM SIGSAC, 2013. doi:10.1145/2508859.2516748.

[cal21] CALYPSO Handbook. Technical report, Calypso Networks Association, 2021. Accessed: 01-02-2023. URL: https://calypsostandard.net/specifications/public-documents/150-100324-calypso-handbook.

[cc17] Common criteria for information technology security evaluation. part 2: Security functional components. Technical report, 2017. Version 3.1, Revision 5, CCMB-2017-04-002.

[CCCK16] Rohit Chadha, Vincent Cheval, Ştefan Ciobâcă, and Steve Kremer. Automated verification of equivalence properties of cryptographic protocols. *Transactions on Computational Logic*, 17(4):23:1–23:32, 2016. doi:10.1145/2926715.

[CCLD17] Vincent Cheval, Hubert Comon-Lundh, and Stéphanie Delaune. A procedure for deciding symbolic equivalence between sets of constraint systems. *Information and Computation*, 255(Part 1):94–125, 2017. doi:10.1016/j.ic.2017.05.004.

[CDD17] Véronique Cortier, Antoine Dallon, and Stéphanie Delaune. SAT-Equiv: An efficient tool for equivalence properties. In *Computer Security Foundations Symposium (CSF)*, pages 481–494. IEEE, 2017. doi:10.1109/CSF.2017.15.

[CdRS18] Tom Chothia, Joeri de Ruiter, and Ben Smyth. Modelling and analysis of a hierarchy of distance bounding attacks. In *USENIX Security Symposium (USENIX)*, pages 1563–1580. USENIX Association, 2018. URL: https://www.usenix.org/conference/usenixsecurity18/presentation/chothia.

[CKR18] Vincent Cheval, Steve Kremer, and Itsaka Rakotonirina. DEEPSEC: Deciding equivalence properties in security protocols theory and practice. In *Symposium on Security and Privacy (S&P)*, pages 529–546. IEEE, 2018. doi:10.1109/SP.2018.00033.

[CM12] Cas Cremers and Sjouke Mauw. *Operational Semantics and Verification of Security Protocols*. Springer, 2012. doi:10.1007/978-3-540-78636-8.

[CS11] Véronique Cortier and Ben Smyth. Attacking and fixing Helios: An analysis of ballot secrecy. In *Computer Security Foundations Symposium (CSF)*, pages 297–311. IEEE, 2011. doi:10.1109/CSF.2011.27.

[CSS15] Tom Chothia, Ben Smyth, and Chris Staite. Automatically checking commitment protocols in proverif without false attacks. In *International Conference on Principles of Security and Trust (POST)*, pages 137–155. Springer, 2015.

[Del18] Stéphanie Delaune. Analysing privacy-type properties in cryptographic protocols. In *International Conference on Formal Structures for Computation and Deduction (FSCD)*, pages 1:1–1:21. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.FSCD.2018.1.

[DM07] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX Security Symposium (USENIX)*, pages 7:1–7:16. USENIX Association, 2007.

[DNV95] Rocco De Nicola and Frits Vaandrager. Three logics for branching bisimulation. *Journal of the ACM (JACM)*, 42(2):458–487, 1995. doi:10.1145/201019.201032.

[Dob05] Ernst-Erich Doberkat. Stochastic relations: congruences, bisimulations and the Hennessy-Milner theorem. *SIAM Journal on Computing*, 35(3):590–626, 2005. doi:10.1137/S009753970444346X.

[DP03] Josée Desharnais and Prakash Panangaden. Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. *Journal of Logic and Algebraic Programming*, 56(1):99–115, 2003. doi:10.1016/S1567-8326(02)00068-1.

[DRHM17a] Petr Dzurenda, Sara Ricci, Jan Hajny, and Lukas Malina. Performance analysis and comparison of different elliptic curves on smart cards. In *Conference on Privacy, Security and Trust (PST)*, pages 365–36509. IEEE, 2017. doi:10.1109/PST.2017.00050.

[DRHM17b] Petr Dzurenda, Sara Ricci, Jan Hajny, and Lukas Malina. Performance analysis and comparison of different elliptic curves on smart cards. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 365–36509, 2017. doi:10.1109/PST.2017.00050.

[DY83] Danny Dolev and Andrew Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983. doi:10.1109/TIT.1983.1056650.

[emv] Google scholar. https://scholar.google.com/scholar?cites=18303427262500543559&as_sdt=2005&sciodt=0,5. Accessed: 03-02-2023.

[emv11] EMV Integrated Circuit Card Specifications for Payment Systems. Books 1-4. Technical report, EMVCo LLC, 2011. Accessed: 26-08-2021. URL: https://www.emvco.com/document-search/.

[emv19] EMVCo Statement – The Advancement of EMV Chip Specifications. Technical report, EMVCo LLC, 2019. Accessed: 23-09-2020. URL: https://www.emvco.com/wp-content/uploads/documents/2nd-Gen-External-Statement-FINAL.pdf.

[emv21] EMV Contactless Specifications for Payment Systems. Book A. Technical report, EMVCo LLC, 2021. Accessed: 20-09-2021. URL: https://www.emvco.com/document-search/.

[emv22] Worldwide EMV Deployment Statistics, 2022. Accessed: 26-01-2023. URL: https://www.emvco.com/about-us/worldwide-emv-deployment-statistics/.

[EPFB13] Maximilian Engelhardt, Florian Pfeiffer, Klaus Finkenzeller, and Erwin Biebl. Extending ISO/IEC 14443 type a eavesdropping range using higher harmonics. In *European Conference on Smart Objects, Systems and Technologies (SmartSysTech)*, pages 1–8. IEEE, 2013. URL: https://ieeexplore.ieee.org/document/6525242.

[Fag87] François Fages. Associative-commutative unification. *Journal of Symbolic Computation*, 3(3):257–275, 1987. doi:10.1016/S0747-7171(87)80004-4.

[FHMS19] Ihor Filimonov, Ross Horne, Sjouke Mauw, and Zach Smith. Breaking unlinkability of the ICAO 9303 standard for e-passports using bisimilarity. In *ESORICS*, volume 11735 of *LNCS*, pages 577–594. Springer, 2019. doi:10.1007/978-3-030-29959-0_28.

[for21] How The Pandemic Made Contactless Payments The New Normal, Forbes, 2021. Accessed: 07-07-2022. URL: https://www.forbes.com/sites/forbestechcouncil/2021/04/15/how-the-pandemic-made-contactless-payments-the-new-normal/?sh=e21bb183b7a4.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptology*, pages 224–280, 2010. doi:10.1007/s00145-009-9048-z.

[ger22] German police used a tracing app to scout crime witnesses, The Washington Post, 2022. Accessed: 25-01-2022. URL: https://www.washingtonpost.com/world/2022/01/13/german-covid-contact-tracing-app-luca/.

[GZZH14] Yanfei Guo, Zhenfeng Zhang, Jiang Zhang, and Xuexian Hu. Security analysis of EMV channel establishment protocol in an enhanced security model. In *International Conference on Information and Communications Security (ICICS)*, volume 8958 of *LNCS*, pages 305–320. Springer, 2014. doi:10.1007/978-3-319-21966-0_22.

[HALT18] Ross Horne, Ki Yung Ahn, Shang-wei Lin, and Alwen Tiu. Quasi-open bisimilarity with mismatch is intuitionistic. In *Symposium on Logic in Computer Science (LICS)*, pages 26–35. ACM/IEEE, 2018. doi:10.1145/3209108.3209125.

[HBD16] Lucca Hirschi, David Baelde, and Stéphanie Delaune. A method for verifying privacy-type properties: the unbounded case. In *Symposium on Security and Privacy (S&P)*, pages 564–581. IEEE, 2016. doi:10.1109/SP.2016.40.

[HDPdR15] René Habraken, Peter Dolron, Erik Poll, and Joeri de Ruiter. An RFID skimming gate using higher harmonics. In *Radio Frequency Identification*, pages 122–137. Springer International Publishing, 2015. doi:10.1007/978-3-319-24837-0_8.

[HM85] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM (JACM)*, 32(1):137–161, 1985. doi:10.1145/2455.2460.

[HM21] Ross Horne and Sjouke Mauw. Discovering ePassport vulnerabilities using bisimilarity. *Logical Methods in Computer Science*, 17(2):24:1–24:52, 2021. doi:10.23638/LMCS-17(2:24)2021.

[HMY21] Ross Horne, Sjouke Mauw, and Semen Yurkov. Compositional analysis of protocol equivalence in the applied π-calculus using quasi-open bisimilarity. In *Theoretical Aspects of Computing (ICTAC)*, pages 235–255. Springer International Publishing, 2021. doi:10.1007/978-3-030-85315-0_14.

[HMY23] Ross Horne, Sjouke Mauw, and Semen Yurkov. When privacy fails, a formula describes an attack: A complete and compositional verification method for the applied *pi*-calculus. *Theoretical Computer Science*, 2023. doi:10.1016/j.tcs.2023.113842.

[Hoa85] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[iso18] Cards and security devices for personal identification — contactless proximity objects — part 3: Initialization and anticollision. Technical Report 14443-3, ISO/IEC, 2018. URL: https://www.iso.org/standard/73598.html.

[KK16] Steve Kremer and Robert Künnemann. Automated analysis of security protocols with global state. *Journal of Computer Security*, 24(5):583–616, 2016. doi:10.3233/JCS-160556.

[KR05] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *Programming Languages and Systems: European Symposium on Programming (ESOP at ETAPS)*, volume 3444 of *LNCS*, pages 186–200. Springer-Verlag, 2005. doi:10.1007/978-3-540-31987-0_14.

[Lau02] Peeter Laud. Encryption cycles and two views of cryptography. In *Nordic Workshop on Secure IT Systems (NORDSEC)*, volume 31, pages 85–100. CiteSeer, 2002. doi:10.1007/s00145-001-0014-7.

[Low96] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Proc. Workshop on Tools and Algorithms for Construction and Analysis of Systems (TACAS)*, volume 1055 of *LNCS*, pages 147–166. Springer-Verlag, 1996.

[Low97] Gavin Lowe. A hierarchy of authentication specifications. In *Computer Security Foundations Workshop*, pages 31–43. IEEE, 1997. doi:10.1109/CSFW.1997.596782.

[MDAB10] Steven J. Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. Chip and PIN is broken. In *Symposium on Security and Privacy (S&P)*, pages 433–446. IEEE, 2010. doi:10.1109/SP.2010.33.

[MDHM18] Lukas Malina, Petr Dzurenda, Jan Hajny, and Zdenek Martinasek. Assessment of cryptography support and security on programmable smart cards. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–5, 2018. doi:10.1109/TSP.2018.8441334.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, Part I and II. *Information and Computation*, 100(1):1–100, 1992. doi:10.1016/0890-5401(92)90008-4.

[MPW93] Robin Milner, Joachim Parrow, and David Walker. Modal logics for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993. doi:10.1016/0304-3975(93)90156-N.

[MSCB13] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The Tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013. doi:10.1007/978-3-642-39799-8_48.

[MSTPTR18] Sjouke Mauw, Zach Smith, Jorge Toro-Pozo, and Rolando Trujillo-Rasua. Distance-bounding protocols: Verification without time and location. In *Symposium on Security and Privacy (S&P)*, pages 549–566. IEEE, 2018. doi:10.1109/SP.2018.00001.

[NGFR08] David R Novotny, Jeffrey R Guerrieri, Michael Francis, and Kate Remley. HF RFID electromagnetic emissions and performance. In *International Symposium on Electromagnetic Compatibility*, pages 1–7. IEEE, 2008. doi:10.1109/ISEMC.2008.4652133.

[NS78] Roger M Needham and Michael D Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[ove14] EMV next generation. next generation kernel system architecture overview. Technical report, EMVCo LLC, 2014.

[pas15] Machine readable travel documents. part 11: Security mechanisms for MRTDs. Technical Report Doc 9303. Seventh Edition, International Civil Aviation Organization (ICAO), 2015. URL: https://www.icao.int/publications/Documents/9303_p11_cons_en.pdf.

[PBE+21] Joachim Parrow, Johannes Borgström, Lars-Henrik Eriksson, Ramunas Gutkovas, and Tjark Weber. Modal logics for nominal transition systems. *Logical Methods in Computer Science*, 17(1):6:1–6:49, 2021. doi:10.23638/LMCS-17(1:6)2021.

[PFB12] Florian Pfeiffer, Klaus Finkenzeller, and Erwin Biebl. Theoretical limits of ISO/IEC 14443 type A RFID eavesdropping attacks. In *European Conference on Smart Objects, Systems and Technologies (SmartSysTech)*, pages 1–9. IEEE, 2012. URL: https://ieeexplore.ieee.org/document/6468861.

[Plo72] Gordon Plotkin. Building-in equational theories. *Machine intelligence*, 7:73–90, 1972.

[RCN+22] Andreea-Ina Radu Radu, Tom Chothia, Christopher JP Newton, Ioana Boureanu, and Liqun Chen. Practical EMV relay protection. In *Symposium on Security and Privacy (S&P)*, pages 1737–1756. IEEE, 2022. doi:10.1109/SP46214.2022.9833642.

[repa] ProVerif model of BDH and UBDH. Accessed: 25-02-2023. URL: https://github.com/ubdhutx/UBDH.

[repb] ProVerif model of UTX. Accessed: 18-03-2023. URL: https://github.com/ubdhutx/UTX.

[rfc12] EMV ECC key establishment protocols. Rfc until 28th january 2013, EMVCo LLC, 2012. Accessed: 01-04-2020. URL: http://www.emvco.com/specifications.aspx?id=243.

[RS99] Peter YA Ryan and Steve A Schneider. Process algebra and non-interference. In *Computer Security Foundations Workshop*, pages 214–227. IEEE, 1999. doi:10.1109/CSFW.1999.779775.

[rsf21] European Court of Human Rights admits RSF complaint against the BND's mass surveillance, Reporters Without Borders, 2021. Accessed: 25-01-2022. URL: https://rsf.org/en/news/european-court-human-rights-admits-rsf-complaint-against-bnds-mass-surveillance.

[RSG+01] Peter Ryan, Steve A Schneider, Michael Goldsmith, Gavin Lowe, and Bill Roscoe. *The modelling and analysis of security protocols: the CSP approach*. Addison-Wesley Professional, 2001.

[San96] Davide Sangiorgi. A theory of bisimulation for the π-calculus. *Acta Informatica*, 33(1):69–97, Feb 1996. doi:10.1007/s002360050036.

[Sim04] Alex Simpson. Sequent calculi for process verification: Hennessy–Milner logic for an arbitrary GSOS. *Journal of Logic and Algebraic Programming*, 60-61:287–322, 2004. doi:10.1016/j.jlap.2004.03.004.

[Sti81] Mark E Stickel. A unification algorithm for associative-commutative functions. *Journal of the ACM (JACM)*, 28(3):423–434, 1981. doi:10.1145/322261.322262.

[SW01a] Davide Sangiorgi and David Walker. On barbed equivalences in π-calculus. In Kim G. Larsen and Mogens Nielsen, editors, *International Conference on Concurrency Theory (CONCUR)*, volume 2154 of *LNCS*, pages 292–304. Springer, 2001. doi:10.1007/3-540-44685-0_20.

[SW01b] Davide Sangiorgi and David Walker. *π-Calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[TNH16] Alwen Tiu, Nam Nguyen, and Ross Horne. SPEC: An equivalence checker for security protocols. In *Asian Symposium on Programming Languages and Systems (APLAS)*, volume 10017 of *LNCS*, pages 87–95. Springer, 2016. doi:10.1007/978-3-319-47958-3_5.

[vdBOYPdR16] Jordi van den Breekel, Diego A Ortiz-Yepes, Erik Poll, and Joeri de Ruiter. EMV in a nutshell. Technical report, 2016. URL: https://www.cs.ru.nl/~erikpoll/papers/EMVtechreport.pdf.

[Ver01] Eric R Verheul. Self-blindable credential certificates from the Weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *LNCS*, pages 533–551. Springer, 2001. doi:10.1007/3-540-45682-1_31.

[vG93] Rob J van Glabbeek. The linear time-branching time spectrum II. In *International Conference on Concurrency Theory (CONCUR)*, LNCS, pages 66–81. Springer, 1993. doi:10.1007/3-540-57208-2_6.

[VG01] Rob J Van Glabbeek. The linear time-branching time spectrum I. The semantics of concrete, sequential processes. In *Handbook of process algebra*, pages 3–99. Elsevier, 2001. doi:10.1016/b978-044482830-9/50019-9.

Curriculum Vitae

2019 – 2023	Ph.D. student, University of Luxembourg, Luxembourg
2010 – 2012	M.Sc. in Economics, Higher School of Economics, Moscow, Russia.
2008 – 2010	Ed.S. in Mathematics, Kuzbass State Pedagogical Academy, Novokuznetsk, Russia.
2004 – 2008	Undergraduate in Mathematics, Novosibirsk State University, Novosibirsk, Russia.

Born on April 30, 1987, Novokuznetsk, USSR.