Provably Unlinkable Smart Card-based Payments

Sergiu Bursuc
University of Luxembourg
Esch-sur-Alzette, Luxembourg
sergiu.bursuc@uni.lu

Ross Horne
University of Luxembourg
Esch-sur-Alzette, Luxembourg
University of Strathclyde
Glasgow, UK
ross.horne@strath.ac.uk

Sjouke Mauw
University of Luxembourg
Esch-sur-Alzette, Luxembourg
sjouke.mauw@uni.lu

Semen Yurkov
University of Luxembourg
Esch-sur-Alzette, Luxembourg
semen.yurkov@uni.lu

ABSTRACT

The most prevalent smart card-based payment method, EMV, currently offers no privacy to its users. Transaction details and the card number are sent in cleartext, enabling the profiling and tracking of cardholders. Since public awareness of privacy issues is growing and legislation, such as GDPR, is emerging, we believe it is necessary to investigate the possibility of making payments anonymous and unlinkable without compromising essential security guarantees and functional properties of EMV. This paper draws attention to trade-offs between functional and privacy requirements in the design of such a protocol. We present the UTX protocol – an enhanced payment protocol satisfying such requirements, and we formally certify key security and privacy properties using techniques based on the applied π-calculus.

CCS CONCEPTS

• **Security and privacy** → **Privacy-preserving protocols**; **Logic and verification**; **Privacy protections**.

KEYWORDS

protocol design, payment protocols, security analysis

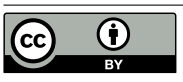
ACM Reference Format:
Sergiu Bursuc, Ross Horne, Sjouke Mauw, and Semen Yurkov . 2023. Provably Unlinkable Smart Card-based Payments. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23), November 26–30, 2023, Copenhagen, Denmark.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3576915.3623109

1 INTRODUCTION

As a payment method, EMV came into place in the mid-1990s to replace magstripe cards as they are incapable of computation and easy to clone. The EMV standard [1] is a series of documents that specify how exactly payments should be done with the main focus on card-terminal communication. This specification 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. It was shown several times that some configurations are not secure [11, 21, 31, 33], thus to achieve the primary goal of EMV, the safety of money, one should carefully select a secure configuration.

On the other hand, currently, the privacy of payments is not an explicit requirement of EMV. To this day the communication between the card and the terminal is not encrypted. Valuable sensitive data 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, the card presents its PAN – a strong form of identity – to *any* device that asks. Nearby smartphones supporting NFC and antennas [25], installed, e.g., at a doorway or by a seat on public transport, are examples of active attackers that can power up cards without cardholders being aware. After being powered-up, a card engages in what it thinks is a legitimate EMV session during which the PAN is transmitted. This enables an attacker to track the movements of anyone who holds a payment card by forcing the card to run a session and obtaining the card's permanent identity, even *without a genuine EMV transaction involved.* Hence, active attackers capable of initiating communication with cards using an unauthorised device should be part of the threat model when privacy is among our concerns.

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 [7, 8]. EMVCo, a consortium of payment processing companies that develops the EMV standard, is aware that privacy issues are present in EMV and have proposed in the next generation of EMV (EMV 2nd Gen) to encrypt communications between the card and the terminal [3] by running an authenticated key establishment before exchanging sensitive data. Obviously, a naıve solution to employ the standard Diffie-Hellman (DH) would not solve the tracking problem described above since the card's permanent identity, its public key, involved in the handshake allows both eavesdroppers and active attackers to identify the same card through different sessions. To mitigate that, EMVCo developed a blinded version of the DH protocol, BDH [2], where in each session the card's public key is blinded with a fresh scalar, making eavesdroppers locked out from subsequent communication. However, even in the presence of



CCS '23, November 26–30, 2023, Copenhagen, Denmark

ACM ISBN 979-8-4007-0050-7/23/11.
https://doi.org/10.1145/3576915.3623109

encryption, the problem of active attackers persists: the card still sends its unblinded signed public key, a strong form of identity, to the terminal allowing the attacker to trace the card.

The recently proposed UBDH [26] protocol, an *unlinkable* version of the BDH protocol, where an attacker cannot link key establishment sessions with the same card, is an example of an authenticated key establishment protocol that satisfies both the initial EMV privacy goals [2], and rules out active attackers. 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. The following table summarises the privacy level each key establishment mechanism achieves.

	passive privacy	active privacy
DH	✗	✗
BDH	✓	✗
UBDH	✓	✓

According to the proposal of EMVCo [2, 3], an EMV 2nd Gen transaction would consist of a key establishment phase followed by a data exchange. Hence we need to consider 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 that the payment system uses to route payments through the network. The table below presents the degrees of privacy obtained by combining a key establishment with the default EMV data exchange.

	passive privacy	active privacy
EMV	✗	✗
BDH + EMV	✓	✗
UBDH + EMV	✓	✗
UBDH + ? = UTX	✓	✓

While there is no privacy in cleartext EMV, encrypting EMV by running BDH or UBDH as the first step does not help achieve an unlinkable protocol where an active attacker cannot link payment sessions, thereby tracing the cardholder. The fact that EMVCo officially abandoned efforts on EMV 2nd Gen to enhance privacy in 2019 [5] also emphasises the need for a newly designed protocol (called UTX in the table) to meet future privacy demands.

To the best of our knowledge, no existing solutions satisfy the basic functional and security requirements of EMV while relying exclusively on the computational resources of a smart card and being unlinkable at the same time. Mobile wallet apps like Apple Pay [28] protect the card number from being revealed by replacing it with a permanent Device Account Number (DAN) stored in the device (e.g. the smartphone). The DAN is exposed to an active attacker in the same way the PAN is exposed in a traditional EMV transaction[1]. At the same time, anonymous credential (AC) schemes [18, 34] are a popular way for establishing unlinkability, e.g. in the context of anonymous access to online services. Some AC schemes have been effectively implemented on smart cards [12, 35]. In principle, an AC scheme could be employed to prove the legitimacy of the card to the terminal without revealing any identifying information. However, the full functionality of an EMV-like transaction requires a much richer functionality. For example, the parties need to agree on the parameters of the transaction, the terminal may need to verify the user PIN, and the bank needs to check that the payment request comes from a valid interaction with the corresponding card. AC schemes can be augmented with attributes that can be used to encode a richer functionality (e.g. attesting that the card is still valid at a certain date). However, such extensions typically rely on zero-knowledge proofs, that we aim to avoid since they would introduce too much overhead for a payment smart card. Furthermore, the design question remains, i.e. how to adapt an AC scheme for use in a larger payment system. In this paper, we demonstrate that a protocol with the desired functional, security and privacy requirements can be designed based on a particular and simple instance of anonymous credentials, namely self-blindable certificates [34]. We discuss some deployment questions at the end of the paper and argue that our protocol could be implemented with minimal overhead on current smart cards.

The main contributions of the paper are as follows.

- *A non-trivial threat model.* We build on recent work [26] that explains why active attackers pose a real threat for contactless payments and how an appropriate Dolev-Yao model [20] fully accounts for them. A key novelty of our model is that we account for both honest and dishonest terminals, but in very different ways. Attackers impersonating terminals not requiring the PIN are implicitly accounted for in the Dolev-Yao model. In contrast, honest terminals requiring the PIN are explicitly represented as processes.
- *Requirements for privacy-preserving card payments.* From EMV we extract functional and security requirements. For privacy requirements we extract from the EMV 2nd Gen draft [5] an unlinkability requirement and clarify it with respect to our threat model.
- *A new payment protocol.* We design a non-trivial protocol that we argue is feasible to implement since it uses standard components that respect limited computational resources of the card. The assemblage, however, is unique. We also explain that new demands imposed by the protocol on infrastructure may be handled by software updates for the existing EMV infrastructure.
- *A proof that the protocol satisfies our requirements.* Notably, unlinkability is proven directly using state-of-the-art bisimulation techniques and does not make use of tools since our particular combination of protocol and threat model is not yet in scope of current tools.

We begin by presenting a design space where we determine the requirements of an unlinkable payment protocol in Section 2, and draw attention to trade-offs between functional and privacy requirements. We then present an unlinkable payment protocol UTX in Section 3, and provide formal analysis in Section 4.

2 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 in

[1]However, an additional layer of security is provided in this case since the device should be ready for communication, e.g. unblocked with the proper app running, etc.

later sections to guide the design of our proposed protocol. We explain the architecture of a payment system that should be respected, and emphasise the functional, security and privacy requirements.

2.1 EMV infrastructure

We present an overview of the payment infrastructure, assumed by the current EMV standard, in Figure 1. The card C is manufactured by the *issuing bank BC* in collaboration with the payment system *PaySys* (e.g. Visa or Amex). The terminal T is connected to an *acquiring bank BT* supporting *PaySys* that processes payments on behalf of the terminal. The acquiring bank processes payments by connecting to the *PaySys* network that exchanges messages between banks.

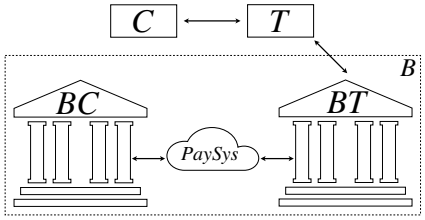


Figure 1: Payment architecture.

A successful run of the protocol results in the generation of an *Application Cryptogram* AC by C. AC is eventually sent by T to BT, either before of 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.

In this paper, we are concerned about hiding the information about the card from the terminal. Thus, when modelling the system, we merge BT, *PaySys*, and BC into a single agent B, modelling their common interface with the terminal when processing payments, as indicated in Figure 1. This is consistent with the fact, that EMV currently does not enforce any exact processing method on the bank's side, i.e. the standard contains an example while "issuers may decide to adopt other methods" [1, Book 2, Section 8].

2.2 Requirements for unlinkable payments

An unlinkable protocol should satisfy three types of requirements: functional, security and privacy requirements. We extract functional and security requirements from the current EMV specification, strengthen some security requirements, and introduce privacy requirements not previously present in EMV.

2.2.1 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 Elliptic-Curve Cryptography (ECC), as already required for the new iteration of the EMV standard [2]. Since, currently, the card must be present within the reader's field for at most 500ms [6], 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 are beyond the scope of this work.

Cards can optionally support offline transactions which carry two risks resulting in the terminal not being paid (when 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.

2.2.2 Security requirements. Recall that some configurations of EMV have been shown to be insecure. 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 the bank accepts the transaction, then T, C, and the bank 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 [2], where a session-specific secret channel was introduced to protect all messages between the card and the terminal from eavesdroppers. Currently, the communication between the card and the terminal is in cleartext, and the AC, that contains transaction details, is always exposed. Formal security definitions reflecting these requirements are introduced in Section 4.4 where we present the analysis of our proposal for a protocol.

2.2.3 Privacy requirements. As mentioned in the introduction and expanded upon next, currently no privacy properties are preserved by EMV. The privacy property we aim for in this paper is *unlinkability*. Unlinkability is standardised in the Common Criteria for Information Technology Security Evaluation ISO 15408 [4], as ensuring that two uses of a card cannot be linked. ISO standard 15408 also covers anonymity. Unlinkability is stronger in the sense that, if two sessions are not anonymous then they can be linked, but the converse does not hold. This explains why unlinkability is a suitable benchmark for privacy.

For this initial discussion, we give an intuitive scheme for defining unlinkability. A formal definition is presented in Section 4.3, where we prove that the protocol we introduce in later sections satisfies unlinkability.

Scheme 1. (unlinkability) *Transactions are 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.*

Let us reflect on the above scheme. The former system represents a real-world scenario where the card is issued and within its lifespan can participate in several protocol sessions. The latter system is an idealised situation, where cards are disposed of after each transaction and can participate in one payment session at most, hence sessions are trivially unlinkable. Whenever, with respect to all attack strategies, there is no distinction between the two scenarios for a given payment protocol, such a protocol is unlinkable. 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 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 Sections 2.2.1, 2.2.2. 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. 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.

3 THE UTX PROTOCOL

In this section, we introduce the UTX (Unlinkable Transactions) protocol that satisfies the security and privacy requirements introduced in Section 2.2. 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 of the protocol. Finally, we thoroughly explain transactions that can either be offline, online, high, or low-value.

3.1 Application selection

The card can generally support several payment methods, or, in EMV lingo, *applications*. In Fig. 2 we schematically show how the terminal currently selects the application. 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.

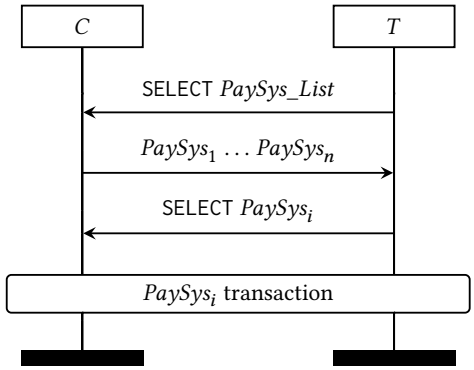


Figure 2: Payment System Selection.

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 their own unlinkable application, and to tolerate that the payment system becomes part of the coarse identity of the card. Our analysis covers both choices.

3.2 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 $\chi_{\mathtt{MM}}$ 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.

3.3 Message theory

We now introduce cryptographic primitives employed by the UTX protocol. Since later in Section 4 we reason about UTX symbolically and assume perfect cryptography, low-level details such as ECC domain parameters are out of scope. In particular, we assume the use of an encryption scheme that guarantees message integrity. Fig. 3 presents the message theory that consists of the syntax, that defines messages agents can form, and the equational theory E, that axiomatises cryptographic operations.

The message theory admits operations for ECC, i.e. multiplication between two field elements (scalars), and multiplication between a scalar and an element of the DH group. Whenever 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 Verheul signature scheme [34], which is invariant under blinding of the message-signature pair (hence can appear as "new" in each session). This scheme supports ECC and has been demonstrated to work sufficiently fast on smart cards [12]. 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 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 $\mathsf{vcheck}(\cdot,\cdot)$ is applied to the signature, blinded with some scalar and the matching Verheul public key, it returns the message, blinded with the same scalar.

$$
\begin{array}{rll}
M, N ::= & \mathfrak{g} & \text{DH group generator (constant)} \\
| & x & \text{variable} \\
| & M \cdot N & \text{multiplication} \\
| & \phi(M, N) & \text{scalar multiplication} \\
| & \mathsf{h}(M) & \text{hash} \\
| & \{M\}_N & \text{symmetric encryption} \\
| & \langle M_1, \ldots, M_k \rangle & n\text{-tuple} \\
| & \mathsf{pk}(M) & \text{public key} \\
| & \mathtt{sig}(M, N) & \text{signature} \\
| & \mathsf{vpk}(M) & \text{Verheul public key} \\
| & \mathtt{vsig}(M, N) & \text{Verheul signature} \\
| & \mathtt{check}(M, N) & \text{check signature} \\
| & \mathtt{vcheck}(M, N) & \text{check Verheul signature} \\
| & \mathsf{p_i}(N) & i\text{th projection} \\
| & \mathtt{dec}(N, M) & \text{symmetric decryption} \\
| & \bot, \mathsf{ok}, \mathtt{accept}, \mathtt{auth}, \mathtt{lo}, \mathtt{hi} & \text{constants}
\end{array}
$$

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

Figure 3: UTX message theory.

3.4 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.

3.4.1 Issuing a card. Here we outline how a card could be manufactured involving a signing authority that payment systems could share as explained in Section 3.2.

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 $\mathsf{pk}(s)$, 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.

3.4.2 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 $\mathtt{MM}$, the authority knowing s issues a certificate of the form $\langle \langle \mathtt{MM}, \phi(b_t, \mathfrak{g}) \rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g}) \rangle, s) \rangle$ 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 3.2, 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.

First, 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 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.

Second, 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 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 cryptogram to gather sensitive information including the PAN.

It is efficient to transmit the month and the bank's public key in a single message, however, in principle, the signatures on each could be separate. In this case, to prevent offline guessing attacks, the payment system should introduce certain padding to small and publicly known constants MM representing months. If a bank requires multiple keys, the payment system could produce multiple certificates.

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.

3.5 The UTX transaction

We introduce online and offline modes of the UTX protocol in Fig. 4. The PIN is asked for in high-value purchases. 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 off and on indicating these two modes of operation. In Fig. 4 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).

3.5.1 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 Section 3.1 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 $k_c := \mathsf{h}(\phi(a \cdot c, \phi(t, \mathfrak{g}))) =_E \mathsf{h}(\phi(t, \phi(a, \phi(c, \mathfrak{g})))) =: k_t$ which they use *to encrypt all further communications.* In Fig. 4, phases of the protocol that are encrypted are represented by a box with a label in the top-left corner indicating the encryption key.

A *passive eavesdropper* who only observes messages is now locked out from the session since it has no access to the derived key. However, an *active attacker* can choose their own public key and engage in the handshake. We will explain below how active attacks are mitigated. The only information about the card exposed at this point is the fact that the card supports the application Unlinkable.

3.5.2 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 $\langle\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t, \mathfrak{g})\rangle, s)\rangle$, the current bank's certificate. 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 month certificate corresponding to MM, the card updates its pointer, leaves it untouched or, aborts the transaction as described in Section 3.2. 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 a the scalar a, and sends to the terminal the following blinded pair $\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathtt{vsig}(\phi(c, \mathfrak{g}), \chi_{\mathtt{MM}}))\rangle$.

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 a session key. This check ensures that the terminal is still communicating with the same card and prevents the construction of fake cards loaded with previously exposed blinded message-signature pairs.

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 MM (since the key $\mathtt{vpk}(\chi_{\mathtt{MM}})$ 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.

3.5.3 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 uPIN to the card together with the transaction details. If this input matches the actual card's PIN, the card includes the ok message both in the reply to the terminal and in the cryptogram to indicate to the issuing bank that the PIN has

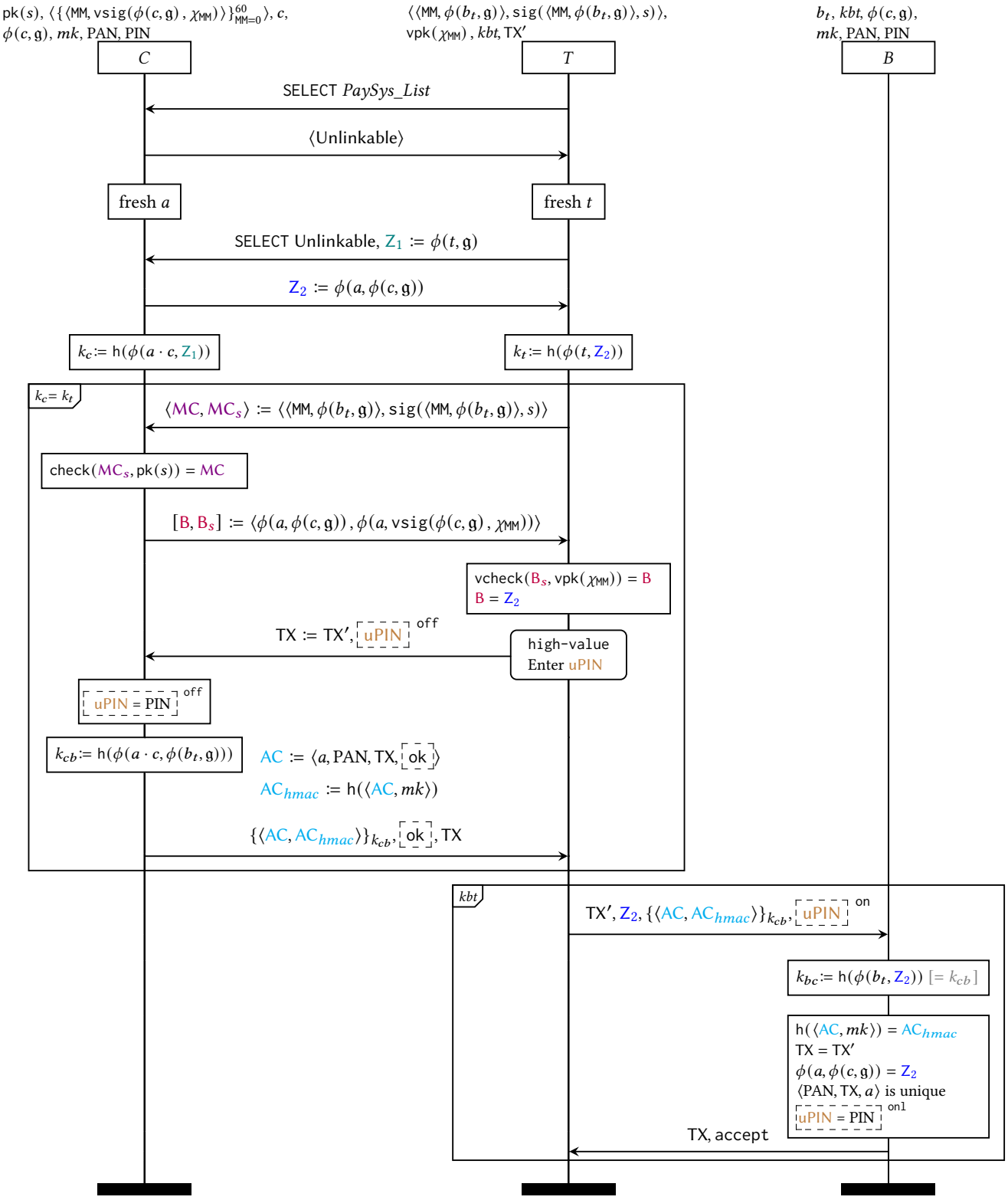


Figure 4: The UTX protocol. Offline and online high-value modes are annotated as `off` and `on` respectively.

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.

3.5.4 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 uPIN (when the transaction is high-value offline). The card computes $k_{cb} := \mathsf{h}(\phi(a \cdot c, \phi(b_t, \mathfrak{g})))$, 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.

- $\mathsf{AC}^{\perp} := \langle a, \mathsf{PAN}, \mathsf{TX}\rangle$ if no uPIN has been received.
- $\mathsf{AC}^{\mathsf{ok}} := \langle a, \mathsf{PAN}, \mathsf{TX}, \mathsf{ok}\rangle$ if the received uPIN is correct.
- $\mathsf{AC}^{\mathsf{no}} := \langle a, \mathsf{PAN}, \mathsf{TX}, \mathsf{no}\rangle$ 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 $\mathsf{h}(\langle \mathsf{AC}, mk\rangle)$ and replies respectively with one the following messages to the terminal.

- $\langle \{\langle \mathsf{AC}^{\perp}, \mathsf{h}(\langle \mathsf{AC}^{\perp}, mk\rangle)\rangle\}_{k_{cb}}, \perp, \mathsf{TX}\rangle$
- $\langle \{\langle \mathsf{AC}^{\mathsf{ok}}, \mathsf{h}\Big(\langle \mathsf{AC}^{\mathsf{ok}}, mk\rangle\Big)\rangle\}_{k_{cb}}, \mathsf{ok}, \mathsf{TX}\rangle$
- $\langle \{\langle \mathsf{AC}^{\mathsf{no}}, \mathsf{h}(\langle \mathsf{AC}^{\mathsf{no}}, mk\rangle)\rangle\}_{k_{cb}}, \mathsf{no}, \mathsf{TX}\rangle$

Each of these messages corresponds to the cryptograms described and contains additional information on whether the PIN was successfully verified by the card (ok entry), or the PIN verification has failed (no entry) because the terminal cannot open the cryptogram encrypted for the acquiring bank.

Notice that the card includes the 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 Section 3.5.2, 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 recent aliveness of the card from the perspective of the bank even in the presence of compromised terminals.

3.5.5 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 uPIN in case the transaction is high-value online, or the message $\perp$ otherwise.

Recall that B in Fig. 4 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 symmetric key with the card $k_{bc} := \mathsf{h}(\phi(b_t, \mathsf{Z}_2)) = \mathsf{h}(\phi(b_t, \phi(a, \phi(c, \mathfrak{g}))))$ and to decrypt the cryptogram. Internally to B, the acquiring bank uses the PAN from the decrypted cryptogram and forwards all the information received from terminal to the issuing bank. In turn, the issuing 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 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 uPIN matches the card's PIN: 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 is required 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 Sections 3.2, 3.4.1, and 3.4.2.

4 UNLINKABILITY AND SECURITY ANALYSIS

We specify and verify our proposed protocol in a variant of the applied π-calculus [9]. In the formulation of the property of transaction unlinkability, we employ *quasi-open bisimilarity* [27] – an equivalence notion that is preserved in all contexts and captures an attacker capable of making dynamic decisions – and its corresponding labelled transition system. For the properties that constitute payment security, we rely on the ProVerif tool and its notion of *correspondence assertions* [13, 14]. 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.

4.1 Attacker model

The attacker model we use for verification of the UTX protocol is a Dolev-Yao attacker [20] who controls the communications between the card, the terminal, and the bank. 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 of 100cm, an attacker can power up the card and interact with it [25], explaining why we insist on this attacker model when verifying our protocol. The connection between the terminal and the bank is not necessarily secure and an attacker could manipulate this connection, e.g. cutting it and forcing the terminal to go offline.

We assume that cardholders only enter their PIN into honest terminals. In other words, the cardholder uses terminals at reputable points of sale in the process of a conscious purchase and never enters their PIN into random terminals that pop up on the street. 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 by relaying the messages between an honest terminal and an honest card [24], 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, if an attacker is physically unable to perform contact transactions, low-value contactless payments are unlinkable even if the PIN is compromised. We analyse this case separately in the extended version of the current paper [16].

There are other attacker models. We could have verified with respect to a weaker distant attacker that operates within a distance of 100cm to 20m from the card and can only eavesdrop on communications [23, 32]. This attacker would have been sufficient to establish privacy for the proposal already considered by EMVCo establishing a channel to encrypt regular EMV transactions [2]. Other attackers may attempt side-channel attacks by measuring execution time of cryptographic operations, or the response time from the bank, which is out of scope of our analysis.

4.2 Formal specification of the protocol

We use the applied π-calculus language [9] to specify the formal model of the UTX protocol where all cards are synchronised to execute within the same month MM. In the essence of this formalism, we have processes that can communicate by sending and receiving messages using channels. We write $\overline{ch}\langle M\rangle$ and $ch(x)$ for sending the message M or receiving the input x on the channel ch, respectively. A process can also generate private values (used e.g. for fresh secret keys and nonces), written as va, be replicated using the ! operator (allowing an unbounded number of its instances to execute), and run in parallel with other processes using the | operator. In Fig. 5 we have three processes that model the execution of a session of our protocol by the three roles in the UTX protocol: the terminal, the card, and the bank. Events, marked with `ev:`, will be used in the security analysis and can be ignored until Section 4.4. Fig. 6 specifies the top-level process that expresses how these processes are assembled and instantiated across multiple payment sessions in a full execution of the protocol.

4.2.1 The card process. C, described in Fig. 5a, represents the execution of a payment session by a card.

$$vch.\overline{card}\langle ch\rangle.C(ch, c, pk_s, \mathtt{vsig}_{\mathtt{MM}}, \mathrm{PAN}, mk, \mathrm{PIN})$$

It is parameterised by the session channel ch, the card's secret key c, the system-wide public key pk_s used to check the bank's certificate 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. First, the card establishes a key with the terminal, then checks the certificate of the terminal and sends back its own month 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 $\mathrm{AC}^{\mathsf{no}}$ is generated and sent as reply to the terminal.

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

$$vch.\overline{term}\langle ch\rangle.T(user, ch, pk_{\mathtt{MM}}, \mathrm{crt}, kbt)$$

T is parametrised by the secret channel $user$ used to enter the PIN, the session channel ch, the public key used for verifying the card certificate for the given month $pk_{\mathtt{MM}}$, and the shared secret key between the terminal and the bank kbt. 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 $T_{\mathtt{offhi}}$, and for low-value transactions $T_{\mathtt{lo}}$.

Initially, each terminal proceeds with the key establishment phase with the card, sends its certificate, and checks the received month certificate. High-value terminals rely on the PIN entered by the cardholder to perform transaction authorisation. To represent the different types of transactions that can occur, we have constants `lo` and `hi` for low-value and high-value transactions respectively.

The online high-value terminal process $T_{\mathtt{onhi}}$ is given in Fig. 5b. Since the transaction is high-value, the PIN is required and after the initialisation, the user enters the PIN using the private channel

(a) **Card** $C(\boldsymbol{ch}, c, pk_s, \mathbf{vsig_{MM}}, \mathbf{PAN}, mk, \mathbf{PIN})$

$ch(z_1).$
$va.\,\mathtt{let}\, z_2 := \phi(a, \phi(c, \mathfrak{g}))\ \mathtt{in}$
$\overline{ch}\langle z_2\rangle.$
$\mathtt{let}\, k_c := \mathsf{h}(\phi(a \cdot c, z_1))\ \mathtt{in}$
$ch(m).$ *
$\mathtt{let}\, \langle\langle \mathsf{MM}, y_B\rangle, \mathsf{MC}_s\rangle := \mathsf{dec}(m, k_c)\ \mathtt{in}$
$\mathtt{if}\ \mathsf{check}(\mathsf{MC}_s, pk_s) = \langle \mathsf{MM}, y_B\rangle\ \mathtt{then}$
$\mathtt{let}\, emc := \{\langle \phi(a, \phi(c, \mathfrak{g})), \phi(a, \mathsf{vsig}_{\mathsf{MM}})\rangle\}_{k_c}\ \mathtt{in}$
$\overline{ch}\langle emc\rangle.$
$ch(x).$
$\mathtt{let}\, \langle \mathsf{TX}, \mathsf{uPin}\rangle := \mathsf{dec}(x, k_c)\ \mathtt{in}$
$\mathtt{let}\, \mathsf{AC}^{\perp} := \langle a, \mathsf{PAN}, \mathsf{TX}\rangle\ \mathtt{in}$
$\mathtt{let}\, \mathsf{AC}^{\mathsf{ok}} := \langle a, \mathsf{PAN}, \mathsf{TX}, \mathsf{ok}\rangle\ \mathtt{in}$
$\mathtt{let}\, \mathsf{AC}^{\mathsf{no}} := \langle a, \mathsf{PAN}, \mathsf{TX}, \mathsf{no}\rangle\ \mathtt{in}$
$\mathtt{let}\, k_{cb} := \mathsf{h}(\phi(a \cdot c, y_B))\ \mathtt{in}$
$\mathtt{if}\ \mathsf{uPin} = \perp\ \mathtt{then}$
  $\mathtt{let}\, \mathsf{HAC} := \langle \mathsf{AC}^{\perp}, \mathsf{h}(\langle \mathsf{AC}^{\perp}, mk\rangle)\rangle\ \mathtt{in}$
  $\mathtt{let}\, eac := \{\langle \{\mathsf{HAC}\}_{k_{cb}}, \perp, \mathsf{TX}\rangle\}_{k_c}\ \mathtt{in}$
  $\mathsf{ev}:\mathsf{CRunB}\,(eac)$
  $\mathsf{ev}:\mathsf{CRun}\,(z_1, z_2, m, emc, x, eac)$
  $\overline{ch}\langle eac\rangle$
$\mathtt{else if}\ \mathsf{uPin} = \mathsf{PIN}\ \mathtt{then}$
  $\mathtt{let}\, \mathsf{HAC} := \langle \mathsf{AC}^{\mathsf{ok}}, \mathsf{h}(\langle \mathsf{AC}^{\mathsf{ok}}, mk\rangle)\rangle\ \mathtt{in}$
  $\mathtt{let}\, eac := \{\langle \{\mathsf{HAC}\}_{k_{cb}}, \mathsf{ok}, \mathsf{TX}\rangle\}_{k_c}\ \mathtt{in}$
  $\mathsf{ev}:\mathsf{CRunB}\,(\{\mathsf{HAC}\}_{k_{cb}})$
  $\mathsf{ev}:\mathsf{CRun}\,(z_1, z_2, m, emc, x, eac)$
  $\overline{ch}\langle eac\rangle$
$\mathtt{else}$
  $\mathtt{let}\, \mathsf{HAC} := \langle \mathsf{AC}^{\mathsf{no}}, \mathsf{h}(\langle \mathsf{AC}^{\mathsf{no}}, mk\rangle)\rangle\ \mathtt{in}$
  $\mathtt{let}\, eac := \{\langle \{\mathsf{HAC}\}_{k_{cb}}, \mathsf{no}, \mathsf{TX}\rangle\}_{k_c}\ \mathtt{in}$
  $\mathsf{ev}:\mathsf{CRunB}\,(\{\mathsf{HAC}\}_{k_{cb}})$
  $\mathsf{ev}:\mathsf{CRun}\,(z_1, z_2, m, emc, x, eac)$
  $\overline{ch}\langle eac\rangle$

(b) **Terminal** $T_{\mathtt{onhi}}(\boldsymbol{user}, \boldsymbol{ch}, pk_{\mathsf{MM}}, \mathbf{crt}, \boldsymbol{kbt})$

$\nu\,\mathsf{TXdata}.$
$\mathtt{let}\, \mathsf{TX} := \langle \mathsf{TXdata}, \mathsf{hi}\rangle\ \mathtt{in}$
$vt.\mathtt{let}\, z_1 := \phi(t, \mathfrak{g})\ \mathtt{in}$
$\overline{ch}\langle z_1\rangle.$
$ch(z_2).$
$\mathtt{let}\, k_t := \mathsf{h}(\phi(t, z_2))\ \mathtt{in}$
$\overline{ch}\langle \{\mathsf{crt}\}_{k_t}\rangle.$
$ch(n).$
$\mathtt{let}\, \langle \mathsf{B}, \mathsf{B}_s\rangle := \mathsf{dec}(n, k_t)\ \mathtt{in}$
$\mathtt{if}\ \mathsf{vcheck}(\mathsf{B}_s, pk_{\mathsf{MM}}) = \mathsf{B}\ \mathtt{then}$
$\mathtt{if}\ \mathsf{B} = z_2\ \mathtt{then}$
$user(\mathsf{uPIN}).$
$\mathtt{let}\, etx = \{\langle \mathsf{TX}, \perp\rangle\}_{k_t}\ \mathtt{in}$
$\overline{ch}\langle etx\rangle.$
$ch(y).$
$\mathtt{let}\, \langle \mathsf{EHAC}, \mathsf{pinV}, \mathsf{tx}\rangle := \mathsf{dec}(y, k_t)\ \mathtt{in}$
$\mathtt{if}\ \mathsf{tx} = \mathsf{TX}\ \mathtt{then}$
$\mathsf{ev}:\mathsf{TComC}\,(z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, etx, y)$
$\mathtt{let}\, req = \{\langle \mathsf{TX}, z_2, \mathsf{EHAC}, \mathsf{uPIN}\rangle\}_{kbt}\ \mathtt{in}$
$\mathsf{ev}:\mathsf{TRunBC}\,(req, z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, etx, y)$
$\overline{ch}\langle req\rangle.$
$ch(r).$
$\mathtt{if}\ \mathsf{dec}(r, kbt) = \langle \mathsf{TX}, \mathsf{rtype}\rangle\ \mathtt{then}$
  $\mathsf{ev}:\mathsf{TComBC}\,(req, r, z_1, z_2, \{\mathsf{crt}\}_{k_t}, n, etx, y)$
  $\mathtt{if}\ \mathsf{rtype} = \mathsf{accept}\ \mathtt{then}$
    $\mathsf{ev}:\mathsf{TAccept}\,(kbt, \mathsf{TX})$
    $\overline{ch}\langle \mathsf{auth}\rangle$

(c) **Bank** $B(\boldsymbol{ch}, si, \boldsymbol{kbt}, b_t)$

$ch(x).$
$\mathtt{let}\, \langle \mathsf{TX}', z_2, \mathsf{EAC}, \mathsf{uPIN}\rangle := \mathsf{dec}(x, kbt)$
$\mathtt{let}\, k_{bc} := \mathsf{h}(\phi(b_t, z_2))\ \mathtt{in}$
$\mathtt{let}\, \langle \mathsf{AC}, \mathsf{AC}_{hmac}\rangle := \mathsf{dec}(\mathsf{EAC}, k_{bc})\ \mathtt{in}$
$\mathtt{let}\, \langle x_a, \mathsf{PAN}, \mathsf{TX}, \mathsf{pinV}\rangle = \mathsf{AC}\ \mathtt{in}$
$\langle si, \mathsf{PAN}\rangle(\mathsf{PIN}, mk, pk_c).$
$\mathtt{if}\ \mathsf{h}(\langle \mathsf{AC}, mk\rangle) = \mathsf{AC}_{hmac}\ \mathtt{then}$
$\mathtt{if}\ \mathsf{TX} = \mathsf{TX'}\ \mathtt{then}$
$\mathtt{if}\ \phi(x_a, pk_c) = z_2$
$\mathtt{let}\, \langle \mathsf{TXdata}, \mathsf{TXtype}\rangle := \mathsf{TX'}\ \mathtt{in}$
$\mathtt{if}\ \mathsf{TXtype} = \mathsf{lo}\ \mathtt{then}$
  $\mathsf{ev}:\mathsf{BComC}\,(\mathsf{EAC})$
  $\mathsf{ev}:\mathsf{BRunT}\,(x, \{\langle \mathsf{TX'}, \mathsf{accept}\rangle\}_{kbt})$
  $\mathsf{ev}:\mathsf{BComTC}\,(x)$
  $\overline{ch}\langle \{\langle \mathsf{TX'}, \mathsf{accept}\rangle\}_{kbt}\rangle$
$\mathtt{else if}\ \mathsf{TXtype} = \mathsf{hi}\ \mathtt{then}$
  $\mathtt{if}\ (\mathsf{pinV} = \mathsf{ok}) \vee (\mathsf{uPIN} = \mathsf{PIN})\ \mathtt{then}$
    $\mathsf{ev}:\mathsf{BComC}\,(\mathsf{EAC})$
    $\mathsf{ev}:\mathsf{BRunT}\,(x, \{\langle \mathsf{TX}', \mathsf{accept}\rangle\}_{kbt})$
    $\mathsf{ev}:\mathsf{BComTC}\,(x)$
    $\overline{ch}\langle \{\langle \mathsf{TX}', \mathsf{accept}\rangle\}_{kbt}\rangle$
  $\mathtt{else}$
    $\mathsf{ev}:\mathsf{BReject}\,(kbt, \mathsf{TX'})$
    $\mathsf{ev}:\mathsf{BComC}\,(\mathsf{EAC})$
    $\mathsf{ev}:\mathsf{BRunT}\,(x, \{\langle \mathsf{TX'}, \mathtt{reject}\rangle\}_{kbt})$
    $\mathsf{ev}:\mathsf{BComTC}\,(x)$
    $\overline{ch}\langle \{\langle \mathsf{TX'}, \mathtt{reject}\rangle\}_{kbt}\rangle.$

In addition, there are processes $T_{\mathtt{offhi}}$ and $T_{\mathtt{lo}}$ defining the behavior for offline high-value and low-value transactions, respectively (presented in the proof of Theorem 1 [16]). Moreover there is T defined as $T_{\mathtt{onhi}} + T_{\mathtt{offhi}} + T_{\mathtt{lo}}$. The star * indicates at which point the card selects the appropriate month certificate to present (see Appendix D of [16] for a larger specification making this choice explicit).

Figure 5: Specifications for the three roles in the UTX protocol.

(a) The real protocol specification $UTX_{\mathbf{impl}}$

$$
\begin{array}{l}
\nu\, user, s, si, \chi_{\mathtt{MM}}.\overline{out}\langle \mathsf{pk}(s)\rangle.\overline{out}\langle \mathsf{vpk}(\chi_{\mathtt{MM}})\rangle.\Big(\\
\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
\quad\quad \mathtt{let}\ \mathrm{crtC} \coloneqq \mathtt{vsig}(\phi(c,\mathfrak{g})\,, \chi_{\mathtt{MM}})\ \mathtt{in} \\
\quad\quad\quad !\nu ch.\overline{card}\langle ch\rangle.C(ch, c, \mathsf{pk}(s)\,, \mathrm{crtC}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
\quad\quad\quad \mid !\overline{user}\langle \mathrm{PIN}\rangle \mid !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big)\ \mid \\
\quad \nu b_t.!\nu kbt.\big(\\
\quad\quad \nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \mid \\
\quad\quad \mathtt{let}\ \mathrm{crt} \coloneqq \langle\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in} \\
\quad\quad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathsf{vpk}(\chi_{\mathtt{MM}})\,, \mathrm{crt}, kbt)\big)\Big)
\end{array}
$$

(b) The ideal unlinkable protocol specification $UTX_{\mathbf{spec}}$

$$
\begin{array}{l}
\nu\, user, s, si, \chi_{\mathtt{MM}}.\overline{out}\langle \mathsf{pk}(s)\rangle.\overline{out}\langle \mathsf{vpk}(\chi_{\mathtt{MM}})\rangle.\Big(\\
\quad !\nu \mathrm{PIN}, mk, c, \mathrm{PAN}.\big(\\
\quad\quad \mathtt{let}\ \mathrm{crtC} \coloneqq \mathtt{vsig}(\phi(c,\mathfrak{g})\,, \chi_{\mathtt{MM}})\ \mathtt{in} \\
\quad\quad\quad \nu ch.\overline{card}\langle ch\rangle.C(ch, c, \mathsf{pk}(s)\,, \mathrm{crtC}, \mathrm{PAN}, mk, \mathrm{PIN}) \\
\quad\quad\quad \mid !\overline{user}\langle \mathrm{PIN}\rangle \mid !\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle\ \big)\ \mid \\
\quad \nu b_t.!\nu kbt.\big(\\
\quad\quad \nu ch.\overline{bank}\langle ch\rangle.B(ch, si, kbt, b_t)\ \mid \\
\quad\quad \mathtt{let}\ \mathrm{crt} \coloneqq \langle\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle \mathtt{in} \\
\quad\quad \nu ch.\overline{term}\langle ch\rangle.T(user, ch, \mathsf{vpk}(\chi_{\mathtt{MM}})\,, \mathrm{crt}, kbt)\big)\Big)
\end{array}
$$

Figure 6: Specifications for the real UTX protocol and its ideal unlinkable version

user, which models that the PIN can only be entered into honest terminals. Then the terminal sends the transaction details to the card, receives the application cryptogram in the response, and sends it to the bank together with the entered PIN. Since we are in the online mode, the terminal authorises the transaction only after receiving confirmation from the bank. In contrast, offline terminals authorise transactions right after receiving the reply from the card.

The offline high-value and low-value modes are similar, and their specifications appear in Appendix B of [16]. The offline high-value mode requires the terminal to send the entered PIN to the card since only the card can verify the PIN if the terminal is offline. Terminals operating in this mode accept transactions only if the ok reply has been received from the card, however, regardless of the outcome, the cryptogram is always sent to the bank eventually. Low-value transactions are PINless, hence the corresponding role specification $T_{\mathtt{lo}}$ does not require that online and offline modes are distinguished.

4.2.3 The bank process. B, specified in Fig. 5c, 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 $!\overline{\langle si, \mathrm{PAN}\rangle}\langle\langle \mathrm{PIN}, mk, \phi(c,\mathfrak{g})\rangle\rangle$, and the corresponding entry can be read by receiving a message on the channel consisting of the pair $\langle si, \mathrm{PAN}\rangle$ 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 cryptogram, and obtains 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.

4.2.4 The full protocol. To complete the specification, in Fig. 6 we present the full system, which operates as follows. At the start, the system-wide parameters are generated and public data that includes the system public key $\mathsf{pk}(s)$ and the month public key $\mathsf{vpk}(\chi_{\mathtt{MM}})$ 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 the 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 a session-wise (hence the replication) symmetric key between the bank and the terminal kbt. Notice also that we are using public session channels ch to give an attacker the power to observe which agents are communicating.

4.2.5 The Dolev-Yao model accounts for malicious terminals. Terminals operated by attackers should be accounted for in our threat model, since, consistent with EMV, terminals are not authenticated by the card and hence can be implemented and operated by anyone. In our model, indeed, an attacker can impersonate a terminal, either up until the point when the PIN is requested, or, in modes where the PIN is never requested, proceed to obtain the encrypted application cryptogram produced by the card. To operate as a terminal, an attacker only needs the bank's certificate $\langle\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, \mathtt{sig}(\langle \mathtt{MM}, \phi(b_t,\mathfrak{g})\rangle, s)\rangle$ which is straightforward to obtain since an honest terminal gives away this certificate to anyone it communicates with. Indeed, a fake card can be used to obtain new monthly certificates even if authorities only distribute them to honest terminals. Such a fake card would first engage in a Diffie-Hellman handshake with an honest terminal, which establishes a channel on which an attacker can receive the certificate currently loaded into the terminal. No knowledge of any private key is required to implement such fake cards. This viable threat is accounted for in the proofs of unlinkability theorems in the next section.

4.3 Unlinkability definition and analysis

In this section, we clarify the informal definition of unlinkability given by Scheme 1 presented in Section 2.2.3 and formally prove that UTX is unlinkable. We also present some variations on the unlinkability problem that show that unlinkability still holds even if certain marginal coarse identities are tolerated.

4.3.1 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. 6. 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 system 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 now to give the unlinkability definition.

Definition 1. (unlinkability) *We say that the payments are unlinkable if $UTX_{impl} \sim UTX_{spec}$, where $\sim$ is quasi-open bisimilarity.*

There is a difference with the definition of unlinkability for key establishment considered in [26], where the terminal and the bank are deliberately omitted. The reason is that the key establishment in isolation, i.e. the UTX protocol up to the *Cardholder verification* phase, 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.

Finally we are ready to formulate our first result.

Theorem 1. $UTX_{impl} \sim UTX_{spec}$.

The detailed proof of Theorem 1 is given in [16], however, we give a proof sketch here. The key is to give a relation $\mathfrak{R}$ between processes representing states of the two worlds demonstrating that an attacker has no strategy allowing to distinguish between these two worlds. We form such a relation by pairing the appropriate states and checking that it satisfies the conditions for a quasi-open bisimulation. We pair the states based on the number of sessions *started* with terminals, cards, and banks and the respective stages of each session; and we ignore the number of exhausted processes that model entering the PIN and accessing the database for card's details. Then we check that each possible transition that either world can make can be matched by the opposing world; that the resulting states are related by $\mathfrak{R}$, that any two related states are statically equivalent, i.e. indistinguishable by an attacker who can only observe which messages are on the network in this state; and finally, that $\mathfrak{R}$ is *open*, i.e. there is no way for an attacker to distinguish between two worlds by manipulating free variables.

4.3.2 Unlinkability in the face of coarse identities. Below we justify the observation made in Section 2.2.3 where we pointed out that unlinkability can only be achieved up to the fingerprint comprising the coarse identities of the card being revealed. We explain below how such coarse identities of the card can exist in the system without compromising unlinkability.

Signing authority. We demonstrate that UTX is unlinkable even if an attacker can distinguish two cards that use different signing authorities. To do so, we exploit the fact that quasi-open bisimilarity is a congruence [27], i.e. when a *smaller system* satisfies unlinkability, then a *larger system* containing the smaller one as a subsystem also satisfies unlinkability, i.e. process equivalence is preserved in any context. A context is a process "with a hole" such as $O(\cdot) := !(\cdot)$. Notice, by putting UTX_{impl} into O we obtain a system with multiple signing authorities. Similarly, by putting UTX_{spec} into O results in an ideal world in each card engages still in one session, but may use different signing authorities. Now, since quasi-open bisimilarity is a congruence, the following holds.

Corollary 1. $!UTX_{impl} \sim !UTX_{spec}$, *i.e. UTX is unlinkable even in the presence of multiple signing authorities.*

The above means that unlinkability holds for systems with multiple signing authorities as long as we tolerate that coarse identity. That is, we permit a coarse identity, a signing authority, to exist in the system, as represented by building multiple authorities into the ideal world $!UTX_{\text{spec}}$, without compromising unlinkability. In particular, Corollary 1 concerns the degree of unlinkability that can be established in a deployment scenario where multiple payment systems might not agree to provide a common application for unlinkable payments as discussed in Section 3.1, and therefore these different payment systems form a coarse identity of the card.

The card has been used recently. To clarify that the existence of cards valid for several months does not invalidate unlinkability, we consider a model of unlinkability, where cards can respond to two months at any moment. Furthermore, this model admits transitions from one month to the next, maintaining a pointer as described in Section 3.2. To reflect such behaviour of cards, we build into the definition of a process modelling a card the ability to respond to two months at any time and, whenever the new month is asked, to invalidate the oldest of the two months. Notice that this requires a card to carry the state, i.e. to remember that it should respond only to the most recent two months and never respond to older months if asked. In [16] we show how to employ recursion to model such behaviour and prove the following.

Theorem 2. $UTXMM_{impl} \sim UTXMM_{spec}$.

In the above $UTXMM_{\text{impl}}$ and $UTXMM_{\text{spec}}$ define the real and the ideal worlds in an enhanced model. The ideal world models an infinite supply of cards that are used only once, and in that single session, may either respond to the two most recent months, or the three most recent months (the latter modelling the tolerance of cards that are one month behind and can still be updated to the current month). Therefore, a coarse identity of whether or not the card has already been used in a session with an up-to-date terminal in the current month can also exist in the system without compromising unlinkability. There is an additional assumption made in this model, specifically, that we do not verify the unlinkability of cards which have not been used at all in the previous month. The coarse identity of having used the card in the past month, but not in the current month, barely gives any identifying information away at all. However, a card that has not been used for over two month,

is relatively easy to identify among a pool of cards that are used in a normal, more frequent, manner, since it may be tracked with high probability by observing whether it responds rather than blocks when presented with a two-month-old certificate.

4.3.3 Tools cannot yet verify the unlinkability of UTX. In the proofs of Theorems 1 and 2 establishing the unlinkability of UTX, we have constructed and checked by hand a bisimulation between the real and the ideal worlds of the respective models of the protocol. Below, we briefly address the question of whether current tools can be used to confidently reach the same conclusion.

Widely-used tools such as Tamarin [10] and ProVerif [15] offer limited support for bisimilarity checking – they can verify diff-equivalence, i.e., equivalence checking between two processes that differ only in the messages exchanged. Definition 1 does not fall into the diff-equivalence category since the UTX_{impl} and UTX_{spec} processes have different structures. Hence, the use of Tamarin is ruled out. ProVerif, however, makes an attempt to represent the equivalence problem for arbitrary processes as a diff-equivalence problem, thus it can be considered as a candidate for verifying the unlinkability of UTX. Moreover, recently, a new version of ProVerif [19] that may verify observational equivalence (aka early bisimilarity) between two arbitrary processes, lifting restrictions on diff-equivalence, has been introduced. Our experiments show that neither ProVerif nor the recent update can yet verify the unlinkability of the single-month model of UTX as given in Definition 1. This means that, for the time being, our manual proofs are justified. We provide evidence in the repository [17], which contains ProVerif models that fail to terminate. Hence, the investigation of whether it is possible to use current tools to verify the unlinkability of UTX, and to transform Definition 1 into a diff-equivalence problem in a sound and complete manner is left as future work.

4.4 Authentication in UTX

Our security definition supporting the requirements identified in Section 2.2.2 relies on an authentication property called *injective agreement* [29]. A party X injectively agrees with the parties Y and Z whenever if X thinks it has authenticated Y and Z, then Y and Z executed the protocol exchanging the same messages as X (*agreement*), and each run of X corresponds to a unique run of Y and Z (*injectivity*).

To verify injective agreement in UTX we have already included events in role specifications in Fig. 5 marking certain stages reached by processes during the execution of the protocol and then evaluate correspondence assertions [14] between events listed in Fig. 7 using the ProVerif tool [13]. The extended paper [16] contains further details regarding using ProVerif.

The events in Fig. 7 are parametrised by the messages the card, the terminal and the bank exchange, i.e. z_1 and z_2 stand for the ephemeral terminal's key and the blinded card's public key, *ec*, *emc*, *etx*, *eac* represent the messages the card exchanges with the terminal and *req*, *resp* the message the terminal exchanges with the bank. Finally, EAC represents the encrypted cryptogram $\{\langle \mathsf{AC}, \mathsf{AC}_{hmac}\rangle\}_{k_{cb}}$.

The first three assertions in Fig. 7 are straightforward – whenever the terminal or the bank thinks it has executed the session with the rest of the agents, they have exchanged the same messages, thereby agreeing on crucial data such as the derived keys, transaction details,

The terminal agrees with the card (before contacting bank)
$\mathtt{TComC}(z_1, z_2, ec, emc, etx, eac) \Rightarrow$
$\mathtt{CRun}(z_1, z_2, ec, emc, etx, eac)$

The terminal agrees wih the bank and the card
$\mathtt{TComBC}(req, resp, z_1, z_2, ec, emc, etx, eac) \Rightarrow$
$\mathtt{BRunT}(req, resp) \ \wedge \ \mathtt{CRun}(z_1, z_2, ec, emc, etx, eac)$

The bank agrees with the terminal and the card
$\mathtt{BComTC}(req) \Rightarrow$
$\mathtt{TRunBC}(req, z_1, z_2, ec, emc, etx, eac) \wedge$
$\mathtt{CRun}(z_1, z_2, ec, emc, etx, eac))$

Bank agrees with the card on the encrypted cryptogram
$\mathtt{BComC}(\mathsf{EAC}) \Rightarrow \mathtt{CRunB}(\mathsf{EAC})$

Figure 7: Injective agreement correspondences in UTX.

the cryptogram, etc. The last assertion, representing the agreement between the bank and the card, ensures that an honest card was involved in low-value contactless payment even if terminals are fully compromised. In this scenario the terminals can be omitted in the specification as explained in the related work [36].

Security under compromised terminals. Using ProVerif, we support the point made at the end of Section 3.5.4 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 $\chi_{\mathtt{MM}}$. Another scenario in which the terminal accepts a potentially fake card is when 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.

The repository [17] contains the code specifying the injective agreement in the UTX protocol and the expected secrecy of the private data. All properties are successfully verified within 100 minutes. The code verifying additional scenarios described above is provided in the directory *compromised*.

4.4.1 Remark on replay protection. We explain here a small difference between the model used to verify unlinkability and authentication concerning replay protection. Recall that replay protection is enforced by the uniqueness by the bank of the triple $\langle \mathsf{PAN}, \mathsf{TX}, a\rangle$, as specified in Section 3.5.4 and in Fig. 4. This check is an essential ingredient for authentication, without which authentication could not be verified. Hence replay protection is accounted for in the threat model used in Section 4.4, specifically in line 165 of the ProVerif model.

In contrast, the threat model we use for unlinkability simplifies this aspect by allowing terminals to replay cryptograms, that is the bank skips the uniqueness check. This does not introduce problems for the following two reasons. Firstly, the bank in the UTX protocol

sends no message intended for the card, hence there is no way for the terminal to probe cards with such a message in an attempt to track them. Secondly, an observable auth output by the terminal that reveals whether the cryptogram was accepted by the bank introduces no issues regardless of the presence of the check. With replay protection, any attempt to replay the message from the terminal to the bank, i.e. the cryptogram with auxiliary data would result in the absence of auth, while, without replay protection, the replay would result in the message auth being always present. In both cases it is impossible for an attacker to link the presence or the absence of the auth with any session other than the one in which the cryptogram was created, and hence it cannot be used to link two sessions with the same card.

4.5 Estimation 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 [22, 30] for the Multos Card ML3 supporting ECC scalar multiplication. The table below summarises the amount of time for individual operations performed by the card in the UTX protocol. 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 [6].

Operation	$\phi(\cdot,\cdot)$	$\mathsf{h}(\cdot)$	$\mathsf{dec}(\cdot,\cdot)$	$\{\cdot\}.$	$\mathsf{check}(\cdot,\cdot)$
# 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.

5 CONCLUSION

In this paper, we have identified in Section 2 the requirements for a smartcard-based payments protocol, and have demonstrated that at least one protocol satisfying these requirements exists – the UTX protocol presented in Fig. 4. We strengthen the initial security of EMV as explained in Section 2.2. In particular, we request 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 as we have explained in Sections 3.4.2, 3.5.4. Fig. 7 summarises how we have proved in ProVerif that UTX satisfies all security requirements we have identified.

We explain how ISO 15408 supports targeting unlinkability as our privacy requirement in Section 2.2.3, and highlight that the fingerprint of the card, comprising coarse identities of a card that permit groups of cards to be tracked, should be minimised. Since strong identities compromise unlinkability, we have hidden any strong identity of the card by utilising Verheul signatures to make the validity signature distinct in every session, as explained in Section 3.5.2, and by encrypting the cryptogram that contains the PAN to hide it from the terminal, as explained in Section 3.5.4. 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, as explained in Sections 3.2, 3.5.2. If payment systems agree on a common certification authority we may reduce the card's fingerprint further by introducing the Unlinkable application as explained in Section 3.1. Theorem 1 proves that these measures indeed achieve unlinkability in UTX.

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 contrasts to the current EMV standard, which has many different modes of operation, defined in 2000 pages split into several books; thus the variety of *implementations* 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 and the applied π-calculus specification of UTX in Fig. 5 go some way towards this aim.

Roll-out of the UTX protocol is feasible. The software of banks and terminals can be updated in advance across a region so both accept unlinkable payments, while continuing the support of old payment methods. Then cards supporting unlinkable payments can be issued. Of course, new cards must implement only one application to avoid attacks that downgrade cards to EMV.

Regarding the protocol design future work includes introducing relay protection [33]. Firstly, it should mitigate the situation where a high-value online transaction is compromised via a relay attack if the PIN is exposed, as we mention in Section 4.1. Secondly, it is essential for the protocol that supports PIN tries counter, which limits the number of incorrect attempts to enter the PIN. An active attacker can exploit PIN tries counter by relaying messages from the honest terminal waiting to process an online high-value transaction to the card and entering the PIN incorrectly enough times to exceed the limit, thereby blocking the card from any 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 this scenario, making it impossible to enter the PIN remotely since the user should be physically close and aware of someone entering the PIN.

Since the question of whether current tools can analyse the unlinkability of UTX remains open (as we have demonstrated in Section 4.3.3), future work includes developing automated methods for proving the privacy of security protocols which would lower the analysis effort – the proof of Theorem 1 in [16] illustrates the high cost of the manual analysis of a single protocol.

ACKNOWLEDGMENTS

Semen Yurkov is supported by the Luxembourg National Research Fund through grant PRIDE15/10621687/SPsquared. We thank the

reviewers for their thorough analysis of our threat model and assessment of the scope of related work on tools.

REFERENCES

[1] 2011. EMV Integrated Circuit Card Specifications for Payment Systems. Books 1-4. https://www.emvco.com/document-search/ Accessed: 26-08-2021.

[2] 2012. *EMV ECC Key Establishment Protocols.* RFC until 28th January 2013. EMVCo LLC. http://www.emvco.com/specifications.aspx?id=243 Accessed: 01-04-2020.

[3] 2014. EMV Next Generation. Next Generation Kernel System Architecture Overview.

[4] 2017. Common Criteria for Information Technology Security Evaluation. Part 2: Security functional components. Version 3.1, Revision 5, CCMB-2017-04-002.

[5] 2019. EMVCo Statement – The Advancement of EMV Chip Specifications. https://www.emvco.com/specifications/ Accessed: 30-03-2023.

[6] 2021. EMV Contactless Specifications for Payment Systems. Book A. https://www.emvco.com/document-search/ Accessed: 20-09-2021.

[7] 2021. *European Court of Human Rights admits RSF complaint against the BND's mass surveillance.* https://rsf.org/en/news/european-court-human-rights-admits-rsf-complaint-against-bnds-mass-surveillance Accessed: 25-01-2022.

[8] 2022. *German police used a tracing app to scout crime witnesses.* https://www.washingtonpost.com/world/2022/01/13/german-covid-contact-tracing-app-luca/ Accessed: 25-01-2022.

[9] Martín Abadi, Bruno Blanchet, and Cédric Fournet. 2018. The Applied Pi Calculus: Mobile Values, New Names, and Secure Communication. *Journal of the ACM* 65, 1 (2018), 1:1–1:41. https://doi.org/10.1145/3127586

[10] David Basin, Jannik Dreier, and Ralf Sasse. 2015. Automated Symbolic Proofs of Observational Equivalence. In *Computer and Communications Security (CCS'15)*. Association for Computing Machinery. https://doi.org/10.1145/2810103.2813662

[11] David Basin, Ralf Sasse, and Jorge Toro-Pozo. 2021. The EMV Standard: Break, Fix, Verify. In *IEEE Symposium on Security and Privacy (S&P'21)*. 1766–1781. https://doi.org/10.1109/SP40001.2021.00037

[12] Lejla Batina, Jaap-Henk Hoepman, Bart Jacobs, Wojciech Mostowski, and Pim Vullers. 2010. Developing efficient blinded attribute certificates on smart cards via pairings. In *International Conference on Smart Card Research and Advanced Applications (CARDIS'10) (LNCS, Vol. 6035)*. Springer, 209–222. https://doi.org/10.1007/978-3-642-12510-2_15

[13] Bruno Blanchet. 2001. An efficient cryptographic protocol verifier based on prolog rules. In *14th IEEE Computer Security Foundations Symposium (CSF'01)*. IEEE, 82–96. https://doi.org/10.1109/CSFW.2001.930138

[14] Bruno Blanchet. 2009. Automatic verification of correspondences for security protocols. *Journal of Computer Security* 17, 4 (2009), 363–434. https://doi.org/10.3233/JCS-2009-0339

[15] Bruno Blanchet, Martín Abadi, and Cédric Fournet. 2008. Automated verification of selected equivalences for security protocols. *The Journal of Logic and Algebraic Programming* 75, 1 (2008), 3–51. https://doi.org/10.1016/j.jlap.2007.06.002

[16] Sergiu Bursuc, Ross Horne, Sjouke Mauw, and Semen Yurkov. 2023. Provably Unlinkable Smart Card-based Payments (Extended Version). https://arxiv.org/abs/2309.03128 Accessed: 07-09-2023.

[17] Sergiu Bursuc, Ross Horne, Sjouke Mauw, and Semen Yurkov. 2023. ProVerif models of UTX. https://github.com/utxprotocol/proverif Accessed: 05-09-2023.

[18] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology — EUROCRYPT 2001*, Birgit Pfitzmann (Ed.). Springer Berlin Heidelberg, 93–118. https://doi.org/10.1007/3-540-44987-6_7

[19] Vincent Cheval and Itsaka Rakotonirina. 2023. Indistinguishability Beyond Diff-Equivalence in ProVerif. In *36th IEEE Computer Security Foundations Symposium (CSF'23)*. IEEE. https://doi.org/10.1109/CSF57540.2023.00036

[20] D. Dolev and A. Yao. 1983. On the security of public key protocols. *IEEE Transactions on Information Theory* 29, 2 (1983), 198–208. https://doi.org/10.1109/TIT.1983.1056650

[21] Saar Drimer and Steven J. Murdoch. 2007. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In *16th USENIX Security Symposium (USENIX Security 07)*. USENIX Association.

[22] Petr Dzurenda, Sara Ricci, Jan Hajny, and Lukas Malina. 2017. Performance Analysis and Comparison of Different Elliptic Curves on Smart Cards. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. 365–36509. https://doi.org/10.1109/PST.2017.00050

[23] Maximilian Engelhardt, Florian Pfeiffer, Klaus Finkenzeller, and Erwin Biebl. 2013. Extending ISO/IEC 14443 Type A Eavesdropping Range using Higher Harmonics. In *Proceedings of 2013 European Conference on Smart Objects, Systems and Technologies (SmartSysTech)*. 1–8.

[24] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. 2010. Practical NFC Peer-to-Peer Relay Attack Using Mobile Phones. In *Radio Frequency Identification (LNCS, Vol. 6370)*, Siddika Berna Ors Yalcin (Ed.). Springer, 35–49. https://doi.org/10.1007/978-3-642-16822-2_4

[25] René Habraken, Peter Dolron, Erik Poll, and Joeri de Ruiter. 2015. An RFID Skimming Gate Using Higher Harmonics. In *Radio Frequency Identification (LNCS, Vol. 9440)*, Stefan Mangard and Patrick Schaumont (Eds.). Springer, 122–137. https://doi.org/10.1007/978-3-319-24837-0_8

[26] Ross Horne, Sjouke Mauw, and Semen Yurkov. 2022. Unlinkability of an Improved Key Agreement Protocol for EMV 2nd Gen Payments. In *35th IEEE Computer Security Foundations Symposium (CSF'22)*. IEEE. https://doi.org/10.1109/CSF54842.2022.9919666

[27] Ross Horne, Sjouke Mauw, and Semen Yurkov. 2023. When privacy fails, a formula describes an attack: A complete and compositional verification method for the applied pi-calculus. *Theoretical Computer Science* 959 (2023), 113842. https://doi.org/10.1016/j.tcs.2023.113842

[28] Apple Inc. 2023. Apple Pay. https://www.apple.com/apple-pay/ Accessed: 31-03-2023.

[29] Gavin Lowe. 1997. A hierarchy of authentication specifications. In *Proceedings 10th Computer Security Foundations Workshop*. IEEE, 31–43. https://doi.org/10.1109/CSFW.1997.596782

[30] Lukas Malina, Petr Dzurenda, Jan Hajny, and Zdenek Martinasek. 2018. Assessment of Cryptography Support and Security on Programmable Smart Cards. In *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. 1–5. https://doi.org/10.1109/TSP.2018.8441334

[31] Steven J. Murdoch, Saar Drimer, Ross Anderson, and Mike Bond. 2010. Chip and PIN is Broken. In *IEEE Symposium on Security and Privacy (S&P'10)*. 433–446. https://doi.org/10.1109/SP.2010.33

[32] David R. Novotny, Jeffrey R. Guerrieri, Michael Francis, and Kate Remley. 2008. HF RFID electromagnetic emissions and performance. In *2008 IEEE International Symposium on Electromagnetic Compatibility*. 1–7. https://doi.org/10.1109/ISEMC.2008.4652133

[33] Andreea-Ina Radu, Tom Chothia, Christopher J.P. Newton, Ioana Boureanu, and Liqun Chen. 2022. Practical EMV Relay Protection. In *IEEE Symposium on Security and Privacy (S&P'22)*. 433–452. https://doi.org/10.1109/SP46214.2022.00026

[34] Eric R. Verheul. 2001. Self-blindable credential certificates from the Weil pairing. In *International Conference on the Theory and Application of Cryptology and Information Security (LNCS, Vol. 2248)*, Colin Boyd (Ed.). Springer, 533–551. https://doi.org/10.1007/3-540-45682-1_31

[35] Pim Vullers and Gergely Alpár. 2013. Efficient Selective Disclosure on Smart Cards Using Idemix. In *Policies and Research in Identity Management*, Simone Fischer-Hübner, Elisabeth de Leeuw, and Chris Mitchell (Eds.). Springer Berlin Heidelberg, 53–67. https://doi.org/10.1007/978-3-642-37282-7_5

[36] Semen Yurkov. 2023. *Analysis of Smartcard-based Payment Protocols in the Applied Pi-calculus using Quasi-Open Bisimilarity.* PhD thesis. University of Luxembourg. http://hdl.handle.net/10993/55510