Getting Chip Card Payments Right*

David Basin(✉)[1][0000−0003−2952−939X], Xenia Hofmeier[1][0009−0002−6909−8010], Ralf Sasse[1][0000−0002−5632−6099], and Jorge Toro-Pozo[2]

[1] Department of Computer Science, ETH Zurich, Switzerland
{basin,xenia.hofmeier,ralf.sasse}@inf.ethz.ch
[2] SIX Digital Exchange, Switzerland
jorge.toro@sdx.com

Abstract. EMV is the international protocol standard for smart card payments and is used in billions of payment cards worldwide. Despite the standard's advertised security, various issues have been previously uncovered, deriving from logical flaws that are hard to spot in EMV's lengthy and complex specification. We have formalized various models of EMV in Tamarin, a symbolic model checker for cryptographic protocols. Tamarin was extremely effective in finding critical flaws, both known and new, and in many cases exploitable on actual cards. We report on these past problems as well as followup work where we verified the latest, improved version of the protocol, the EMV kernel C8. This work puts C8's correctness on a firm, formal basis, and clarifies which guarantees hold for C8 and under which assumptions. Overall our work supports the thesis that cryptographic protocol model checkers like Tamarin have an essential role to play in improving the security of real-world payment protocols and that they are up to this challenge.

Keywords: Formal Methods · Security · Model Checking · EMV.

1 Introduction

EMV is the de facto standard for smart card payments. It is named after Europay, Mastercard, and Visa, the three founding companies that initiated this standard, which is now managed by EMVCo. With 12.9 billion EMV cards in circulation and over 90 percent of card payments using EMV, the EMV protocol is by far the most prominent in-person payment protocol used worldwide [11].

EMVCo provides specifications for the different technologies used for card, mobile, and online payment. The card payment standards include specifications for *contact* transactions, where the payment card must be inserted into the payment terminal, and *contactless* transactions, where the card and terminal communicate wirelessly over NFC. The contactless protocol has numerous variants called *kernels*, associated with the different EMVCo members.

* We thank Mastercard for their past support. All opinions and conclusions expressed in this paper are those of the authors.

1.1 Attacks on EMV

Security is central to the proper functioning and acceptance of electronic payments. Unfortunately, there has been a long history of attacks on EMV cards and protocols. These range from cloning attacks [17], where a functioning card clone is produced, to PIN-bypass attacks where transactions that should require a PIN are performed without it. Early PIN-bypass attacks targeted contact transactions. For example, the attack of Murdoch et al. [15] uses a wired machine-in-the-middle (MITM) infrastructure between the card and the terminal. While effective, such attacks are not practically relevant as the MITM infrastructure is difficult to conceal. In contrast, MITM attacks on the NFC channel are a serious threat as the attack infrastructure is inconspicuous. Such attacks can be carried out, for example, using two smart phones that forward and modify the communication between the card and the terminal. Examples of this are the recent PIN-bypass attacks on EMV contactless [5–7, 16].

Given EMV's lengthy and complex specification, running over 2,000 pages, it is not surprising that many weaknesses went undiscovered for quite some time, even long after the protocol became widely used. The weaknesses exploited were manifold and included issues in EMV's legacy modes, like the magstripe mode exploited by the cloning attack of Roland and Langer [17], the interoperability of the different kernels, as exploited by our previous card brand mixup attack [5], and most importantly weaknesses in EMV's options for different authentication methods that authenticate different data, as highlighted by our previous analysis of the EMV protocol [6].

To address weaknesses like those above, and to improve overall payment security, EMVCo recently developed its new, eighth, contactless kernel called C8. This new kernel reduces the protocol's complexity, introduces new security features, and removes known insecure features such as magstripe mode. Its new security features include modern cryptographic algorithms, privacy protection mechanisms, relay protection, and new authentication methods.

1.2 Applying Formal Methods

In this paper, we focus on the use of the Tamarin prover [3, 14, 18], a robust verification tool for cryptographic protocols, to uncover weaknesses in EMV and validate recently deployed countermeasures and other protocol improvements. We will explain Tamarin in Section 2.2 and we provide a brief survey here on how it has been used in the past to analyze EMV.

As mentioned above, we analyzed the EMV contact and contactless protocol in our previous work [6] using Tamarin. Our analysis revealed known attacks, such as the PIN-bypass attack on contact transactions by Murdoch et al. [15], as well as new attacks on EMV contactless. These new attacks include a PIN-bypass attack on the Visa kernel that was demonstrated on live systems and a separate attack that targets merchants. In the latter attack, the adversary pays for some goods, the terminal accepts the transaction, the adversary walks out of

the store with the goods, and the bank later declines the transaction. This attack leaves the merchant "holding the bag" in that the adversary gets the goods but the merchant is cheated out of payment. These attacks target EMV transactions with weak authentication methods. By evaluating the security properties of transactions with different authentication methods, we could not only identify such attacks but also identify secure methods and prove that the security properties hold for our model of transactions with these methods. In this way, we could prove the security of the most common contactless Mastercard transactions.

Subsequent work of ours extended our EMV model to also specify the routing of transaction information between the terminal and bank [5]. As this communication is not described by the public specification of EMV, our initial models made assumptions about this communication. These assumptions included that transactions between a Mastercard card and a terminal running the Visa kernel would not be accepted by the bank. However, our experiments showed that this was not the case. After adapting our model to account for such transactions, Tamarin found an attack that we named the "card brand mixup attack." The attack is quite surprising: the terminal is tricked into running the Visa kernel with a Mastercard card, which in turn allows the adversary to perform the PIN-bypass attack targeting the Visa kernel. This attack was also successfully tested on live systems. Mastercard implemented countermeasures against this attacks, which we verified by attempting, and failing, to reproduce the attacks after their countermeasures were in place.

Over time, we found many different kinds of attacks using Tamarin. This reflects the multiple models we made, at increasing levels of precision, which allowed us to produce stronger verification results or, alternatively, find increasingly subtle problems in the design of the different EMV kernels. This was the case for our initial models, which were unable to capture the card brand mixup attack as routing aspects were initially omitted. Another example of this was that our initial models of EMV employed certain abstractions to aid Tamarin's termination. In particular, we abstracted away from certain failure modes that were part of the complex decision tree used to determine when the terminal rejects or accepts transactions. As a result, our original models were too abstract to capture our latest attack on the Mastercard kernel, which exploits failure modes associated with certificate lookup failures [7].

Tamarin however is a verification tool, not just a tool for attack finding. In all our previous work, after discovering attacks, we used Tamarin to verify proposed countermeasures. In addition to our own work, Tamarin has also been used by other researchers to verify EMV protocol extensions for relay protection, as done by Radu et al. [16] and Coppola et al. [8].

1.3 Contributions

In this work, we analyze the security of the new C8 kernel using Tamarin. In contrast to past work on formally modeling EMV, which occurred after the kernels analyzed were implemented and released, we report here on the analysis of C8 during its standardization. This provides confidence in the protocol's design

before its deployment and provides an alternative to the many iterations of penetrate-and-patch, caused by the design errors discovered and exploited in the past.[3]

Our model of C8 includes its new security features, including its new authentication methods, its relay resistance protocol, and its new privacy features. As C8 is based on the other EMV protocols, we could reuse parts of the Tamarin models from our previous verification efforts [5, 6], which sped up the modeling process significantly. After modeling, we analyzed C8's different configurations individually, each configuration consisting of different supported authentication methods for the card and cardholder.

Our analysis shows that C8 is a well-designed protocol that can be used securely, although not with all configurations. Specifically, we use Tamarin to identify both secure and insecure configurations, prove the security properties of the transactions with secure configurations, and find potential attacks on transactions with insecure configurations. We find, for example, that at least one of the available authentication methods must be used in each transaction to prevent attacks. Moreover, offline accepted transactions specifically require the terminal to verify the card's certificate. Overall, our analysis puts the security of this new EMV kernel on a firm, formal basis by highlighting assumptions on its configuration and implementation that are necessary and sufficient for its secure usage.

Outline. In Section 2 we provide background on C8 and Tamarin. In Section 3 we describe our Tamarin model of C8, including its desired security properties. In Section 4, we present our results and in Section 5 we draw conclusions.

2 Background

2.1 The C8 Protocol

The EMV specification describes the communication between a payment card and a payment terminal consisting of the terminal's commands followed by the card's responses. At the end of such a transaction, the terminal sends the transaction data to the bank that issued the card.

As observed in the introduction, EMV offers two protocol variants: contact and contactless. Moreover, there are variants of the contactless protocol called kernels and further complexity is introduced by the kernels' different configuration options. We first provide a general overview of EMV transactions that apply to all protocol variants and configurations.

To prevent fraud, the payment card authenticates transaction data to two different parties and in two different ways: once to the bank and once to the terminal. The card authenticates to the bank using a message authentication code

[3] Note that our work is unlikely to be the final word on C8's security as it focuses on an abstract model of the design and we cannot rule out other weaknesses that adversaries may exploit, such as errors in the implementation. Moreover, adversaries may have capabilities not captured by our models, such as the ability to carry out side-channel attacks on the cryptography used.

(MAC) that is calculated using a session key derived from a symmetric long-term key shared between the card and the bank. The authentication to the terminal is signature based, and the asymmetric public key associated to the card's private signing key is authenticated using a certificate chain. The authenticated transaction data may differ for the two authentication methods and it also depends on the protocol variant and the configuration. It generally contains static data such as the card number (also called Primary Account Number (**PAN**)), payment details such as the amount and currency, and transaction specific data, for example, identifying the configuration.

In addition to the card authenticating the transaction data, the cardholder's presence is ensured using a Cardholder Verification Method (CVM). The CVMs include providing a PIN to the terminal, providing a paper signature, and Consumer Device CVM (CDCVM) where the card authenticates the cardholder, usually using a mobile device such as a smart phone. The PIN can either be verified offline by the terminal or online by the bank. Offline PIN is only offered by contact transactions, whereas contactless transactions require online PIN. Transactions with a value above the *CVM Required Limit* require some CVM, whereas transactions below this limit allow for no CVM.

At the end of a transaction, the terminal chooses to either decline the transaction offline, authorize the transaction offline, or send the transaction to the bank for online authorization. Note that transactions with online PIN also require online authorization.

The new C8 kernel is described in Book C8 [9] and Book E [10]. It offers many improved security features over past kernels. These include new methods to authenticate the transaction to the terminal and bank, modern cryptographic algorithms, privacy protection mechanisms, and a relay resistance protocol. Known insecure features such as the contactless magstripe mode were removed from the specification. The CVM performed is now chosen by the card and cards support Elliptic Curve Cryptography (ECC), AES, and RSA. Figure 1 shows a simplified C8 transaction. In what follows, we describe the abstraction of C8 that we modeled in Tamarin.

In C8, the card authenticates the transaction to the bank as with other EMV kernels using an Application Cryptogram (**AC**). The **AC** is a MAC computed over transaction data using a session key derived from the symmetric long-term key **mk** shared between the card and the bank and the Application Transaction Counter (**ATC**), which is increased for each transaction.

The signature-based authentication to the terminal used in prior EMV kernels is replaced by a MAC-based authentication. The card constructs the Enhanced Data Authentication (**EDA**)-**MAC** using a session key. The **EDA-MAC** authenticates the **AC** and the new Issuer Application Data (**IAD**)-**MAC**. The **IAD-MAC** is calculated over transaction data including the transaction amount, a terminal-sourced nonce, and a card-sourced nonce.

For the card to create these two MACs, the card and terminal establish symmetric session keys using a *blinded Diffie-Hellman* key exchange. The card has a static private-public key pair ($\mathbf{d_C}$, $\mathbf{Q_C}$) while the terminal generates a fresh

ephemeral key pair ($\mathbf{d_T}$, $\mathbf{Q_T}$) for each transaction. The key exchange starts with the terminal sending its ephemeral public key $\mathbf{Q_T}$ to the card. The card generates a random *blinding factor* $\mathbf{r}$ and calculates a blinded public key $\mathbf{R}$ from its secret key $\mathbf{d_C}$ and $\mathbf{r}$. The card also calculates a shared secret $\mathbf{z} = \mathbf{Q_T}^{\mathbf{r} \times \mathbf{d_C}}$ from its secret key $\mathbf{d_C}$, the blinding factor $\mathbf{r}$, and the terminal's ephemeral public key $\mathbf{Q_T}$. From this shared secret $\mathbf{z}$, it derives two symmetric session keys, one for confidentiality $\mathbf{sk_c}$ and one for integrity $\mathbf{sk_i}$, using two Key Derivation Functions (KDF_c and KDF_i). The card then encrypts $\mathbf{r}$ with the session key for confidentiality $\mathbf{sk_c}$ and sends it together with the blinded public key $\mathbf{R}$ to the terminal. The terminal can derive the shared secret $\mathbf{z}$ from the blinded public key $\mathbf{R}$ and its ephemeral secret key $\mathbf{d_T}$ and recover $\mathbf{r}$.

To protect this key exchange from a machine-in-the-middle (MITM) attack, there are two options: *local authentication* or *copying the **IAD-MAC** into **IAD***. During local authentication, the terminal authenticates the card's public key $\mathbf{Q_C}$ by validating the card's certificates and the blinding factor $\mathbf{r}$. This verification is only performed if the terminal and card support local authentication. The second option of copying the **IAD-MAC** to the **IAD** lets the bank detect a MITM attack through the **IAD-MAC**. Namely, the card authenticates the **IAD-MAC** to the bank by including it in the **AC** and the terminal recalculates the **IAD-MAC** with its view of the session key and sends it to the bank as part of the **IAD**. The bank then verifies that the **IAD-MAC** received from the card in the **AC** is the same as the **IAD-MAC** received from the terminal in the **IAD**. Including the **IAD-MAC** in the **AC** is optional for the card. If the card performs this action, it indicates to the terminal that it must include the **IAD-MAC** in the **IAD**. We thus call this authentication method *copy **IAD-MAC** into **IAD***.

The use of blinding provides privacy protection against eavesdroppers by protecting sensitive data. Namely, data that identifies the card, like the card number (**PAN**), is encrypted with the session key for confidentiality $\mathbf{sk_c}$.

C8 also supports the Relay Resistance Protocol (RRP), which protects against relay attacks. In a relay attack, messages between a terminal and a remote card are forwarded and potentially modified. The RRP prevents such attacks by requiring the card to be close to the terminal. The RRP includes the exchange of two nonces, the terminal-sourced Terminal Relay Resistance Entropy (**TRRE**) and the card-sourced Device Relay Resistance Entropy (**DRRE**). The terminal's **TRRE** is included in the **AC** and **IAD-MAC** and the card's **DRRE** is included in the **IAD-MAC**. The terminal times this exchange to determine the distance between the card and terminal. If the estimated distance exceeds a given limit, the terminal may decline the transaction.

2.2 The Tamarin Prover

Tamarin is an automated tool for modeling and analyzing cryptographic protocols. Given a protocol model, security properties, and adversary capabilities, Tamarin can prove that the property holds for the protocol and adversary model, or provide an attack violating the property. Due to the undecidable nature of the underlying verification problem, Tamarin may sometimes fail to terminate.

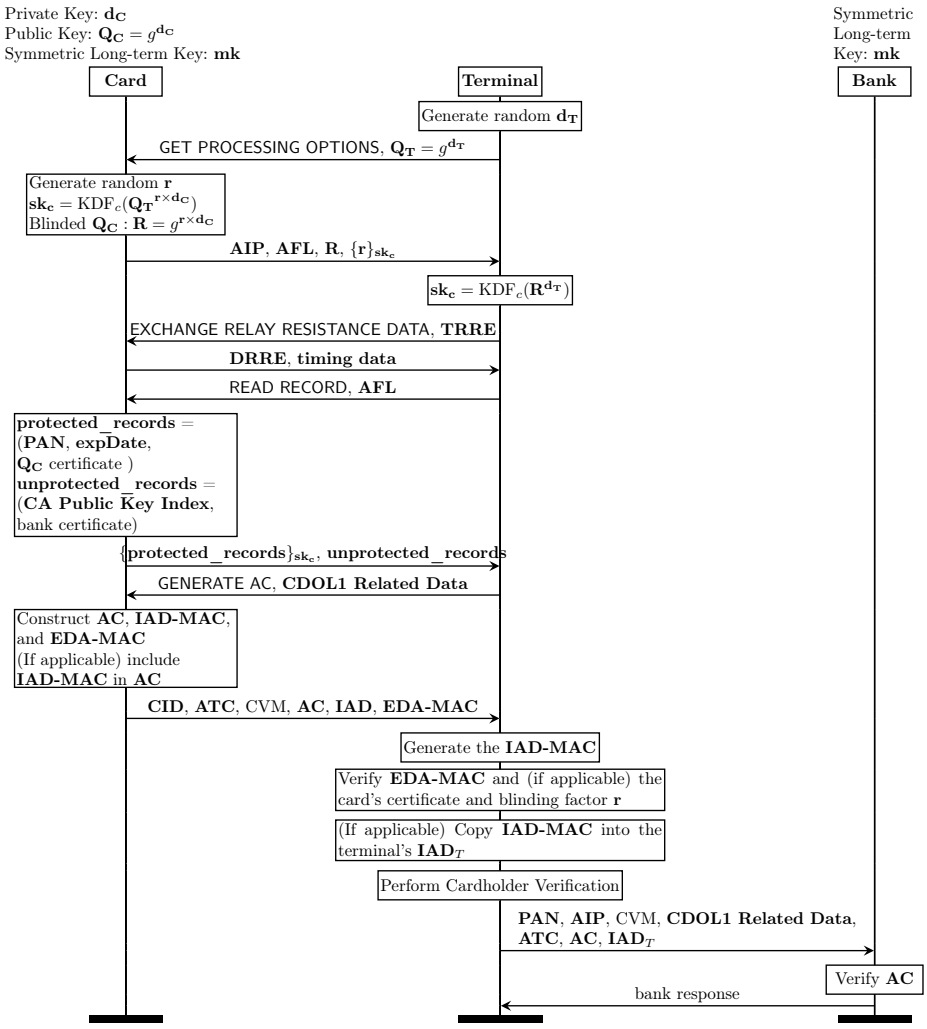

Fig. 1. Message sequence chart of our C8 protocol model abstraction.

Tamarin analyzes designs, not code, and cryptographic functionality is handled not by considering its implementation, but rather its abstract properties. In particular, Tamarin works with a symbolic model of protocols where bitstring messages are represented as terms, cryptographic operators are modeled as function symbols, and their properties are expressed with equational theories. For example, symmetric encryption is modeled by a function senc of arity two and the symmetric decryption is modeled using the function sdec also of arity two. The properties of these functions are modeled by the equation $\mathrm{sdec}(\mathrm{senc}(m,k),k) = m$, expressing that the decryption of a ciphertext with the correct key results in the original plaintext.

In Tamarin, *facts* are used to represent agents' local states and messages on the network. A fact has the form $\mathsf{F}(t_1,...,t_n)$, consisting of the fact's name F applied to terms t_i. The protocol's state consists of the agents' local states and the messages on the network and thus is represented by a multiset of facts.

The protocol's state makes a transition to a new state by the application of a labeled multiset-rewrite rule. A rule consists of three multisets of facts: the premises, the labels (also called action facts), and the conclusions. A rule can be applied if an instantiation of the premises is a subset of the current protocol state. In that case, the instantiations of the premises are removed from the protocol state and replaced by instantiations of the conclusion under the matching substitution. An exception is made for so-called *persistent* facts, which are facts that stay in the protocol state and are not removed by rule application.

Tamarin's built-in adversary is a Dolev-Yao adversary who controls the network. This means the adversary learns every message sent over the network and can send messages that it knows. Moreover, the adversary can derive messages from those messages it knows. However, cryptography is assumed to work perfectly, meaning, for example, that the adversary requires a matching decryption key to derive the plaintext from a ciphertext. This is an abstraction of the real world where, for example, side-channel attacks are possible or cryptography may leak partial information about the plaintext.

A protocol P is modeled by a set of multiset rewrite rules. An execution of P is represented by a sequence of applications of rules from this set. The trace α of such an execution consists of the associated sequence of labels $(\alpha_1,...,\alpha_n)$ of the rules applied, where α_i is a multiset of action facts. We denote the set of all of P's traces by $\mathrm{traces}(P)$. Security properties are defined as trace properties, which are expressed as first-order formulas on traces, called *lemmas* (as they must be proven), and Tamarin analyzes whether they hold for all (or, in special cases, some) of the protocol's traces.

3 Tamarin Model of C8

In this section, we present our Tamarin model of C8, the formalized security properties, and our methodology. Our model is available at [4].

3.1 Protocol Model

We model the actions of three parties: the card, the terminal, and the bank. The channel between the terminal and the bank is modeled as a secure channel, which provides confidentiality and authenticity. The NFC channel between the card and the terminal is modeled as being controlled by a Dolev-Yao adversary since the adversary can tamper with the NFC channel as it is not cryptographically protected. That protection is what the C8 protocol should achieve.

All EMV kernels rely on a public key infrastructure so that the terminal can authenticate the card's public key $\mathbf{Q_C}$. We abstractly model this as the CA's self-signed certificate, which can be accessed by the terminal, the bank's certificate signed by the CA, and the card's public key certificate signed by the bank, which the card stores and sends to the terminal. The CA's certificate and the card's records are modeled as persistent facts, accessed by the bank and card respectively. The symmetric long-term key **mk** shared between the card and the bank is also modeled as a persistent fact that the card and bank can access.

EMV requires cardholder verification for high value transactions. In practice, terminals should reject high value transactions with a card that apparently does not support cardholder verification and instruct the cardholder to use the contact interface. Thus, our model does not allow a terminal to complete high value transactions without cardholder verification.

As previously mentioned, the C8 protocol is quite complex. In addition to those abstractions that are standard in symbolic models, we incorporated further abstractions and simplifications where required to aid proof termination. This includes omitting some configuration options, such as only modeling Version 2, which we described in Section 2.1, requiring the optional RRP, omitting some features such as Data Storage or the Select phase, and omitting some data objects, such as the **PDOL**.

As the C8 kernel is based on the other EMV kernels, we were able to reuse substantial parts of our previous EMV models [5, 6] for our new model of C8. While a public specification of C8 was available for our work, this specification only covers the communication between the card and the terminal and not the actions of the bank. In addition, some proprietary data such as the **CDOL1 Related Data** is not part of the specification. In our previous works, we clarified underspecified protocol aspects by inspecting actual transaction transcripts that were collected using our MITM infrastructure. However, this was not possible for C8, as it is not yet implemented on publicly available cards. Fortunately, we were able to discuss the protocol with one of the EMVCo partners to resolve ambiguities or missing information.

3.2 Security Properties

The EMV protocol should provide guarantees to the cardholder, the merchant, and the bank. We express these guarantees as security properties that hold from the perspective of these three parties. We formulate the same properties as in our previous work [6] plus additional ones, which we explain next.

Authentication After a successful transaction, the card's bank transfers funds from the cardholder's account to the merchant's account. The C8 protocol should provide the bank the information needed for this transfer. This includes the card's **PAN**, which identifies the cardholder's account, the terminal's identity, which identifies the merchant's account, the amount, and the currency. Providing the correct information means that the card, the terminal, and the bank should agree on this data. Situations (which likely are attacks) should be avoided where the card sees some low value while the terminal and bank agree on a different high value, or where a card sees its own **PAN** whereas the terminal sees a different card's **PAN** that was not involved in the transaction.

We formalize such agreement as authentication properties. Namely, we formalize two injective agreement properties [12], one for the authentication of the card and bank to the terminal and one for the card and terminal to the bank. Injective agreement states that whenever agent A in role r_A finished the protocol apparently with agent B in role r_B, then B was running the protocol with A and both agree on the data t. In addition, injective agreement rules out replay attacks by enforcing unique protocol runs with the data t.

The views of the protocol participants are expressed with the action facts Commit and $\mathsf{Running}$, where $\mathsf{Commit}(A, B, \langle r_B, r_A, t\rangle)$ states that agent A in role r_A finished the protocol apparently with agent B in role r_B with data t and $\mathsf{Running}(B, A, \langle r_B, r_A, t\rangle)$ states that agent B in role r_B is running the protocol with agent A in role r_A and data t. Note that the order of the arguments A and B, which are the agents in Commit and $\mathsf{Running}$, is intentionally swapped as the first argument represents the agent doing that action. Additionally, we mark agents as being expected to be non-compromised, also called honest, with Honest. We also track when agents have been compromised with $\mathsf{Compromised}$. When an agent that should not be compromised has been compromised, then the property holds vacuously, which we explain in more detail shortly. Using these action facts, we formalize injective agreement to the terminal in the following lemma, which we subsequently check with Tamarin.

Lemma 1 (Authentication to the Terminal). *A protocol P satisfies* authentication to the terminal *if for every trace $\alpha \in traces(P)$:*

$$
\begin{aligned}
\forall T, P, r, t, i.\, &\mathsf{Commit}(T, P, \langle r, \texttt{'Terminal'}, t\rangle) \in \alpha_i \implies \\
&\big(\exists j.\ \mathsf{Running}(P, T, \langle r, \texttt{'Terminal'}, t\rangle) \in \alpha_j \wedge \\
&\quad \nexists i_2, T_2, P_2.\ \mathsf{Commit}(P_2, T_2, \langle r, \texttt{'Terminal'}, t\rangle) \in \alpha_{i_2} \wedge i_2 \neq i\big) \vee \\
&\exists A, k.\ \mathsf{Honest}(A) \in \alpha_i \wedge \mathsf{Compromise}(A) \in \alpha_k.
\end{aligned}
$$

The first two lines of this lemma express that whenever the terminal T finished the protocol with the apparent communication partner P in role $r \in \{\texttt{'Card'}, \texttt{'Bank'}\}$, then P was running the protocol with T and both agree on the transaction data t. The third line specifies unique protocol runs with transaction data t by forbidding any terminals to finish the protocol with t apart from the above terminal that committed at time point i. In other words, should such a replay be possible in the protocol, it would violate this property.

In our model, we consider the security of the protocol in the presence of compromised agents. Compromise means that an agent's key material has been revealed and the adversary can thereby impersonate that agent. The compromisable key material includes the private keys of the CA, bank, and card, as well as the symmetric long-term key **mk** shared between the card and the terminal, and the terminal's ephemeral secret key $\mathbf{d_T}$. While a protocol cannot provide authentication guarantees when run with compromised agents, it should for sessions involving non-compromised agents, even when other agents (not involved in the session) are compromised. This is expressed in the last line of the above lemma. Agents involved in the session are named with the action fact Honest. The property does not have to hold if these agents were compromised, indicated by Compromise.

In addition to agreeing on the **PAN**, terminal identity, amount, and currency, the parties should agree on control data to ensure the correct transaction flow. This ensures that the parties have the same view of the performed transaction. This guarantees, for example, that if the card expects the terminal to perform online PIN, then the terminal did so. Thus, we consider the following transaction data to be agreed upon (i.e., the term t in Lemma 1): the **PAN**, the **AIP**, the CVM, the **ATC**, the **CDOL1**, the **AC**, the **IAD**, and the **AID**.

The authentication to the bank is expressed in a lemma very similar to the one above. The only differences is that the ground term $'\texttt{Terminal}'$ is replaced by $'\texttt{bank}'$. Moreover, we formulated additional security properties, which we discuss next and formalize in Appendix A.

Bank Accepts The merchant not only requires that the correct information for the fund transfer is provided, but also that the merchant receives their funds after a successful transaction. In other words, the bank should not decline transactions that were previously accepted by the terminal. This prevents the scenario described in the introduction where the adversary pays with a card for goods, the terminal accepts, the adversary walks out with the goods, and afterwards, the bank declines the transaction. Thus the merchant does not receive the funds for the purchase. This is especially relevant for offline-capable terminals that do not request online authorization.

Secrecy The third property concerns the secrecy of critical data, i.e., the adversary cannot learn this data. This data includes the card's PIN and key material, consisting of the symmetric long-term key **mk** and the asymmetric secret key $\mathbf{d_C}$. The PIN should stay secret as criminals could otherwise steal the card and use the PIN and the card to pay for high amounts or withdraw money. The key material should stay secret as the adversary could misuse it to forge transactions.

Privacy The C8 kernel introduces the blinded Diffie-Hellman key exchange to encrypt sensitive card-sourced data. This sensitive data includes the blinding factor **r** and any data returned by the READ RECORD response that uniquely

identifies the card, for example, the card's **PAN**, certificate, and public key $\mathbf{Q_C}$. The encryption of this data protects the cardholder's privacy as, without it, an adversary observing data such as the **PAN** could track the cardholder's movement. We used Tamarin to prove the secrecy of the card's **PAN**.

Relay Resistance For the C8 kernel, we verify properties of C8's Relay Resistance Protocol (RRP). We specify *relay resistance* using the formalism defined by Mauw et. al [13]. They reduce the correctness of distance-bounding protocols, such as RRP, to the order of messages being sent and received and they abstract away time. For C8's RRP, this means that the following actions must be performed sequentially in the following order: first, the terminal sends the EXCHANGE RELAY RESISTANCE DATA command with its nonce **TRRE**, then the card receives this message and responds with the **DRRE**, and finally the terminal receives this message. If these actions were not performed in this order, an adversarial card could send the response before the card sent the command. This would reduce the terminal's time measurements and thus reduce the estimated distance.

Note that our symbolic abstraction of relay resistance does not cover timing and physical layer attacks on RRP. Thus, our model does not capture relay attacks exploiting inaccurate timings, or exploiting properties of the physical layer. As a result, attacks like Radu et al's. [16] that exploit inaccurate timings and the Early-Detect and Late-Commit attacks targeting the NFC layer pointed out by Coppola et al. [8] fall outside of our analysis.

Executability As a sanity check, we prove *executability* lemmas. These lemmas describe an expected protocol execution without adversary interference and provide a sanity check that the protocol is not inoperable due to modeling errors.

3.3 Analysis Approach

As mentioned, C8 offers multiple configuration options. As depicted in Table 1, we model the options to perform local authentication, copying the **IAD-MAC** into the **IAD**, performing online PIN or no CVM, and high or low value transactions. Each transaction has a fixed combination of configuration options, corresponding to instances of these four parameters, which we call a *configuration*. In our analysis of C8, we follow the approach taken in [5, 6]: we model transactions arising from all configurations running and interacting in parallel. In our formalization of the security properties though, we consider each configuration separately. To do this, for each configuration we generate a so-called *target model* that we analyze with Tamarin. This allows us to determine which configurations are secure or insecure. Details on this approach can be found in [6].

Our previous models considered offline and online authorized transactions in the same model. An attack discovered for this model might only be possible for offline authorized transactions, but not for online authorized transactions, and this would not be apparent from the verification results. The C8 specification

Table 1. The four parameters comprising a configuration

Parameter	Instances	Determines
LocalAuth	- Yes - No	Whether local authentication is performed, i.e. whether the terminal verified the certificates and the blinding factor (note that the **EDA-MAC** is always validated).
CopyIAD	-Yes - No	Whether the **IAD-MAC** is copied into the **IAD** itself and whether the **IAD-MAC** is included in the **AC**.
CVM	- NoCVM - OnlinePIN	The CVM used in the transaction.
Value	- Low - High	Whether the transaction amount is below (low) or above (high) the CVM Required Limit.

has the option for online and offline authorized transactions. However, since there is an industry push for online transactions, we additionally analyze online authorized transactions separately. Thus, we analyze the 16 configurations twice: first we consider just online authorized transactions and second we allow for both offline and online authorization. Note that the second model includes all the online traces from the first model and also traces from offline transactions.

4 Results

In this section, we present the results of our analysis of the C8 protocol. In Table 2 we summarize the results for transactions requiring online authorization and in Table 3 we show the results for transactions supporting both offline and online authorization. The tables show for each configuration which lemmas were verified (✓) or falsified (×), the lines of code of the target model, and the time Tamarin required for analyzing the model. For our analysis, we used Tamarin version 1.9.0 [1] on a compute server running Ubuntu 20.04.3 with two Intel(R) Xeon(R) E5-2650 v4 @ 2.20GHz CPUs, with 12 cores each. We used 14 threads and at most 32GB of RAM per configuration.

The analysis of some of the models required hours due to their size and complexity. Moreover, to achieve termination, we needed to write so-called oracles, which are Python scripts that guide Tamarin's proof search. Also note that some of the models supporting offline (and as always online) authorization have longer proof times than the models requiring strictly online authorization. This is because the models supporting offline and online authorization have a larger search space than the models with only online authorization. Additionally, the proofs for the relay resistance lemma did not terminate within a few days for some of the configurations supporting offline authorization, namely Models 3.11 and 3.15 in Table 3, marked with  . Hence, in these cases, we cannot draw conclusions with Tamarin about whether the respective statement holds.

As our model does not allow for high value transactions without CVM, the models with NoCVM and High are not executable. Thus, we did not analyze

Table 2. Results for lemmas for configurations requiring online authorization.

No.	Configuration				Exec.	Bank Auth. to Acc.	Auth. to Term.	Auth. to Bank	Relay Resist.	Lines of code	Analysis time
	LocalAuth	CopyIAD	CVM	Value							
2.1	No	No	OnlinePIN	Low	✓	✓	×	×	×	610	3h29m45s
2.2	No	No	OnlinePIN	High	✓	✓	×	×	×	610	3h26m37s
2.3	No	No	NoCVM	Low	✓	✓	×	×	×	603	2h42m23s
2.4	No	No	NoCVM	High	×	NA	NA	NA	NA	561	4m08s
2.5	No	Yes	OnlinePIN	Low	✓	✓	✓	✓	✓	628	6m35s
2.6	No	Yes	OnlinePIN	High	✓	✓	✓	✓	✓	628	6m02s
2.7	No	Yes	NoCVM	Low	✓	✓	✓	✓	✓	621	7m13s
2.8	No	Yes	NoCVM	High	×	NA	NA	NA	NA	561	4m05s
2.9	Yes	No	OnlinePIN	Low	✓	✓	✓	✓	✓	628	1h18m33s
2.10	Yes	No	OnlinePIN	High	✓	✓	✓	✓	✓	628	1h16m38s
2.11	Yes	No	NoCVM	Low	✓	✓	✓	✓	✓	621	3h31m35s
2.12	Yes	No	NoCVM	High	×	NA	NA	NA	NA	561	4m03s
2.13	Yes	Yes	OnlinePIN	Low	✓	✓	✓	✓	✓	628	7m30s
2.14	Yes	Yes	OnlinePIN	High	✓	✓	✓	✓	✓	628	8m12s
2.15	Yes	Yes	NoCVM	Low	✓	✓	✓	✓	✓	621	8m01s
2.16	Yes	Yes	NoCVM	High	×	NA	NA	NA	NA	561	4m19s

these configurations. Since security properties for non-executable protocols hold trivially, these lemmas are marked with NA.

4.1 Secure Configurations

To begin with, the secrecy of the PIN, the symmetric long-term key **mk**, and the card's secret key $\mathbf{d_C}$, not included in Tables 2 and 3, hold for all configurations. We cover the secrecy of the **PAN** in Section 4.3. Note that the lemma *bank accepts* is verified for all the configurations that require online authorization, shown in Table 2, as the bank can only reject offline authorized transactions but not online authorized transactions (as the bank was already involved in an online transaction and has agreed).

Tamarin verified that most configurations requiring online authorization are secure, namely, the configurations with one or both of the authentication methods *local authentication* and *copy **IAD-MAC** into **IAD*** (Configurations 2.5–2.16 in Table 2). For the configurations supporting offline and online authorization, Tamarin verified the same secure configurations with two exceptions. First, Tamarin found attacks for the configuration without local authentication, with copying the **IAD-MAC** into the **IAD**, with no CVM, and a low value, i.e., Configuration 3.7 in Table 3. These attacks violate the lemmas *bank accepts*, *authentication to the terminal*, and *relay resistance*. We present these attacks in Section 4.2. The second exception was already mentioned above: Tamarin did not terminate for the relay resistance lemma for two configurations supporting offline authorization. However, Tamarin verified the other lemmas for these configurations.

Table 3. Results for lemmas for configurations supporting offline and online authorization.

No.	Configuration				Exec.	Bank Auth. to Acc.	Auth. to Term.	Auth. to Bank	Relay Resist.	Lines of code	Analysis time
	LocalAuth	CopyIAD	CVM	Value							
3.1	No	No	OnlinePIN	Low	✓	✓	×	×	×	641	3h01m30s
3.2	No	No	OnlinePIN	High	✓	✓	×	×	×	641	3h00m31s
3.3	No	No	NoCVM	Low	✓	×	×	×	×	633	4h37m23s
3.4	No	No	NoCVM	High	×	NA	NA	NA	NA	591	7m49s
3.5	No	Yes	OnlinePIN	Low	✓	✓	✓	✓	✓	659	11m41s
3.6	No	Yes	OnlinePIN	High	✓	✓	✓	✓	✓	659	11m29s
3.7	No	Yes	NoCVM	Low	✓	×	×	✓	×	651	4h29m14s
3.8	No	Yes	NoCVM	High	×	NA	NA	NA	NA	591	9m45s
3.9	Yes	No	OnlinePIN	Low	✓	✓	✓	✓	✓	659	1h18m46s
3.10	Yes	No	OnlinePIN	High	✓	✓	✓	✓	✓	659	1h14m59s
3.11	Yes	No	NoCVM	Low	✓	✓	✓	✓	🕓	640	4h14m22s
3.12	Yes	No	NoCVM	High	×	NA	NA	NA	NA	591	10m47s
3.13	Yes	Yes	OnlinePIN	Low	✓	✓	✓	✓	✓	659	13m42s
3.14	Yes	Yes	OnlinePIN	High	✓	✓	✓	✓	✓	659	12m25s
3.15	Yes	Yes	NoCVM	Low	✓	✓	✓	✓	🕓	640	1h16m21s
3.16	Yes	Yes	NoCVM	High	×	NA	NA	NA	NA	591	12m31s

4.2 Insecure Configurations

Tamarin found attacks on two sets of configurations: a MITM attack on blinded Diffie-Hellman that targets configurations without local authentication and without copying the **IAD-MAC** into the **IAD** (Configurations 2.1–2.3 and 3.1–3.3) and an attack targeting configurations with offline authorization and without local authentication and without Online PIN (Configuration 3.7). We describe these attacks next.

MITM Attack on Blinded Diffie-Hellman The Blinded Diffie-Hellman key exchange is vulnerable to the same MITM attack as the naive Diffie-Hellman key exchange without authentication. The C8 protocol prevents this attack by the two authentication methods of local authentication and copying the **IAD-MAC** into the **IAD**. Moreover, this should not be a problem in practice as C8 is designed to be used with these authentication methods. Nevertheless, our formal analysis highlights why these authentication methods are essential and that the protocol must be used as intended: Configurations with neither of these authentication methods, namely Configurations 2.1-2.3 and 3.1-3.3, are vulnerable to this MITM attack. Variations of this attack violate the lemmas *bank accepts*, *authentication to the terminal*, *authentication to the bank*, and *relay resistance*.

We present the MITM attack on the authentication to the terminal in Figure 2. The adversary injects their own public keys $\mathbf{Q_T}_{\text{adv.}} = g^{\mathbf{d_T}_{\text{adv.}}}$, $\mathbf{R}_{\text{adv.}} = g^{\mathbf{r}_{\text{adv.}} \times \mathbf{d_C}_{\text{adv.}}}$ and thus shares the secret $\mathbf{z}_T = g^{\mathbf{r}_{\text{adv.}} \times \mathbf{d_C}_{\text{adv.}} \times \mathbf{d_T}}$ with the terminal and the secret $\mathbf{z}_C = g^{\mathbf{r} \times \mathbf{d_C} \times \mathbf{d_T}_{\text{adv.}}}$ with the card. The adversary can then modify messages that are authenticated using the malicious Diffie-Hellman keys. Our analysis showed that this attack results in a disagreement on the **IAD**, the card's nonce **DRRE**, and the **CID**, which encodes if and how the transaction is

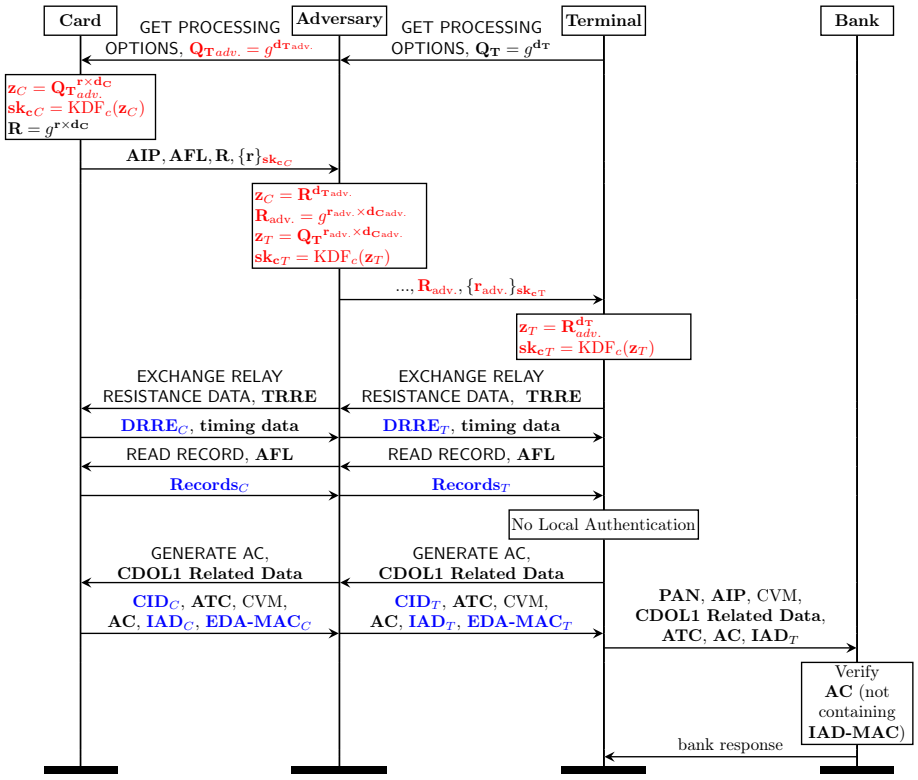

Fig. 2. MITM attack on blinded Diffie-Hellman violating the authenticity property for transactions with configurations without local authentication and without copying the **IAD-MAC** into the **IAD**. The keys injected by the adversary are highlighted in red and the terms that the terminal and the card disagree on are highlighted in blue.

authorized. The rest of the (dynamic) transaction data objects are agreed upon by the parties. The disagreement on the **DRRE** leads to the attack on the relay resistance property that we present in Section 4.4. Disagreement on the **IAD** and **CID** means that these data objects are vulnerable to adversarial modification; however we have not identified a real-world exploit that is possible using such a modification.

This attack is only possible if local authentication is not performed and the **IAD-MAC** is not included in the **IAD**. Both local authentication and copying the **IAD-MAC** into the **IAD** are in principle optional. However, clearly transactions with neither authentication method should be prevented. We now describe the mechanisms with which the terminal and card decide if they perform these authentication methods, starting with copying the **IAD-MAC** into the **IAD**.

According to Book E [10], each payment system may choose how to generate the **AC** and thus including the **IAD-MAC** in the **AC** input is optional, however

it is recommended. If the **AC** authenticates the **IAD-MAC**, the card indicates in the Application Interchange Profile (**AIP**) to the terminal that it must include the **IAD-MAC** in the **IAD**. The terminal follows the instructions to include or not include the **IAD-MAC** inside the **IAD**, but does not perform additional checks regarding this authentication method. Thus, the decision to include or not include the **IAD-MAC** inside the **IAD** lies with the card's payment system.

In contrast, local authentication is only performed if the card and terminal support it. This is indicated by the **AIP** and **kernel qualifier**. In addition, the terminal may choose to decline or request online authorization for transactions where local authentication is not performed or failed. This is configured for each terminal in the Terminal Action Code-Denial (**TAC-Denial**) and Terminal Action Code-Online (**TAC-Online**).

From the discussion above, it follows that terminals could generally require local authentication and cards could generally require including the **IAD-MAC** in the **IAD**. Thus, both the card and the terminal could require an authentication method that prevents this attack. However, we could not verify that this holds in practice as cards and terminals supporting C8 are not yet publicly available.

Exploiting Offline Authorization In offline authorized transactions, the terminal accepts the transaction and later sends the transaction data to the bank for processing. Tamarin found an attack for offline authorized transactions for the configurations without local authentication, with copying the **IAD-MAC** into the **IAD**, and without CVM (Configuration 3.7). Note that the configurations with online PIN (Configurations 3.5 and 3.6) are not vulnerable to this attack, as this attack targets offline authorized transactions and online PIN requires the transaction to be authorized online. The attack also does not violate the lemma *authentication to the bank*, as this lemma only considers online authorized transactions. The attack is similar to the above MITM attack. However, it also targets the configurations with copying the **IAD-MAC** into the **IAD** that are secure for transactions without offline authorization. Copying the **IAD-MAC** into the **IAD** does not prevent the attack because the transaction is authorized offline and the terminal cannot verify the **IAD-MAC** inside the **AC**. However, the bank will decline the transaction after the terminal accepted offline, which violates the *bank accepts* lemma and represents a *merchant holding the bag* attack as presented in our previous work [6] and explained in the introduction.

The above discussion shows that offline authorized transactions must perform local authentication. This is also stated by the specification regarding the RRP: the RRP requires local authentication for offline transactions. As stated above, the terminal can be configured to decline transactions without local authentication or request online authorization for such transactions.

4.3 Privacy

The blinded Diffie-Hellman key exchange was incorporated into C8 to provide privacy protection against a passive adversary by encrypting the **PAN** with the

session key $\mathbf{sk_c}$ to ensure its confidentiality. In our analysis, we consider an active adversary on the NFC channel between the card and the terminal. Tamarin finds a trivial MITM attack on the secrecy of the **PAN**: since the terminal is not authenticated to the card, an active adversary can start a protocol run with a card, establish a session key, and learn the **PAN** since the card sends it encrypted with the established, adversary-known session key. Thus, C8's privacy protection mechanisms are not effective against an active MITM adversary.

This attack is not, however, unexpected. The specification explicitly states that the blinded Diffie-Hellman key exchange only provides privacy protection for a *passive* adversary, not an *active* one. An active adversary can send, receive, and modify messages on the NFC channel and thus communicate with both the card and the terminal. In contrast, a passive adversary can only listen in on the communication between a legitimate card and terminal.

We observe that in situations where a malicious device listens in on such communication, the adversary could just as easily install an active device. For example, an adversary could cooperate with a merchant that presents the cardholder with a terminal emulator that reads card data, including the **PAN**. To avoid raising the cardholder's suspicion, this emulator could first perform the attack, abort the transaction, and afterwards start a new, legitimate transaction and relay this transaction to a legitimate terminal. Due to the large number of different terminal providers and soon also phone-based terminals [2], such a terminal emulator would not raise any suspicions. Moreover, as the terminal emulator is not connected to a bank, such an attack would not be detected by the bank.

4.4 Relay Resistance

Our analysis shows that relay resistance holds for the secure configurations, namely Configurations 2.5-2.16, 3.5-3.6, and 3.9-3.16. For the insecure configurations, namely Configurations 2.1-2.3, 3.1-3.3, and 3.7, Tamarin identified an attack. Thus, for the RRP to be effective, offline authorized transactions require local authentication and online authorized transactions require either local authentication or copying the **IAD-MAC** into the **IAD**. This attack should not be possible for specification-conform implementations, as the specification also states these requirements. Namely, it states that the RRP requires local authentication for offline transactions and for online transactions the RRP relies on "**IAD-MAC** combined with online card authentication" ([9], page 49), which we understand as a requirement to include the **IAD-MAC** in the **IAD**.

The attack targeting these insecure configurations is a variant of the MITM attack described in Section 4.2, targeting the relay resistance property. After this attack, the adversary and terminal share a session key for integrity $\mathbf{sk_{i}}_T = \mathrm{KDF}_c(\mathbf{z}_T)$ with which the adversary can forge the **IAD-MAC**. And as the card's **DRRE** is only authenticated by the **IAD-MAC**, the adversary can send the terminal its own **DRRE**, authenticating it with its forged **IAD-MAC**. Thus, the adversary can send the **DRRE** before the card received the terminal's **TRRE**, thereby violating the relay resistance property.

5 Conclusion

EMV has been wildly successful as a payment standard, but unfortunately it has also witnessed years of penetrate-and-patch as weaknesses have been discovered. This experience is not unique to electronic payments: testing, code review, and other forms of scrutiny, while helpful for quality assurance, ultimately fall short given a highly complex protocol whose failures can have enormous consequences. Over the past years, our experience analyzing EMV has shown that building formal models and constructing proofs is essential both in finding attacks and sharpening the assumptions needed (on the adversary or the security properties) for security proofs.

Concretely, in past work we found numerous weaknesses that lead to direct attacks on Visa and Mastercard cards that we could demonstrate in the wild. EMVCo has developed a new kernel, C8, that incorporates many of our suggested improvements. In the work reported here, we were able to construct correctness proofs for most of C8's configurations. For those configurations where we had counterexamples (i.e., attacks), we were unable to validate them in reality as cards supporting C8 have not yet been released. Overall our results here show that EMV's security has indeed been improved. They also highlight the importance of EMVCo's requirements on which configurations may be safely used, which must be carefully followed through into the implementation.

Appendix A Lemmas

In this section, we state the lemmas formalizing the security properties presented in Section 3.2.

The lemma *bank accepts* states that if the terminal accepts a transaction t at some time point i, there cannot be a time point j at which the bank declines the transaction or an agent claimed to be honest at time point i is compromised.

Lemma 2 (Bank Accepts). *A protocol P satisfies the property that the* bank accepts terminal-accepted transactions *if for every $\alpha \in traces(P)$:*

$$\begin{aligned}\forall t, i.\ \mathsf{TerminalAccepts}(t) \in \alpha_i \implies \\ \nexists j.\ \mathsf{BankDeclines}(t) \in \alpha_j \vee \\ \exists A, k.\ \mathsf{Honest}(A) \in \alpha_i \wedge \mathsf{Compromise}(A) \in \alpha_k.\end{aligned}$$

Secrecy states that a term x claimed to be secret at time point i is not known by the adversary at any time point j or an agent claimed to be honest at time point i is compromised. The adversary's knowledge of x is expressed by the action fact $\mathsf{KU}(x)$, which is part of Tamarin's built-in adversary rules. [4] By proving the

[4] Note that there are different ways to represent the adversary's knowledge in Tamarin. $\mathsf{K}(x)$ means that the adversary has sent the value as input to the protocol, whereas $\mathsf{KU}(x)$ states that the adversary has the ability to construct the value. Thus, we use $\mathsf{KU}(x)$ here to formalize that the adversary cannot construct the value x.

lemma with Tamarin we establish there is no scenario whatsoever in which the adversary can learn x, which was claimed secret, unless the compromise has happened.

Lemma 3 (Secrecy). *A protocol P satisfies* secrecy *if for every $\alpha \in traces(P)$:*

$$\begin{aligned}\forall x, i.\ \mathsf{Secret}(x) \in \alpha_i \implies \\ \nexists j.\ \mathsf{KU}(x) \in \alpha_j \vee \\ \exists A, k.\ \mathsf{Honest}(A) \in \alpha_i \wedge \mathsf{Compromise}(A) \in \alpha_k.\end{aligned}$$

The relay resistance property, adapted from [13], states that if the terminal reaches the end of a transaction with a card C, the terminal's nonce x, and the card's nonce y, at time point i, indicated by $\mathsf{CheckRelayResistance}(C, x, y) \in \alpha_i$, then the following actions were performed in the given order: first, the terminal started the RRP by sending the EXCHANGE RELAY RESISTANCE DATA with its nonce x; second, the card C received this command and responds; and third, the terminal receives the card's nonce y or an agent claimed to be honest at time point i is compromised.

Lemma 4 (Relay Resistance). *A protocol P satisfies* relay resistance *if for every $\alpha \in traces(P)$:*

$$\begin{aligned}\forall C, x, y, i. \\ \mathsf{CheckRelayResistance}(C, x, y) \in \alpha_i \implies \\ \big(\exists s, a, e.\ s < a \ \wedge\ a < e \ \wedge\ \mathsf{FastPhaseStarts}(x) \in \alpha_s \wedge \\ \mathsf{FastPhaseAction}(C) \in \alpha_a \ \wedge\ \mathsf{FastPhaseEnds}(y) \in \alpha_e\big) \vee \\ \exists A, k.\ \mathsf{Honest}(A) \in \alpha_i \wedge \mathsf{Compromise}(A) \in \alpha_k.\end{aligned}$$

The executability lemma requires a trace, where both the card and terminal and the card and bank have matching Running and Commit facts and where no agent was compromised. The Running and Commit facts are also used to express the authentication property, see Section 3.2.

Lemma 5 (Executability). *A protocol P is* executable *if $\alpha \in traces(P)$ exists such that:*

$$\begin{aligned}\exists t, C, B, nc, i, j, k, l. \\ \mathsf{Running}(C, nc, \langle \texttt{'Card'}, \texttt{'Terminal'}, t\rangle) \in \alpha_i \wedge \\ \mathsf{Commit}(nc, C, \langle \texttt{'Card'}, \texttt{'Terminal'}, t\rangle) \in \alpha_j \wedge \\ \mathsf{Running}(C, B, \langle \texttt{'Card'}, \texttt{'Bank'}, t\rangle) \in \alpha_k \wedge \\ \mathsf{Commit}(B, C, \langle \texttt{'Card'}, \texttt{'Bank'}, t\rangle) \in \alpha_l \wedge \\ \nexists A, a.\ \mathsf{Compromise}(A) \in \alpha_a.\end{aligned}$$

Appendix B Acronyms

AC	Application Cryptogram	The MAC authenticating the card to the terminal
AFL	Application File Locator	Used by the terminal to request the card's static data
AID	Application Identifier	Identifies the supported kernels
AIP	Application Interchange Profile	Informs the terminal of the card's capabilities
ATC	Application Transaction Counter	Counter used to derive the session key for the AC
CA	Certificate Authority	Issues certificates
CDCVM	Consumer Device CVM	The card (usually a smart phone) authenticates the cardholder
CDOL1	Card Risk Management Data Object List	List of data that the terminal must provide to the card
CID	Cryptogram Information Data	Encodes if and how the transaction is authorized
CVM	Cardholder Verification Method	Method of cardholder authentication
DRRE	Device Relay Resistance Entropy	Card-sourced nonce exchanged during the RRP
ECC	Elliptic Curve Cryptography	Asymmetric cryptographic technique
EDA-MAC	Enhanced Data Authentication-MAC	MAC authenticating the card to the terminal
IAD	Issuer Application Data	Contains proprietary application data
IAD-MAC	Issuer Application Data-MAC	MAC authenticating the card to the terminal
KDF	Key Derivation Function	Used to establish a key from some input
MAC	Message Authentication Code	Symmetric authentication technique
MITM	machine-in-the-middle	Also known as man-in-the-middle
NFC	Near Field Communication	Standard for wireless communication
PAN	Primary Account Number	Card number
PDOL	Processing Data Object List	List of data that the terminal must provide to the card
PIN	Personal Identification Number	Short number authenticating the card holder
RRP	Relay Resistance Protocol	Protocol protecting against relay attacks
TAC	Terminal Action Code	Configures under which conditions the terminal should take certain actions
TRRE	Terminal Relay Resistance Entropy	Terminal-sourced nonce exchanged during the RRP

References

1. Tamarin version 1.9.0, git revision: 57e619fef32033293e4a83c0be67cc6e296bf166, branch: develop
2. Apple Inc.: Tap to Pay on iPhone, https://developer.apple.com/tap-to-pay/
3. Basin, D., Cremers, C., Jannik, D., Sasse, R.: Modeling and Analyzing Security Protocols with Tamarin: A Comprehensive Guide. Information Security and Cryptography, Springer (2024), to appear
4. Basin, D., Hofmeier, X., Sasse, R., Toro-Pozo, J.: Tamarin models of C8, https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/fm24-cardpayments
5. Basin, D., Sasse, R., Toro-Pozo, J.: Card Brand Mixup Attack: Bypassing the PIN in non-Visa Cards by Using Them for Visa Transactions. In: 30th USENIX Security Symposium (USENIX Security 21). pp. 179–194. USENIX Association (Aug 2021)
6. Basin, D., Sasse, R., Toro-Pozo, J.: The EMV Standard: Break, Fix, Verify. In: 2021 IEEE Symposium on Security and Privacy (SP). pp. 1766–1781. IEEE, San Francisco, CA, USA (May 2021)
7. Basin, D., Schaller, P., Toro-Pozo, J.: Inducing Authentication Failures to Bypass Credit Card PINs. In: 32nd USENIX Security Symposium (2023). p. 15. USENIX Association (Aug 2023)
8. Coppola, D., Camurati, G., Anliker, C., Hofmeier, X., Schaller, P., Basin, D., Capkun, S.: PURE: Payments with UWB RElay-protection. In: 33rd USENIX Security Symposium (USENIX Security 2024) (2024)
9. EMVCo: EMV Contactless Specifications for Payment Systems, Book C-8, Kernel 8 Specification, Version 1.1. https://www.emvco.com/specifications (Jun 2023)
10. EMVCo: EMV Contactless Specifications for Payment Systems, Book E, Security and Key Management, Version 1.0. https://www.emvco.com/specifications/ (Jun 2023)
11. Ferro, C.: Annual Report 2023: Enhancing EMV Technologies to Supporting Emerging Payments. https://www.emvco.com/knowledge-hub/annual-report-2023-enhancing-emv-technologies-to-supporting-emerging-payments/
12. Lowe, G.: A hierarchy of authentication specifications. In: Proceedings 10th Computer Security Foundations Workshop. pp. 31–43 (Jun 1997)
13. Mauw, S., Smith, Z., Toro-Pozo, J., Trujillo-Rasua, R.: Distance-Bounding Protocols: Verification Without Time and Location. In: 2018 IEEE Symposium on Security and Privacy (SP). pp. 549–566 (May 2018)
14. Meier, S., Schmidt, B., Cremers, C., Basin, D.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: Sharygina, N., Veith, H. (eds.) Computer Aided Verification. pp. 696–701. LNCS, Springer, Berlin, Heidelberg (2013)
15. Murdoch, S.J., Drimer, S., Anderson, R., Bond, M.: Chip and PIN is Broken. In: 2010 IEEE Symposium on Security and Privacy. pp. 433–446. IEEE, Oakland, CA, USA (2010)
16. Radu, A.I., Chothia, T., Newton, C.J., Boureanu, I., Chen, L.: Practical EMV Relay Protection. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1737–1756. IEEE, San Francisco, CA, USA (May 2022)
17. Roland, M., Langer, J.: Cloning Credit Cards: A Combined Pre-play and Downgrade Attack on EMV Contactless. In: 7th USENIX Workshop on Offensive Technologies (WOOT 13) (2013)

18. Schmidt, B., Meier, S., Cremers, C., Basin, D.: Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In: 2012 IEEE 25th Computer Security Foundations Symposium. pp. 78–94. IEEE, Cambridge, MA, USA (Jun 2012)