The EMV Standard: Break, Fix, Verify

David Basin, Ralf Sasse, and Jorge Toro-Pozo
Department of Computer Science, ETH Zurich
{basin, ralf.sasse, jorge.toro}@inf.ethz.ch

***Abstract*—EMV is the international protocol standard for smartcard payment and is used in over 9 billion 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, running over 2,000 pages.**

We formalize a comprehensive symbolic model of EMV in Tamarin, a state-of-the-art protocol verifier. Our model is the first that supports a fine-grained analysis of all relevant security guarantees that EMV is intended to offer. We use our model to automatically identify flaws that lead to two critical attacks: one that defrauds the cardholder and another that defrauds the merchant. First, criminals can use a victim's Visa contactless card for high-value purchases, without knowledge of the card's PIN. We built a proof-of-concept Android application and successfully demonstrated this attack on real-world payment terminals. Second, criminals can trick the terminal into accepting an unauthentic offline transaction, which the issuing bank should later decline, after the criminal has walked away with the goods. This attack is possible for implementations following the standard, although we did not test it on actual terminals for ethical reasons. Finally, we propose and verify improvements to the standard that prevent these attacks, as well as any other attacks that violate the considered security properties. The proposed improvements can be easily implemented in the terminals and do not affect the cards in circulation.

***Index Terms*—EMV; payment security; credit card fraud; Visa; PIN bypass; authentication; formal analysis**

I. Introduction

EMV, named after its founders Europay, Mastercard, and Visa, is the worldwide standard for smartcard payment, developed in the mid 1990s. As of December 2019, more than 80% of all card-present transactions globally use EMV, reaching up to 98% in many European countries. Banks have a strong incentive to adopt EMV due to the *liability shift*, which relieves banks using the standard from any liability from payment disputes. If the disputed transaction was authorized by a PIN then the consumer (EMV terminology for the payment-card customer) is held liable. If a paper signature was used instead, then the merchant is charged.

EMV: 20 Years of Vulnerabilities

Besides the liability shift, EMV's global acceptance is also attributed to its advertised security. However, EMV's security has been challenged numerous times. Man-in-the-middle (MITM) attacks [1], card cloning [2], [3], downgrade attacks [3], relay attacks [4]–[7], and card skimming [8], [9] are all examples of successful exploits of the standard's shortcomings. The MITM attack reported by Murdoch *et al.* [1] is believed to have been used by criminals in 2010–11 in France and Belgium to carry out fraudulent transactions for ca. 600,000 Euros [10]. The underlying flaw of Murdoch *et al.*'s attack is that the card's response to the terminal's offline PIN verification request is not authenticated.

Some of the security issues identified result from flawed implementations of the standard. Others stem from logical flaws whose repairs would require changes to the entire EMV infrastructure. Identifying such flaws is far from trivial due to the complexity of EMV's execution flow, which is highly flexible in terms of card authentication modes, cardholder verification methods, and online/offline authorizations. This raises the question of how we can systematically explore all possible flows and improve the standard to avoid another twenty years of attacks.

Approach Taken: Break, Fix, Verify

In this paper we focus on weakness of and improvements to the EMV protocol design. We present a formal, comprehensive model for the symbolic analysis of EMV's security. Our model is written in Tamarin [11], [12], a state-of-the-art verification tool that has been used to study numerous real-world protocols, including TLS 1.3 [13] and 5G authentication [14]. Tamarin supports protocol verification in the presence of powerful adversaries and unboundedly many concurrent protocol sessions.

Our model supports the analysis of all properties that must hold in any EMV transaction. An informal description of the three most relevant properties is as follows:

1) *Bank accepts terminal-accepted transactions*: No transaction accepted by the terminal can be declined by the bank.
2) *Authentication to the terminal*: All transactions accepted by the terminal are authenticated by the card and, if authorized online, the bank.
3) *Authentication to the bank*: All transactions accepted by the bank are authenticated by the card and the terminal.

Our model faithfully considers the three roles present in an EMV session: the bank, the terminal, and the card. Previous symbolic models merge the terminal and the bank into a single agent [15]–[17]. This merging incorrectly entails that the terminal can verify all card-produced cryptographic proofs that the bank can. This is incorrect as the card and the bank share a symmetric key that is only known to them.

Using our model, we identify a critical violation of authentication properties by the Visa contactless protocol: the cardholder verification method used in a transaction, if any, is neither authenticated nor cryptographically protected against

modification. We developed a proof-of-concept Android application that exploits this to **bypass PIN verification** by mounting a man-in-the-middle attack that instructs the terminal that PIN verification is not required because the cardholder verification was performed on the consumer's device (e.g., a mobile phone). This enables criminals to use any stolen Visa card to pay for expensive goods without the card's PIN. In other words, *the PIN is useless in Visa contactless transactions!*

We have successfully tested our PIN bypass attack on real-world terminals for a number of transactions with Visa-branded cards such as Visa Credit, Visa Electron, and V Pay cards. For example, we performed a transaction of ca. $190 in an attended terminal in an actual store. As it is now common for consumers to pay with their smartphones, the cashier cannot distinguish the attacker's actions from those of any legitimate cardholder. For ethical reasons, we carried out all our tests using our own credit/debit cards. However, we stress that the attack works for any Visa card that the attacker possesses, in particular with stolen cards.

Our symbolic analysis also reveals that, in an offline contactless transaction with a Visa or an old Mastercard card, the card does not authenticate to the terminal the Application Cryptogram (AC), which is a card-produced cryptographic proof of the transaction that the terminal *cannot* verify (only the card issuer can). This enables criminals to **trick the terminal into accepting an unauthentic offline transaction**. Later on, when the acquirer submits the transaction data as part of the clearing record, the issuing bank will detect the wrong cryptogram, but the criminal is already long gone with the goods. We did not test this attack on actual terminals for ethical reasons as this would defraud the merchant.

Contributions

First, we present a comprehensive symbolic model of the EMV standard that accounts for the 3 Offline Data Authentication methods (SDA, DDA, and CDA), the 5 Cardholder Verification Methods (no PIN, plaintext PIN, offline enciphered PIN, online PIN, and CDCVM), the 2 types of Transaction Authorizations (offline and online), and the 2 (major) types of contactless transactions (Visa and Mastercard). Our model considers the three roles present in a transaction, and supports the fine-grained analysis of all relevant security properties.

Second, we identify and demonstrate, for the first time in actual terminals, a practical attack that allows attackers to make fraudulent, high-value purchases, without knowledge of the card's PIN. We also identify an attack that allows one to effectively steal goods by tricking terminals into accepting unauthentic offline transactions. Our attacks demonstrate that EMV's liability shift should be voided because credit card fraud is not necessarily the result of negligent behavior of consumers or merchants.

Finally, based on our full-scale, automatic, Tamarin-supported analysis of EMV's fundamental security properties, we identify the EMV configurations that guarantee secure transactions. Based on these configurations, we propose solutions that can be implemented in the payment terminals and rule out security breaches.

Note that our focus is on EMV's design, not implementations themselves. In this way, we can end the penetrate-and-patch arms race where attackers continually find and exploit protocol weaknesses. Of course this is only one part of the overall picture, as attackers can still exploit implementation weaknesses; but it is a substantial part and it is also a prerequisite for any "full stack" effort to formally develop a verified protocol, down to the level of code

Organization

In Section II we describe related work, focusing on previous EMV security analyses. In Section III we provide background on the EMV protocol. In Section IV we present our formal model of EMV, focusing on how we model EMV's numerous configurations and how we define and analyze its security properties. In Section V we present the results of this analysis. Later, in Section VI, we describe an Android app that we developed and used to show that our Tamarin findings can be turned into real-world attacks. As a result of our analysis, we suggest improvements to terminals that guarantee secure transactions. We draw conclusions in Section VII.

Ethical Considerations

We carried out all our tests using our own credit/debit cards. Furthermore, we have notified Visa of the attacks discovered.

II. Related Work

Given its financial importance, it is not surprising that the EMV standard has been extensively studied. We review here the most relevant related work. This previous work concerns either implementation flaws, or protocol flaws discovered by analysis of selected and possibly simplified parts of the EMV specification. In contrast, our analysis integrates all the different configurations for card authentication, cardholder verification, and transaction authorization in a single symbolic model. This provides a basis not only for discovering all relevant design errors, but also producing correctness proofs.

In 2010, Murdoch *et al.* [1] identified a serious flaw in EMV's offline Cardholder Verification Methods (CVMs). Namely, the card's response to the terminal's PIN verification request is not authenticated. Therefore, a man-in-the-middle (MITM) could reply with the *success* message to *any* PIN the terminal would request verification for. The dummy PIN could be blocked from reaching the card, which would then assume that either the chosen CVM was paper signature or no CVM was required at all. All subsequent steps would be carried out normally and the transaction would be accepted.

Even though Murdoch *et al.*'s attack comes with some engineering challenges, such as miniaturizing the MITM infrastructure, these challenges appear to have been overcome as observed in the aforementioned forgery of credit cards in France and Belgium [10]. Our analysis demonstrates that this attack still exists in old cards that support neither asymmetric

cryptography nor online PIN verification (see Section V-A). Unfortunately, many modern cards that support both features are still vulnerable to our own PIN bypass attack, which we present in this paper.

Soon after, De Ruiter and Poll [15] gave a ProVerif [18] model of a variant of the EMV contact protocol. They summarize over 700 pages of EMV specifications into 370 lines of F# code, which they transform into the ProVerif language using the FS2PV tool [19]. Their analysis does not identify the attack of [1] because the terminal's selection of the CVM is over-simplified to always opt for the offline plaintext PIN. This makes the card always expect a PIN verification request, with the correct PIN, before continuing with the transaction.

Some of EMV's flaws have also been identified from empirical studies in the field [3], [8], [9]. For example, UK researchers, together with unsatisfied consumers who were denied refunds for fraud claims, were given access to the bank logs of the disputed transactions. This, together with reverse-engineering some ATMs, revealed flawed implementations of EMV. They noted that the supposedly unpredictable numbers generated by some terminals were actually pretty predictable, allowing criminals to *pre-play* payments and use the retrieved data for later purchases [8].

Bond *et al.* [8] also observed that a pre-play attack is still possible even if the terminal's random number generator works correctly. In this case, the pre-play consists of the attacker replacing the terminal-generated nonce with one used in an earlier transaction between the attacker and the victim's card.

Symbolic models consider the Dolev-Yao threat model [20], where the adversary only knows public knowledge, the data sent over the network, and the outcome of public functions on known input. The adversary is also an active attacker, who can modify, block, and inject data on the network. In this model, however, random number generators are assumed to be sound, i.e., random numbers cannot be predicted. Therefore, attacks of this kind are usually not part of a symbolic analysis that examines the specification (not the implementation) for logical errors. Our analysis thus does not uncover Bond *et al.*'s attacks [8]. Note that it is possible though to incorporate weak random generators and compromised channels into symbolic models, as described in [21].

The EMV contactless protocol's security has been challenged multiple times too. For example, Roland and Langer [3] detected a downgrade attack that exploits Mastercard's *MagStripe* mode, a legacy authentication mode kept for backward compatibility. They showed that a mobile phone supporting Near Field Communication (NFC) can collect all authentication codes that a card could produce in response to all potential challenges from a terminal. Hence, a clone card pre-loaded with the codes can be used for fraudulent payments. This attack is feasible because the MagStripe mode reduces the terminal's pool of unpredictable numbers to 1000 values only. In this paper we do not consider the MagStripe mode because the random generators are assumed sound (as explained above) and this mode has been deprecated in many countries worldwide.

Other attacks demonstrated against EMV contactless payment protocols are well-known *relay attacks* [4], [6], [7]. The works [4], [7] suggest using distance bounding protocols [22], [23] as a countermeasure to such attacks. Although distance bounding does prevent relay attacks, only Mastercard seems to be inclined to use it. Relay attacks are usually ignored because they are presumably feasible only for small transactions, since larger transactions require cardholder verification.

In 2014, Emms *et al.* [9] observed that some UK-issued contactless Visa credit cards drop the PIN verification for transactions in foreign currencies. The authors developed a proof-of-concept implementation of the attack, where they faked a transaction of almost one million US dollars. We reproduced the experiments of [9] but all modern cards we tested did ask for PIN verification for large-value transactions in both domestic and foreign currencies.

There exist various symbolic models that showcase the EMV contactless protocols [16], [17], [24]–[26]. All of these focus on verifying proximity between the card and the terminal. They also consider the terminal and the bank as a single agent and consequently do not observe the pre-play attack [3].

Galloway and Yunusov [27] recently presented a man-in-the-middle attack that also circumvents Visa's PIN verification. Their attack is similar to ours in that it modifies a card-sourced message that instructs the terminal that cardholder verification was performed on the consumer's device. In contrast to our attack, Galloway and Yunusov's attack also modifies a terminal-sourced message in which the cardholder verification request is encoded. According to EMV's (generic) cryptogram definition, such message should be protected against modification. Their attack works because Visa's proprietary cryptograms do not prevent such modification, or at least not the ones implemented by the cards they tested. Interestingly, and worrisome, our own attack demonstrates that the *strongest* cryptogram proposed by EMV still does not suffice to correctly verify the cardholder. Details are given in Section VI.

III. EMV Description

The EMV specification runs over 2,000 pages split across several books. Moreover, many of the statements in these books are quite complex and cross-reference other books. In this section we give a detailed description of the standard. Given its complexity, creating this specification and its formal model in Tamarin was a major undertaking that took over six months of full-time work. Our methodology included not only carefully reading the standard, but also cross-checking and disambiguating its statements with data from over 30 real-world transaction logs that we obtained using the Android app we developed, described in later sections.

An EMV transaction consists of a series of Application Protocol Data Unit (APDU) command/response exchanges and can be divided into four phases:

1) *Initialization*: the card and the terminal agree on the application to be used for the transaction and exchange static data such as the card's records containing information

about the card and the issuing bank (or simply the bank from now on, unless otherwise specified).
2) *Offline Data Authentication (ODA)*: the terminal performs a PKI-based validation of the card. Once the card has provided the terminal with the Certificate Authority (CA) index, the CA-issued bank's PK certificate, and the bank-issued card's PK certificate, the terminal validates the card's signature on the transaction details.
3) *Cardholder Verification*: the terminal determines whether the person presenting the card is the legitimate cardholder. This is done using a method that the card and the terminal both support. The most common method is online enciphered PIN verification, in which the terminal sends (an encryption of) the entered PIN to the bank for verification. The card is not involved.
4) *Transaction Authorization (TA)*: the transaction is declined offline, accepted offline, or sent to the issuing bank for online authorization.

An overview of the full EMV transaction flow is depicted in Figure 1 and the details of each phase are given next.

A. Initialization

The first step of an EMV transaction is the application selection. The terminal issues the SELECT command with the string 1PAY.SYS.DDF01 (in bytes), which refers to the contact Payment System Environment (PSE), or 2PAY.SYS.DDF01 for contactless. The card responds with the sequence of Application Identifiers (AIDs). In the response, the card may also request the Processing Data Object List (PDOL), which is a list of terminal-sourced transaction data. The PDOL typically includes the amount, the country code, the currency, the date, the transaction type, and the terminal's random number UN (called Unpredictable Number in EMV's terminology).

The terminal issues the GET PROCESSING OPTIONS command along with the PDOL data, if requested by the card. The card responds with the 2-byte Application Interchange Profile (AIP, which indicates the supported authentication methods and whether cardholder verification is supported) and the Application File Locator (AFL, which points to a list of files and records that the terminal should read from the card). The terminal then learns these records using the READ RECORD command. The records typically include:

- the Primary Account Number (PAN, commonly known as the card number), the card's expiration date, and other static data;
- the index of the used CA, the CA-issued certificate of the bank's PK, and the bank-issued card's PK certificate, if the card supports asymmetric encryption;
- the first and second Card Risk Management Data Object Lists (CDOL1 and CDOL2, respectively), which typically include the PDOL and further transaction data; and
- the list of the supported CVMs.

From the CA's index, the terminal retrieves the CA's PK from an internal data base and then verifies the bank's certificate. Afterwards, from the bank's certificate, the terminal acquires the bank's PK and verifies the card's certificate, if applicable. Finally, the terminal acquires the card's PK from the card's certificate.

B. Offline Data Authentication

For Offline Data Authentication (ODA), also known as Card Authentication, there exist three methods:

1) *Static Data Authentication (SDA)*: the card supplies the terminal with the Signed Static Authentication Data (SSAD), which is the bank's signature on the card's static data such as the PAN, the card's expiration date, and optionally the AIP. The SDA method prevents modification of the card's static data, but it does not prevent cloning.
2) *Dynamic Data Authentication (DDA)*: the terminal transmits the INTERNAL AUTHENTICATE command whose payload is the Dynamic Data Object List (DDOL). The DDOL must contain the terminal's Unpredictable Number. In response to this challenge, the card transmits the Signed Dynamic Authentication Data (SDAD), which is the card's signature on the card's dynamic data (a fresh number NC) and the received DDOL. The DDA method protects against modification of card data and cloning.
3) *Combined Dynamic Data Authentication (CDA)*: this is similar to DDA but it includes the transaction details in the SDAD, e.g., the transaction amount.

C. Cardholder Verification

A Cardholder Verification Method (CVM) can be paper signature, PIN verification, Consumer Device CVM (CDCVM), or a combination of these. For PIN verification, there are three specific methods:

1) *Offline Plaintext PIN* (or simply *plain PIN*): the terminal sends the VERIFY command along with the entered PIN and the card responds with the success message `9000` if the PIN is correct, or the failure message `63C`x, where the digit x is the number of tries left. When no tries remain, i.e., $x = 0$, then the card must respond with the PIN-blocked message `6983` to any subsequent VERIFY requests.
2) *Offline Enciphered PIN* (or simply *enciphered PIN*): the terminal sends the GET CHALLENGE command and the card responds with a random number. Then the terminal issues the VERIFY command whose payload is an encryption, with the card's PK, of the entered PIN, and the received random number, and random padding generated by the terminal. Upon reception, the card decrypts the payload and responds accordingly, using the messages described in the plain PIN method.
3) *Online Enciphered PIN* (or simply *online PIN*): the card is not involved. Instead, the terminal sends the entered PIN encrypted to the issuing bank, when requesting the transaction authorization.

The Consumer Device CVM is intended to be performed by devices such as mobile phones, which authenticate the cardholder through fingerprint or face recognition. How the terminal and the device conduct CDCVM is out of EMV's

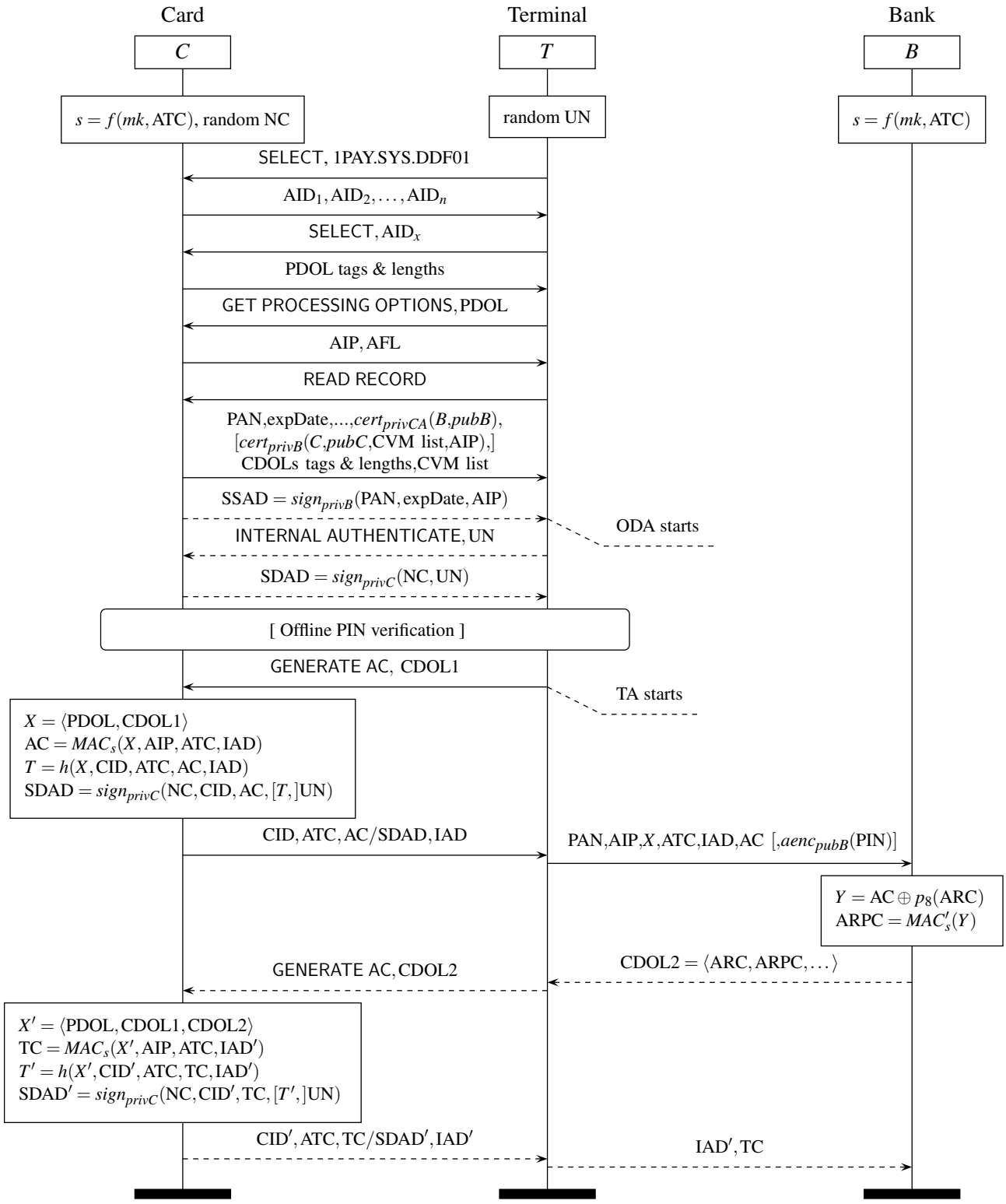


Fig. 1. An overview of the EMV transaction. Dashed messages and bracketed terms are either optional, or depend on previous steps, or depend on the parties' choices. For simplicity, this chart only shows the execution flow in which the card responses have the success trailer 9000. Notation: $\oplus$ is exclusive-OR; f is a one-way function; $(privC, pubC)$, $(privB, pubB)$, and $(privCA, pubCA)$ are the PKI pairs of the card, the bank, and the CA, respectively; $cert_k(cont)$ is the PKI certificate on $cont$ signed with the private key k; $sign_k(m)$ is the signature on m with the key k; $aenc_k(m)$ is the asymmetric encryption of m with the key k; $MAC_k(m)$ and $MAC'_k(m)$ are MACs on m with the key k; $p_b(m)$ is the right-padding of m with b zero bytes.

scope. Nevertheless, this method is the fundamental cause of one of the new attacks that we report on in this paper.

D. Transaction Authorization

The terminal can decide either to decline the transaction offline, to authorize the transaction offline, or to request online authorization from the issuing bank. This decision is made based on various checks such as the offline ceiling limit, above which transactions should be processed online.

The terminal sends the GENERATE AC command to the card, along with the CDOL1. This command instructs the card to supply the 8-byte Application Cryptogram (AC) which is:

- a Transaction Cryptogram (TC), if the terminal decided for offline approval,
- an Authorization Request Cryptogram (ARQC), if the terminal decided for online authorization, or
- an Application Authentication Cryptogram (AAC), if the terminal decided to decline the transaction.

The requested type of cryptogram is encoded in the command payload. The card then issues the AC whose type can be either the requested one, or an ARQC (which forces the transaction to go online), or an AAC (which indicates the transaction is declined). The card cannot generate a TC when an ARQC was requested.

The cryptogram is a MAC computed over the transaction details, the AIP, and the Application Transaction Counter (ATC), which is a 2-byte counter incremented on every transaction. The key for this MAC is a session key s derived from the ATC and a symmetric master key mk shared by the bank and the card. Along with the cryptogram itself, the card sends other data such as the 1-byte Cryptogram Information Data (CID), which indicates the type of cryptogram being sent; the transaction counter ATC, and if CDA was requested in the command payload, the Signed Dynamic Authentication Data (SDAD). In this case, the SDAD is a signature on the card's random number NC, the CID, the cryptogram, a hash of the transaction details, and the terminal's UN.

If the card responds with a TC and the chosen CVM was not online PIN, then the transaction is approved and the TC serves as a settlement to instruct the bank to transfer the funds to the merchant's account.

If the transaction must be authorized online, then the terminal forwards to the bank the transaction details, the ARQC, and if online PIN verification was the selected CVM, the entered PIN. The bank authorizes or declines the transaction by sending back to the terminal the 2-byte Authorization Response Code (ARC, authorize/decline and further data) and the Authorization Response Cryptogram (ARPC). The latter is a MAC generated over the exclusive-OR of the ARC (padded to 8 bytes) and the received cryptogram ARQC, using the session key s. The terminal then issues the EXTERNAL AUTHENTICATE command (or equivalently a second GENERATE AC) to inform the card of the bank's decision. The card constructs the response analogously to its response to the (first) GENERATE AC command, only this time no ARQC is sent, but instead either a TC or an AAC.

IV. Modeling and Analysis Methodology

To model and analyze the EMV standard, we use the protocol verification tool Tamarin [11], [12]. Tamarin is a state-of-the-art model-checker for security protocol verification. It features an expressive language for specifying protocols, their properties, and adversaries, as well as powerful inference procedures for automating much of protocol verification. We first provide some background on Tamarin and then present the properties we analyze and our analysis methodology.

A. Tamarin Background

In Tamarin's underlying theory, cryptographic messages are terms in an order-sorted term algebra $(\mathcal{S}, \leq, \mathcal{T}_\Sigma(\mathcal{V}))$ where $\mathcal{S}$ is a set of sorts, $\leq$ a partial order on $\mathcal{S}$, Σ is a signature, and $\mathcal{V}$ is a countably infinite set of variables. For example, the term $pk(k)$, with $pk \in \Sigma$, denotes the public key associated to the private key $k \in \mathcal{T}_\Sigma(\mathcal{V})$. Similarly, the term $aenc_k(m)$, with $aenc \in \Sigma$, denotes the asymmetric encryption of the message $m \in \mathcal{T}_\Sigma(\mathcal{V})$ with the public key $k \in \mathcal{T}_\Sigma(\mathcal{V})$. The algebraic properties of the cryptographic functions are defined by equations over terms. For example, $adec_k(aenc_{pk(k)}(m)) = m$ specifies the semantics of asymmetric decryption.

Tamarin models a protocol's set of executions as a labeled transition system (LTS). The states of the LTS are multisets of *facts*, which formalize the local states of the agents running the protocol, the adversary's knowledge, and messages on the network. Facts are of the form $\mathsf{F}(a_1, a_2 \ldots, a_n)$ where F is a symbol from an unsorted signature Γ of predicate symbols and $a_i \in \mathcal{T}_\Sigma(\mathcal{V})$.

Transitions between states are determined by *transition rules* (or simply rules). A rule is a triple (l, a, r), also written as $[l]\text{–}[a]\rightarrow[r]$, where l, a, and r are multisets of facts. For example, the following rule specifies the transmission of the hash of a received message:

$$[\mathsf{In}(m)]\text{–}[\mathsf{SentHash}(A, m)]\rightarrow[\mathsf{State1}(A, m), \mathsf{Out}(h(m))]\,.$$

This rule states that, if there is a term m input on the network, then update the local state of A to $\mathsf{State1}(A, m)$, remove m from the network, and output the term $h(m)$ on the network, possibly for reception by A's communication partner. The transition is labeled with $\mathsf{SentHash}(A, m)$, meaning that A sent the hash of m.

In what follows, let $\mathcal{F}$ be the universe of facts and $\mathcal{R}$ the universe of rules. Whereas $\mathcal{P}(.)$ denotes the power set of a set, we use $\mathcal{M}(.)$ to refer to the power *multiset* of a set. We define the function $linear\colon \mathcal{M}(\mathcal{F}) \rightarrow \mathcal{M}(\mathcal{F})$ that yields all *linear* facts from the input multiset of facts. Linear facts annotate resources that can be consumed just once, such as messages on the network. Facts that are not linear are called *persistent* and can be reused arbitrarily often without being consumed. We also define the function $gins\colon \mathcal{P}(\mathcal{R}) \rightarrow \mathcal{P}(\mathcal{R})$ that yields the set of all *ground instances* of the input set of rules. A ground instance of a rule is the rule resulting from the substitution of all variables with *ground terms* (i.e., terms from $\mathcal{T}_\Sigma$). Also, let $\mathcal{A} \subseteq \mathcal{R}$ be the set of global rules modeling

a network controlled by a Dolev-Yao adversary [20] as well as the generation of random, fresh values.

A protocol $P \subseteq \mathcal{R}$ is a set of rules. The associated LTS is $(S, \Lambda, \rightarrow)$, where $S = \mathcal{M}(\mathcal{F})$, $\Lambda = \mathcal{M}(\mathcal{F})$, and $\rightarrow \subseteq S \times \Lambda \times S$ is defined by:

$$s \xrightarrow{a} s' \iff \exists (l, a, r) \in gins(P \cup \mathcal{A}).\ l \subseteq s \wedge s' = (s \setminus linear(l)) \cup r.$$

A transition consumes the linear facts of l from the current state, adds the facts from r, and labels the transition with a. An execution of P is a finite sequence $(s_0, a_1, s_1, \ldots, a_n, s_n)$ such that $s_0 = \emptyset$ and $s_{i-1} \xrightarrow{a_i} s_i$ for all $1 \leq i \leq n$. The sequence $(a_1, \ldots, a_n)$ is a *trace* of P and the set of all of P's traces is denoted $traces(P)$. Security properties are specified using first-order logic formulas on traces. Further details on Tamarin's syntax and semantics can be found in [11], [12].

B. Security Properties

As we have seen, EMV involves three parties: the consumer's card, the merchant's terminal, and the cardholder's bank. Its central security properties concern the parties authenticating each other, guarantees on transaction information, and the secrecy of sensitive data. We formalize these properties next.

The first property we examine is that no terminal-accepted transaction will be declined by the bank. This property is particularly relevant for offline-capable terminals, which typically do not request online authorization for low-value transactions. Such terminals can be cheated if the property fails.

Definition 1 (Bank accepts)**.** *A protocol P satisfies the property that the* bank accepts terminal-accepted transactions *if for every $\alpha \in traces(P)$:*

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

In our model, the $\mathsf{TerminalAccepts}(t)$ fact is added to the trace only if the terminal is satisfied with the transaction t and the associated cryptographic proofs provided by the card. That is, when the terminal issues a purchase receipt. The $\mathsf{BankDeclines}(t)$ fact is produced when the bank receives an authorization request for the transaction with a wrong Application Cryptogram. The last line rules out transactions where an agent, presumed honest, has been compromised. For example, a bank that maliciously rejects a correct transaction should not make the property fail.

Our second property corresponds to the authentication property commonly known as *injective agreement* [28], [29].

Definition 2 (Authentication to terminal)**.** *A protocol P satisfies* authentication to the terminal *if for every $\alpha \in traces(P)$:*

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

The above property, with $\texttt{'Terminal'} \in \mathcal{T}_\Sigma$ and $\langle\rangle \in \Sigma$, states that whenever the terminal T *commits* to a transaction t with its communication partner P, then either P, in role $r \in \{\texttt{'Card'}, \texttt{'Bank'}\} \subseteq \mathcal{T}_\Sigma$, was *running* the protocol with T and they agree on t, or an agent, presumed honest, has been compromised. Additionally, there is a unique Commit fact for each pair of accepted transaction and accepting agent, which means that replay attacks are prevented.

The facts Commit and Running, introduced in [28], are used to specify authentication properties. A Commit fact represents an agent's belief about its communication partner's local state, whereas Running represents the partner's actual state. Authentication properties are therefore expressed in terms of matching pairs of such facts. In our models, Commit facts occur whenever the committing agent is in a satisfactory state when the transaction is ready to be accepted.

Our third property is also an authentication property and is very similar to the second, except that the agent who commits is the bank. That is, the definition is the same except the ground term $\texttt{'Terminal'}$ is now $\texttt{'Bank'} \in \mathcal{T}_\Sigma$.

Another property relevant for formal protocol analysis is *secrecy* (a.k.a. confidentiality). The secrecy of a term x holds when x is not known to the attacker. The attacker's knowledge of a term x is written as $\mathsf{KU}(x)$, where $\mathsf{KU} \in \Gamma$ is a fact symbol defined by Tamarin's built-in rules that model how the attacker acquires knowledge. The definition of secrecy also assumes that the agents involved are not compromised.

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

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

In an EMV transaction, terms that should be secret include the PIN number, the PAN (i.e., the card number), and the keys (i.e., private keys and symmetric shared keys).

We also consider other properties such as *executability*, which allows one to assess whether a protocol execution reaches a state where the bank and the terminal have accepted a transaction and no compromises have occurred. This represents a sanity check showing that the protocol modeled behaves as expected and allows executions of protocol runs without adversary involvement. This ensures that there are no modeling errors that would make the specified protocol inoperable and lead to false results.

Definition 4 (Executability)**.** *A protocol P is* executable *if $\alpha \in traces(P)$ exists such that:*

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

C. Analysis Methodology

We construct our model in a way that accounts for all possible protocol flows and interactions, but gives us a structured analysis of which kinds of executions are vulnerable to attacks. We start by formalizing the EMV standard in two *generic* models:

1) one for the **EMV contact protocol**, modeling the full execution space of a contact transaction, and
2) one for the **EMV contactless protocol**, modeling the full execution space of a Mastercard [30] or Visa [31] contactless transaction.

Each of these two models captures all possible executions of the corresponding Payment System Environment (contact or contactless), including simultaneous transactions with different cards, terminals, types of authentication, cardholder verification methods, and all the other settings. For example, the contactless protocol model allows for executions between a terminal, which believes to be in a Visa transaction, and three cards, which may be different from Visa cards. Clearly, whether the system can reach a state where the transaction is accepted depends on the actual messages and cryptographic proofs that the terminal and the bank receive.

Tamarin exhibits a property violation by constructing a trace that contradicts the given property. Clearly, Tamarin cannot output all such traces as there are infinitely many (simply by adding unrelated steps), if one exists. Running Tamarin on the generic models will therefore either lead to a successful verification or one attack trace, violating the property, with the "least secure" type of card and authentication method, among other settings. However, one might be interested, for example, in the property of authentication to the bank specifically for transactions where the card used Combined Dynamic Data Authentication (CDA, recall from Section III-B) and the transaction value was high, i.e., above the CVM-required limit.

With this in mind, we employed a modeling strategy that automatically generates specific Tamarin models from the two generic models. To automatically generate the specific models, we use *target configurations*. A target configuration is a choice of arguments that selects the transactions for which we want to verify the security properties. A generic model and a target configuration determine what we call a *target model*. For example, Visa_DDA_Low is a target model generated from the contactless protocol (generic) model with the target arguments:

- DDA: referring to the offline data authentication method (known as *fast* DDA in [31]), and
- Low: indicating a low-value transaction.

We automated the generation of target models and the interested reader can find the technical details in Appendix A as well as in our Tamarin theories and their README [32].

In our models, we consider the following transaction data to be agreed upon for the authentication properties (i.e., the term t in the definitions of Section IV):

- the Primary Account Number (PAN);
- the Application Interchange Profile (AIP);
- the Cardholder Verification Method (CVM) used;
- the Application Transaction Counter (ATC);
- the Application Cryptogram (AC) data input (X and X' in Figure 1);
- the Application Cryptogram (AC) itself; and
- the Issuer Application Data (IAD).

For both the contact and contactless models, between the terminal and the card (and vice versa) we modeled a channel controlled by the Dolev-Yao adversary, who can listen, block, inject, and modify the transmitted data. Between the bank and the terminal (and vice versa) we modeled a secure channel that offers authentication and secrecy.

We assumed that physical Mastercard cards have the second bit of AIP's first byte cleared. This bit describes whether the Consumer Device CVM is supported. This assumption is reasonable since only "virtual" cards (i.e., cards registered in mobile apps such as ApplePay or Google Pay) have this bit set. Note that this does not mean that an adversary cannot try to set this bit. Visa's AIPs are transaction-dependent and the selection of the Consumer Device CVM is not determined by the AIP, but by the Card Transaction Qualifiers (CTQ).

We also assumed that terminals do not complete high-value, contactless transactions with cards that (apparently) do not support cardholder verification. In such transactions, the common practice of terminals is to reject the attempt and instruct the cardholder to switch to the contact interface.

V. Analysis Results

We conducted a full-scale, automated security analysis of 40 configurations of EMV, including both types of transactions: contact and contactless. We describe the results of this comprehensive analysis in this section and afterwards show how some of these flaws can be exploited in practical attacks.

A. Analysis Results for the EMV Contact Protocol

Our analysis results for the 24 configurations of the EMV contact protocol are summarized in Table I. Although there are no major surprises here, the results illustrate the benefits of a comprehensive formalization and analysis. In particular, we both rediscovered existing, known attacks on the contact protocols as well as attacks that, to our knowledge are new, but relatively difficult to carry out in practice and therefore have limited practical relevance. Note that we have omitted the results for secrecy from the table because they are identical for all models. All of our models and proofs are available at [32].

Our analysis revealed disagreement, both between the terminal and the card and between the bank and the card, on the

TABLE I

ANALYSIS RESULTS FOR THE EMV CONTACT PROTOCOL. ALL TARGET MODELS HAVE 55 RULES. THE LAST TWO COLUMNS INDICATE, IN THAT ORDER, THE NUMBER OF LINES OF TAMARIN CODE THAT THE MODEL COMPRISES, AND THE TIME TAKEN FOR OUR TAMARIN ANALYSIS, USING 10 THREADS AND AT MOST 20GB OF RAM PER MODEL, ON A COMPUTING SERVER RUNNING UBUNTU 16.04.3 WITH TWO INTEL(R) XEON(R) E5-2650 V4 @ 2.20GHZ CPUS (WITH 12 CORES EACH) AND 256GB OF RAM. THE MODEL(S) FOR WHICH ALL FOUR PROPERTIES WERE VERIFIED ARE HIGHLIGHTED IN BOLD.

No.	Target model	Properties				LoC	Time
		executable	bank accepts	auth. to terminal	auth. to bank		
1	Contact_SDA_PlainPIN_Online	✓	×(2)	×(1,2)	×(1)	758	13m07s
2	Contact_SDA_PlainPIN_Offline	✓	×(2)	×(1,2)	×(1)	761	11m39s
3	Contact_SDA_OnlinePIN_Online	✓	×(2)	×(1,2)	×(1)	758	13m02s
4	Contact_SDA_OnlinePIN_Offline	–	–	–	–	731	11m48s
5	Contact_SDA_NoPIN_Online	✓	×(2)	×(1,2)	×(1)	752	8m21s
6	Contact_SDA_NoPIN_Offline	✓	×(2)	×(1,2)	×(1)	755	6m37s
7	Contact_SDA_EncPIN_Online	–	–	–	–	758	12m21s
8	Contact_SDA_EncPIN_Offline	–	–	–	–	761	11m36s
9	Contact_DDA_PlainPIN_Online	✓	×(2)	×(1,2)	×(1)	766	13m48s
10	Contact_DDA_PlainPIN_Offline	✓	×(2)	×(1,2)	×(1)	769	12m20s
11	Contact_DDA_OnlinePIN_Online	✓	×(2)	×(2)	✓	775	16m04s
12	Contact_DDA_OnlinePIN_Offline	–	–	–	–	739	12m27s
13	Contact_DDA_NoPIN_Online	✓	×(2)	×(2)	✓	769	12m15s
14	Contact_DDA_NoPIN_Offline	✓	×(2)	×(2)	✓	772	8m43s
15	Contact_DDA_EncPIN_Online	✓	×(2)	×(1,2)	×(1)	766	14m07s
16	Contact_DDA_EncPIN_Offline	✓	×(2)	×(1,2)	×(1)	769	12m59s
17	Contact_CDA_PlainPIN_Online	✓	✓	×(1)	×(1)	763	1h55m31s
18	Contact_CDA_PlainPIN_Offline	✓	✓	×(1)	×(1)	766	14m10s
19	**Contact_CDA_OnlinePIN_Online**	✓	✓	✓	✓	781	6h03m05s
20	Contact_CDA_OnlinePIN_Offline	–	–	–	–	739	12m15s
21	**Contact_CDA_NoPIN_Online**	✓	✓	✓	✓	775	2h31m23s
22	**Contact_CDA_NoPIN_Offline**	✓	✓	✓	✓	778	12m16s
23	Contact_CDA_EncPIN_Online	✓	✓	×(1)	×(1)	763	1h59m44s
24	Contact_CDA_EncPIN_Offline	✓	✓	×(1)	×(1)	766	14m00s

Legend:

✓: property verified ×: property falsified –: not applicable

(1): disagrees with the card on the CVM used (2): disagrees with the card on the last AC

selected CVM for transactions using SDA or offline (plain or enciphered) PIN verification (Table I, Remark 1).

For transactions where the terminal performed offline PIN verification, our analysis identifies a trace that represents the PIN bypass attack first observed by Murdoch *et al.* [1] for transactions using SDA. In this attack, a man-in-the-middle sends the *success* response to the terminal's PIN verification request. The actual request is blocked and so the card believes that no PIN verification was required, ergo the disagreement between the terminal and the card. The terminal forwards the transaction to the bank (either for online authorization or to collect the funds), which then leads to the disagreement between the bank and the card.

A prerequisite for this attack to succeed is that, even if the terminal sends to the card the Cardholder Verification Method Results field (CVMR, tag `9F34`), which encodes the terminal's view of the CVM used, and the card detects the mismatch with its own view of the CVM used, the card does *not* abort the transaction. This appears to be the case in practice (although EMV's specification is not explicit about this) and has been successfully tested with three different Mastercard cards using our Android app. Such tests, even though they were conducted contactless, give us a fair degree of confidence that it also occurs with contact transactions.

Our analysis also exhibits that all transactions using SDA or DDA are vulnerable to a Transaction Cryptogram (TC) modification. This is because in neither of these methods the card authenticates the TC to the terminal (Table I, Remark 2).

In terms of secrecy, the results are identical for all models. The keys (private and shared) are secret, whereas the PAN is not. Interestingly, our analysis reports that the PIN is not secret. A man-in-the-middle attack between the card and the terminal can use a compromised bank's private key to produce the card records needed to make the terminal believe that the only CVM the card supports is plain PIN. These (fake) records are twofold: a list of supported CVMs composed of plain PIN only, and either an SSAD or a card's PKI certificate validating such a CVM list. The terminal thus downgrades to

plain PIN verification and consequently the PIN entered by the cardholder can be intercepted and learned by the attacker. This is a non-trivial attack though, as carrying this out in practice requires that the attacker:

1) knows a compromised bank's private key, and
2) inconspicuously controls the terminal's contact interface.

Our model considers these two conditions to be possible, at least in theory. However, in practice they are fairly difficult to achieve. We note that a single compromised bank is sufficient, and it needs not be the one that issued the victim's card.

Summary: We show that only three configurations of the EMV contact protocol guarantee secure transactions in terms of the three main properties we considered. These configurations all use CDA as the authentication method and are typeset in bold in Table I. In combination with online PIN as the cardholder verification method, the resulting target configuration allows all transactions (high and low value) and is secure. It is also the only one of these three that effectively checks that the person presenting the card is the legitimate cardholder. Instead, the other two configurations delegate this check to the cashier, e.g., by paper signature (whose actual verification is out of EMV's scope). This makes these two configurations not usable for high-value transactions in many countries.

B. Analysis Results for the EMV Contactless Protocol

Our analysis results for the 16 configurations of the EMV contactless protocol are summarized in Table II. Here Tamarin uncovered new, potentially high-valued attacks.

Our analysis shows that the Mastercard contactless protocol provides security for all high-value transactions. During transactions using SDA or DDA, the card does not authenticate the Application Cryptogram (AC) to the terminal (Table II, Lines 5, 7, 9, and 11, Remark 2). Therefore, during *offline* transactions using either of these methods, a man-in-the-middle can modify the AC (or Transaction Cryptogram due to being offline), which the terminal accepts given that it cannot verify its correctness. The mismatching AC will later be detected by the issuing bank. This violates both properties formalized in Definitions 1 and 2.

To our surprise, all-but-one of the Visa contactless protocol's configurations fail to provide security (the protocol is shown in Figure 2). For example, transactions authorized offline (Table II, Line 3, Remark 2) can be abused similarly to the aforementioned issue with Mastercard. Particularly critical are the violations of authentication in high-value transactions in EMV mode (Table II, Line 2, Remark 1). Our Tamarin analysis identifies a trace for an accepted transaction where neither the terminal nor the bank agree with the card on the Card Transaction Qualifiers (CTQ). The CTQ is a card-sourced data field that tells the terminal which CVM is to be used. The trace shows that, whereas the card's view of the CTQ is a request for online PIN verification, the terminal's view indicates that the Consumer Device CVM (CDCVM) was performed, which makes the terminal consider the cardholder verification process to be successfully completed. This is possible because no cryptographic protection of the CTQ is offered. This flaw is critical since it allows an attacker to bypass PIN verification for high-value transactions with a victim's card, as pointed out in the introduction.

In terms of secrecy, the results are identical for all models and are as expected. The keys (private and shared) and the PIN are secret, whereas the PAN is not.

Summary: Our analysis verifies that transactions with CDA-capable Mastercard cards that support online PIN verification are secure. Fortunately, this is the most common kind of Mastercard cards that banks are currently issuing. In contrast, critical flaws were encountered in common, currently used configurations of Visa cards. These flaws can be turned into practical attacks, which we describe in the next section.

VI. Attack and Defense

Our analysis of EMV's security uncovered numerous serious shortcomings. Particularly critical are the issues encountered in EMV contactless, because of their practical relevance given that tampering with the card-terminal contactless channel over NFC is much simpler than tampering with this channel over the contact chip. In this section we show how these issues can be exploited by an attacker to carry out fraudulent transactions. We also suggest fixes that lead to verified, secure contactless transactions.

A. Setup

We developed a proof-of-concept Android application to demonstrate the practical impact of the shortcomings uncovered by our formal analysis. Our application supports man-in-the-middle attacks on top of a *relay attack* [5]–[7] architecture, depicted in Figure 3. In this architecture, the attacker employs two mobile devices: one running our app in *Point-Of-Sale (POS) emulator* mode and the other in *card emulator* mode. Both devices must have NFC support and run Android 4.4 KitKat (API level 19) or later. The card emulator device must support Android's Host-based Card Emulation (HCE) [34].

To conduct the attacks, the POS emulator must be held near the card to be attacked and the card emulator must be held near the payment terminal. The two emulators communicate wirelessly through a TCP/IP socket channel over WiFi. A man-in-the-middle attack modifies, as appropriate:

- the inbound commands read from the wireless channel before delivering them to the card through the NFC channel, and
- the card's responses before transmitting them to the card emulator through the WiFi channel.

B. Bypassing Cardholder Verification

In a Visa contactless transaction, the card's response to the terminal's GET PROCESSING OPTIONS command carries the Card Transaction Qualifiers (CTQ). The CTQ is a 2-byte data field that instructs the terminal which Cardholder Verification Method (CVM) is to be used. As explained in Section V-B, our analysis revealed that the card authenticates the CTQ neither to the terminal nor to the bank (Table II,

TABLE II

ANALYSIS RESULTS FOR THE EMV CONTACTLESS PROTOCOL. ALL TARGET MODELS HAVE 60 RULES. THE LAST TWO COLUMNS INDICATE, IN THAT ORDER, THE NUMBER OF LINES OF TAMARIN CODE THAT THE MODEL COMPRISES, AND THE TIME TAKEN FOR OUR TAMARIN ANALYSIS ON A MACBOOK PRO LAPTOP RUNNING MACOS 10.15.4 WITH A QUAD-CORE INTEL CORE I7 @ 2.5 GHZ CPU AND 16 GB OF RAM. THE MODEL(S) FOR WHICH ALL FOUR PROPERTIES WERE VERIFIED ARE HIGHLIGHTED IN BOLD.

No.	Target model	Properties				LoC	Time
		executable	bank accepts	auth. to terminal	auth. to bank		
1	Visa_EMV_Low	✓	✓	×(1)	×(1)	822	2m02s
2	Visa_EMV_High	✓	✓	×(1)	×(1)	822	2m10s
3	Visa_DDA_Low	✓	×(2)	×(2)	✓	832	5m49s
4	**Visa_DDA_High**	✓	✓	✓	✓	840	28m57s
5	Mastercard_SDA_OnlinePIN_Low	✓	×(2)	×(2)	✓	830	4m36s
6	**Mastercard_SDA_OnlinePIN_High**	✓	✓	✓	✓	839	12m36s
7	Mastercard_SDA_NoPIN_Low	✓	×(2)	×(2)	✓	824	4m27s
8	Mastercard_SDA_NoPIN_High	–(3)	–	–	–	792	53s
9	Mastercard_DDA_OnlinePIN_Low	✓	×(2)	×(2)	✓	836	8m40s
10	**Mastercard_DDA_OnlinePIN_High**	✓	✓	✓	✓	845	27m07s
11	Mastercard_DDA_NoPIN_Low	✓	×(2)	×(2)	✓	830	8m22s
12	Mastercard_DDA_NoPIN_High	–(3)	–	–	–	798	1m02s
13	**Mastercard_CDA_OnlinePIN_Low**	✓	✓	✓	✓	845	22m00s
14	**Mastercard_CDA_OnlinePIN_High**	✓	✓	✓	✓	845	44m00s
15	**Mastercard_CDA_NoPIN_Low**	✓	✓	✓	✓	839	18m22s
16	Mastercard_CDA_NoPIN_High	–(3)	–	–	–	798	1m13s

Legend:

✓: property verified ×: property falsified –: not applicable (1): disagrees with the card on the CVM used

(2): disagrees with the card on the AC (3) high-value transactions without CVM are not completed over the contactless interface

Line 2, Remark 1). Our app exploits this and implements a man-in-the-middle attack that:

- **clears the 8th bit of CTQ's first byte**, which tells the terminal that online PIN verification is not required; and
- **sets the 8th bit of CTQ's second byte**, which tells the terminal that the Consumer Device CVM was performed.

Using our app, we have successfully carried out a number of real-world, PIN-less transactions above the domestic CVM-required limit with Visa credit as well as debit cards. Figure 4 shows screenshots of our app and the transaction log displayed in the POS emulator screen corresponds to one of such transactions.

Our attack should also work for the EMV Contactless Kernels 6 [35] (Discover) and 7 [36] (UnionPay), but these have not been tested yet. To avoid defrauding others, all of our tests were carried out with our own debit/credit cards, and in all attacks the purchased goods were paid for in full.

As discussed in Section II, Galloway and Yunusov [27] recently presented at BlackHat Europe another man-in-the-middle attack that also bypasses Visa's PIN verification. In contrast to our PIN bypass attack, their attack does not clear the 8th bit of CTQ's first byte. Instead, it clears the 7th bit of the second byte of the Terminal Transaction Qualifiers (TTQ). This bit tells the card whether the terminal requires cardholder verification for the transaction.

The TTQ is a terminal-sourced data field passed onto the card within the payload of the GET PROCESSING OPTIONS command. The TTQ is part of the Processing Data Object List (PDOL). According to the EMV Security and Key Management book [33], the Application Cryptogram (AC) is a MAC computed on the data referenced by the card's data object lists, namely the PDOL, the CDOL1, and the CDOL2, if applicable. Therefore, this (generic) cryptogram should defend against modification of the PDOL and of the TTQ in particular. Visa's proprietary cryptogram does not, as noted in [27]. Clearly, our attack works even if the TTQ is authenticated as it needs no modification.

Another noticeable difference between our attack and that of [27] is on the implementation side. Their attack prototype is composed of two wired Raspberry Pi boards. This setup is rather conspicuous and could not be easily used outside of a lab environment. In contrast, our proof-of-concept implementation is an innocent-looking phone app that can, and has been, easily used in live, attended terminals. Moreover, as opposed to Galloway and Yunusov's attack, ours does not require that the card and the payment terminal are physically close. In fact, one can extend our app so that the relay channel covers even overseas distances. Surprisingly, Visa has shown no intention to fix such vulnerabilities, as noted in [27].

Observe that our attack, as well as that of [27], presume that the attacker's device is physically within NFC proximity of the victim's card. These attacks can therefore be carried out by acquiring the actual card (e.g., stealing it or finding it if lost) or by holding the POS emulator near the card in the

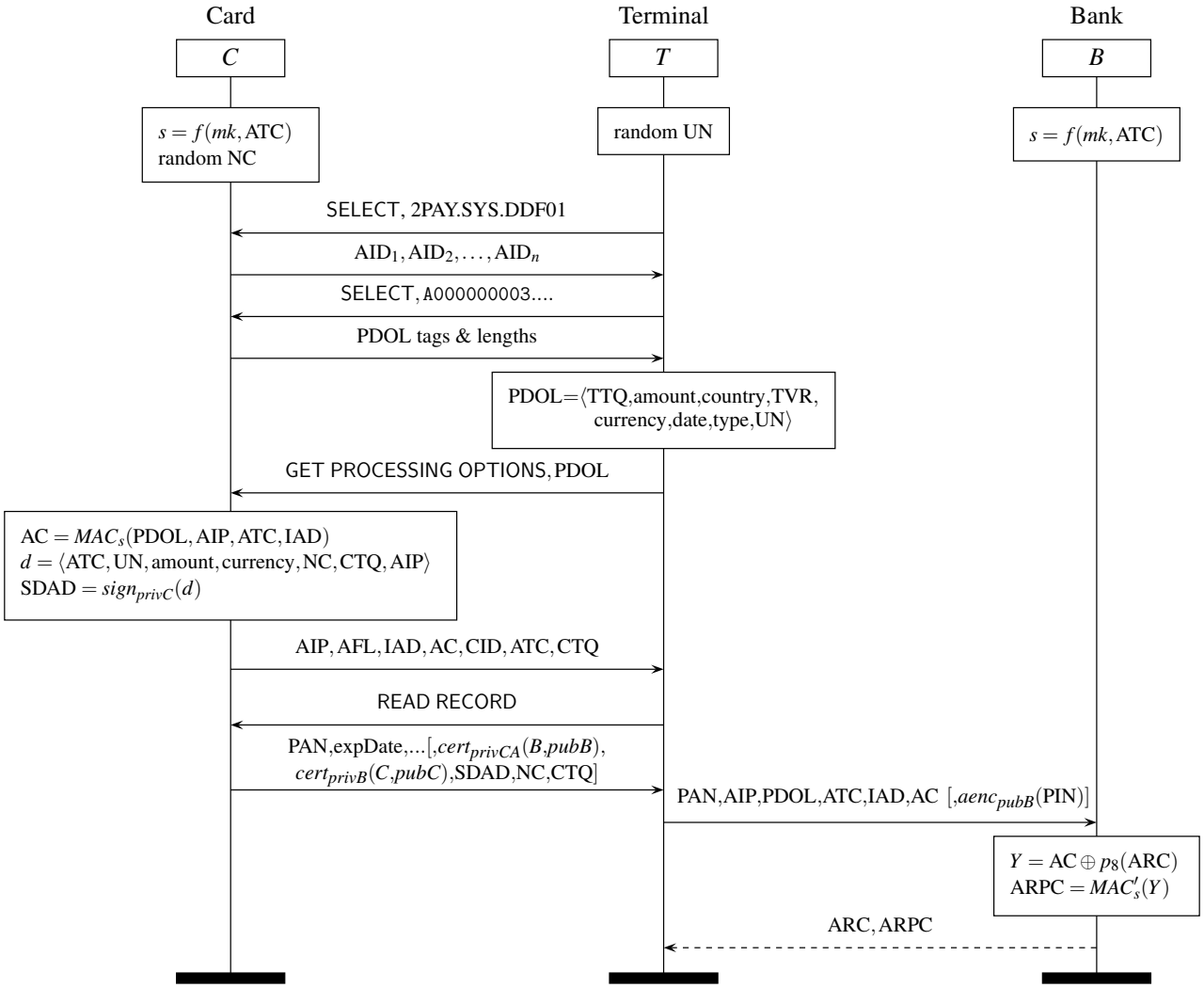


Fig. 2. The Visa contactless protocol. The terminal's request for cardholder verification and online authorization is encoded in the PDOL, specifically in the Terminal Transaction Qualifiers (TTQ, tag `9F66`). The card's response to the TTQ requests is encoded in the Card Transaction Qualifiers (CTQ, tag `9F6C`). The input to the AC represented here includes the full PDOL as per [33]; proprietary cryptograms might use fewer data objects [27].

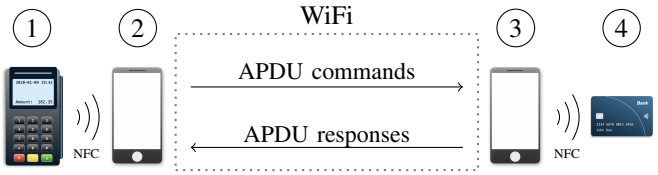


Fig. 3. A relay attack on contactless payment, where (1) is a payment terminal, (4) is a contactless card, and the attacker's equipment are the devices (2) and (3), which are the card emulator and the POS emulator, respectively.

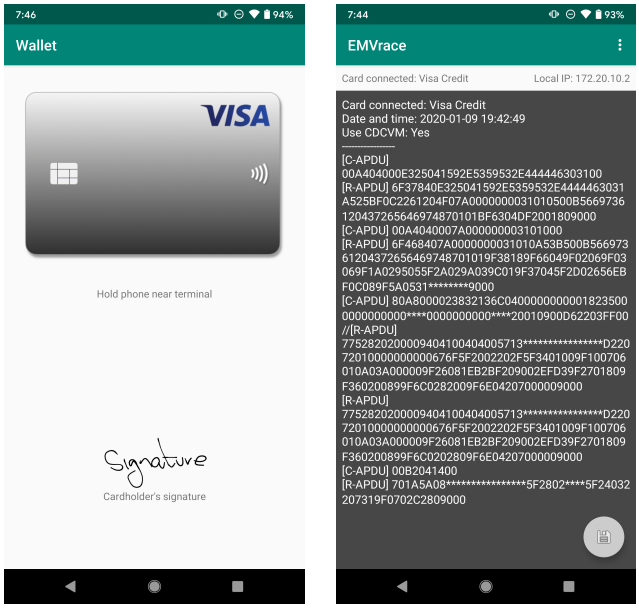
victim's possession.

Mastercard implements cardholder verification more in line with the traditional contact version of EMV. The card's support for the Consumer Device CVM is defined by the 2nd bit of AIP's first byte. Physical cards have this bit cleared. Hence, given that the AIP is included in the cryptogram, setting this bit will result in a declined transaction.

C. Unauthenticated Offline Transactions

For all low-value transactions of Visa as well as Mastercard with either SDA or DDA offline authentication, our Tamarin analysis uncovers a trace that violates the property that the bank accepts all terminal-accepted transactions (Table II, Remark 2). The trace represents a transaction where the attacker modifies the Transaction Cryptogram (TC) before delivering it to the terminal. The terminal reaches a state where the transaction is accepted given that the Signed Dynamic Authentication Data (SDAD), if produced and returned by the card, passes the terminal's verification. However, the issuing bank should later decline the transaction due to the wrong TC. Recall that the terminal can only verify the correctness of the SDAD but not of the TC since the latter is verified using a symmetric key only known to the card and the bank.

This constitutes a "free lunch" attack in that the criminal can purchase low-value goods or services without actually being charged at all. This however is unlikely to be an



(a) Card emulator (b) POS emulator

Fig. 4. Screenshots of our app. The card emulator may display the cardholder's signature, in which case, it should match the attacker's signature. The log displayed in the POS emulator corresponds to a real transaction of ca. $190 in the local currency.

attractive business model for criminals for two reasons. First, the fraudulent transactions are of low value. Second, the criminal's bank will likely not ignore the defrauded merchant's complaints indefinitely. For ethical reasons, we did not test this attack as it would constitute actual fraud.

D. Defenses against Attacks on Visa

As reported in Section V-B, the most common configuration of the Mastercard contactless protocol in current use (namely CDA in conjunction with online PIN) is secure. Visa's configurations, on the other hand, are not. Fortunately, Visa's problems can be fixed by implementing the three changes that we describe next. These changes can be realized by Visa and the banks in a reasonable amount of time and effort, without affecting those cards currently in circulation.

The Visa contactless protocol [31] specifies that special-purpose readers may perform Dynamic Data Authentication (DDA) for online transactions. This is indeed the only configuration of this protocol where all three security properties hold (Table II, Line 4). This is not a common configuration though, as indicated by our tests. We performed tests on over ten different live terminals at different merchants, and none of them used this configuration. Therefore, to prevent the PIN bypass attack described in Section VI-B, we recommend that terminals should use DDA for online transactions. That is, all the terminals must, for all transactions:

1) set the 1st bit of TTQ's first byte, and
2) verify the SDAD.

If implemented, these two measures would make high-value transactions be processed with Visa's secure configuration. This is of course assuming that the cards used for such transactions are capable of producing digital signatures, which modern cards are. Furthermore, to prevent the offline attack of Section VI-C, we propose that either:

3a) all terminals set the 8th bit of TTQ's second byte for all transactions; or
3b) $\langle \mathrm{NC}, \mathrm{CID}, \mathrm{AC}, \mathrm{PDOL}, \mathrm{ATC}, \mathrm{CTQ}, \mathrm{UN}, \mathrm{IAD}, \mathrm{AIP} \rangle$ is the input to the SDAD, i.e., d in Figure 2.

The fix 3(a) makes all transactions be processed online and is preferable over 3(b) because 3(a) does not require changes to the standard and therefore does not affect the consumer cards in circulation. Furthermore, offline transactions are rarely supported nowadays; none of the transactions we carried out during our tests were authorized offline. However, if the capability to process certain transactions offline is imperative (e.g., in transit systems or street parking meters) then more aggressive fixes would be needed such as that of 3(b).

We have verified the three fixes recommended here. Together, they defend against the attacks reported in this paper as well as any other attacks that might try to violate the considered security properties. These fixes, except for 3(b), can be deployed on the terminals' software/firmware and so they are attractive in terms of implementation because terminals' software updates should be significantly less expensive and faster than other, more aggressive actions such as blocking cards in circulation and issuing new ones.

VII. Conclusions

We have presented a formal model of the latest version of the EMV standard that features all relevant methods for offline data authentication, cardholder verification, and transaction authorization. Using the Tamarin tool, we conducted a full-scale, automatic, formal analysis of this model, uncovering numerous security flaws. These flaws violate fundamental security properties such as authentication and other guarantees about accepted transactions. We also used our model to identify EMV configurations that lead to secure transactions, and proved their correctness.

Our analysis revealed surprising differences between the security of the contactless payment protocols of Mastercard and Visa, showing that Mastercard is more secure than Visa. We found no major issues with the Mastercard protocol version running in modern cards. Our analysis revealed only minor shortcomings arising from older authentication modes (SDA and DDA) that seem hard to exploit in practice. In contrast, Visa suffers from several critical issues. The shortcomings we report on lead to serious, practical attacks, including a PIN bypass for transactions that surpass the cardholder verification limit. Using our proof-of-concept Android application, we successfully tested this attack on real-world transactions in actual stores. Our attack shows that the PIN is useless for Visa contactless transactions. As a result, in our view, the liability shift from banks to consumers or merchants is unjustified for such transactions: Banks, EMVCo, Visa, or some entity other than the consumer or merchant should be liable for such fraudulent transactions.

As part of our analysis we suggested and verified fixes that banks and Visa can deploy on existing terminals to prevent current and future attacks. The good news is that these fixes do not require changes to the EMV standard itself or to consumer cards currently in circulation and they can therefore be feasibly deployed by software updates.

References

[1] S. J. Murdoch, S. Drimer, R. J. Anderson, and M. Bond, "Chip and PIN is broken," in *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pp. 433–446, 2010.

[2] T. S. Heydt-Benjamin, D. V. Bailey, K. Fu, A. Juels, and T. O'Hare, "Vulnerabilities in first-generation rfid-enabled credit cards," in *Financial Cryptography and Data Security, 11th International Conference, FC 2007, and 1st International Workshop on Usable Security, USEC 2007, Scarborough, Trinidad and Tobago, February 12-16, 2007. Revised Selected Papers*, pp. 2–14, 2007.

[3] M. Roland and J. Langer, "Cloning credit cards: A combined pre-play and downgrade attack on EMV contactless," in *7th USENIX Workshop on Offensive Technologies, WOOT '13, Washington, D.C., USA, August 13, 2013*, 2013.

[4] S. Drimer and S. J. Murdoch, "Keep your enemies close: Distance bounding against smartcard relay attacks," in *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*, 2007.

[5] L. Francis, G. P. Hancke, K. Mayes, and K. Markantonakis, "Practical relay attack on contactless transactions by using NFC mobile phones," *IACR Cryptology ePrint Archive*, vol. 2011, p. 618, 2011.

[6] L. Sportiello and A. Ciardulli, "Long distance relay attack," in *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, pp. 69–85, 2013.

[7] T. Chothia, F. D. Garcia, J. de Ruiter, J. van den Breekel, and M. Thompson, "Relay cost bounding for contactless EMV payments," in *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pp. 189–206, 2015.

[8] M. Bond, O. Choudary, S. J. Murdoch, S. P. Skorobogatov, and R. J. Anderson, "Chip and skim: Cloning EMV cards with the pre-play attack," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pp. 49–64, 2014.

[9] M. Emms, B. Arief, L. Freitas, J. Hannon, and A. P. A. van Moorsel, "Harvesting high value foreign currency transactions from EMV contactless credit cards without the PIN," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pp. 716–726, 2014.

[10] H. Ferradi, R. Géraud, D. Naccache, and A. Tria, "When organized crime applies academic results: a forensic analysis of an in-card listening device," *J. Cryptographic Engineering*, vol. 6, no. 1, pp. 49–59, 2016.

[11] S. Meier, B. Schmidt, C. Cremers, and D. A. Basin, "The TAMARIN prover for the symbolic analysis of security protocols," in *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, pp. 696–701, 2013.

[12] B. Schmidt, S. Meier, C. J. F. Cremers, and D. A. Basin, "Automated analysis of Diffie-Hellman protocols and advanced security properties," in *25th IEEE Computer Security Foundations Symposium, CSF 2012, Cambridge, MA, USA, June 25-27, 2012*, pp. 78–94, 2012.

[13] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, "A comprehensive symbolic analysis of TLS 1.3," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pp. 1773–1788, 2017.

[14] D. A. Basin, J. Dreier, L. Hirschi, S. Radomirovic, R. Sasse, and V. Stettler, "A formal analysis of 5g authentication," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pp. 1383–1396, 2018.

[15] J. de Ruiter and E. Poll, "Formal analysis of the EMV protocol suite," in *Theory of Security and Applications - Joint Workshop, TOSCA 2011, Saarbrücken, Germany, March 31 - April 1, 2011, Revised Selected Papers*, pp. 113–129, 2011.

[16] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Distance-bounding protocols: Verification without time and location," in *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pp. 549–566, 2018.

[17] A. Debant, S. Delaune, and C. Wiedling, "A symbolic framework to analyse physical proximity in security protocols," in *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2018, December 11-13, 2018, Ahmedabad, India*, pp. 29:1–29:20, 2018.

[18] B. Blanchet, "An efficient cryptographic protocol verifier based on Prolog rules," in *14th IEEE Computer Security Foundations Workshop (CSFW-14 2001), 11-13 June 2001, Cape Breton, Nova Scotia, Canada*, pp. 82–96, 2001.

[19] K. Bhargavan, C. Fournet, A. D. Gordon, and S. Tse, "Verified interoperable implementations of security protocols," in *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy*, pp. 139–152, 2006.

[20] D. Dolev and A. C. Yao, "On the security of public key protocols," *IEEE Trans. Information Theory*, vol. 29, no. 2, pp. 198–207, 1983.

[21] D. A. Basin and C. Cremers, "Know your enemy: Compromising adversaries in protocol analysis," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 2, pp. 7:1–7:31, 2014.

[22] T. Beth and Y. Desmedt, "Identification tokens - or: Solving the chess grandmaster problem," in *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, pp. 169–177, 1990.

[23] S. Brands and D. Chaum, "Distance-bounding protocols (extended abstract)," in *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, pp. 344–359, 1993.

[24] T. Chothia, J. de Ruiter, and B. Smyth, "Modelling and analysis of a hierarchy of distance bounding attacks," in *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018.*, pp. 1563–1580, 2018.

[25] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Post-collusion security and distance bounding," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019.*, pp. 941–958, 2019.

[26] A. Debant and S. Delaune, "Symbolic verification of distance bounding protocols," in *Principles of Security and Trust - 8th International Conference, POST 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, pp. 149–174, 2019.

[27] L.-A. Galloway and T. Yunusov, "First contact: New vulnerabilities in contactless payments," in *Black Hat Europe 2019*, 2019.

[28] G. Lowe, "A hierarchy of authentication specification," in *10th Computer Security Foundations Workshop (CSFW '97), June 10-12, 1997, Rockport, Massachusetts, USA*, pp. 31–44, 1997.

[29] C. Cremers and S. Mauw, *Operational Semantics and Verification of Security Protocols*. Information Security and Cryptography, Springer, 2012.

[30] EMVCo, *EMV Contactless Specifications for Payment Systems, Book C-2, Kernel 2 Specification, Version 2.8.* April 2019.

[31] EMVCo, *EMV Contactless Specifications for Payment Systems, Book C-3, Kernel 3 Specification, Version 2.8.* April 2019.

[32] Anonymous, "A Tamarin model of EMV." https://github.com/EMVrace/EMVerify, 2020. Accessed: June 2020.

[33] EMVCo, "EMV Integrated Circuit Card Specifications for Payment Systems, Book 2, Security and Key Management, Version 4.3," November 2011.

[34] Google, "Host-based card emulation overview." https://developer.android.com/guide/topics/connectivity/nfc/hce, 2019. Accessed: December 2019.

[35] EMVCo, *EMV Contactless Specifications for Payment Systems, Book C-6, Kernel 6 Specification, Version 2.8.* April 2019.

[36] EMVCo, *EMV Contactless Specifications for Payment Systems, Book C-7, Kernel 7 Specification, Version 2.8.* April 2019.

Acronyms

AAC Application Authentication Cryptogram. 6
AC Application Cryptogram. 2, 6, 8–12
AFL Application File Locator. 4

AID Application Identifier. 4
AIP Application Interchange Profile. 4, 6, 8, 12
APDU Application Protocol Data Unit. 3, 12
ARC Authorization Response Code. 6
ARPC Authorization Response Cryptogram. 6
ARQC Authorization Request Cryptogram. 6
ATC Application Transaction Counter. 6, 8

CDA Combined Dynamic Data Authentication. 2, 4, 6, 8, 10, 13
CDCVM Consumer Device CVM. 2, 4, 8, 10–12
CDOL Card Risk Management Data Object List. 4, 6, 11
CID Cryptogram Information Data. 6
CTQ Card Transaction Qualifiers. 8, 10–12
CVM Cardholder Verification Method. 2–4, 6, 8–12, 15
CVMR Cardholder Verification Method Results. 9

DDA Dynamic Data Authentication. 2, 4, 8–10, 12, 13
DDOL Dynamic Data Object List. 4

HCE Host-based Card Emulation. 10

IAD Issuer Application Data. 8

NFC Near Field Communication. 3, 10–12

ODA Offline Data Authentication. 2, 4

PAN Primary Account Number. 4, 7–10
PDOL Processing Data Object List. 4, 11, 12
POS Point-Of-Sale. 10, 12
PSE Payment System Environment. 4, 8

SDA Static Data Authentication. 2, 4, 9, 10, 12, 13
SDAD Signed Dynamic Authentication Data. 4, 6, 12, 13
SSAD Signed Static Authentication Data. 4, 9

TA Transaction Authorization. 2, 4
TC Transaction Cryptogram. 6, 9, 10, 12
TTQ Terminal Transaction Qualifiers. 11–13

UN Unpredictable Number. 4, 6

APPENDIX

A. Target Models Generation

We construct the *target models* from the rules of a generic model as well as extra rules that produce the Commit facts used for the (in)validation of properties. We have written a `Makefile` script that generates the target models by instantiating the following variables:

- `generic`: defines the generic model. Valid instances are:
  - `Contact`, and
  - `Contactless`.
- `kernel`: defines the kernel of the contactless transaction. Valid instances are:
  - `Mastercard`, and
  - `Visa`.

```
/*if(Visa)
rule Terminal_Commits_ARQC_Visa:
    let PDOL = <TTQ, $amount, country, currency,
                date, type, ~UN>
        /*if(DDA) AIP = <'DDA', data> endif(DDA)*/
        /*if(EMV) AIP = <'EMV', data> endif(EMV)*/
        /*if(Low) value = 'Low' endif(Low)*/
        /*if(High) value = 'High' endif(High)*/
        transaction = <~PAN, AIP, CVM, PDOL, ATC,
                       AC, IAD>
    in
    [ Terminal_Received_AC_Visa($Terminal, $Bank,
        $CA, nc, 'ARQC', transaction, ~channelID),
      !Value($amount, value),
      Recv($Bank, $Terminal,
        <~channelID, 'Visa', '2'>, <'ARC', ARPC>)
    ]
  --[ TerminalAccepts(transaction),
      Commit(nc, ~PAN,
        <'Card', 'Terminal', transaction>),
      Commit($Terminal, $Bank,
        <'Bank', 'Terminal', transaction>),
      Honest($CA), Honest($Bank),
      Honest($Terminal), Honest(~PAN)]->
    [ ]
endif(Visa)*/
```

Fig. 5. Tamarin code snippet from the EMV contactless protocol model.

- `auth`: defines the Offline Data Authentication (ODA) method. Valid instances are:
  - `SDA`,
  - `DDA`,
  - `CDA`, and
  - `EMV` (for contactless transactions only).
- `CVM`: defines the cardholder verification method used/-supported. Valid instances are:
  - `NoPIN`,
  - `PlainPIN` (for contact transactions only),
  - `EncPIN` (enciphered PIN, for contact transactions only), and
  - `OnlinePIN`.
- `value`: defines the value of the contactless transaction. Valid instances are:
  - `Low` (below the CVM-required limit), and
  - `High` (above the CVM-required limit).
- `authz`: defines the type of authorization of the contact transaction. Valid instances are:
  - `Offline`, and
  - `Online`.

The execution of `make` with a choice of variable instances determining a target configuration generates the target model and analyzes it with Tamarin. To understand how we instrument the actual target models auto-generation, consider the code snippet depicted in Figure 5, taken from our generic model of the EMV contactless protocol.

This piece of code is *activated* (uncommented), and so the rule becomes part of the target model, if the target configuration includes `kernel=Visa`. Furthermore, depending on

```
rule Terminal_Commits_ARQC_Visa:
    let PDOL = <TTQ, $amount, country, currency,
                date, type, ~UN>
        AIP = <'DDA', data>
        value = 'High'
        transaction = <~PAN, AIP, CVM, PDOL, ATC,
                       AC, IAD>
    in
    [ Terminal_Received_AC_Visa($Terminal, $Bank,
        $CA, nc, 'ARQC', transaction, ~channelID),
      !Value($amount, value),
      Recv($Bank, $Terminal,
        <~channelID, 'Visa', '2'>, <'ARC', ARPC>)
    ]
  --[ TerminalAccepts(transaction),
      Commit(nc, ~PAN,
        <'Card', 'Terminal', transaction>),
      Commit($Terminal, $Bank,
        <'Bank', 'Terminal', transaction>),
      Honest($CA), Honest($Bank),
      Honest($Terminal), Honest(~PAN)]->
    [ ]
```

Fig. 6. Tamarin code snippet from the Visa_DDA_High target model.

the rest of the target configuration, the AIP and value are activated. For example, if our target configuration includes `auth=DDA` and `value=High`, then the rule becomes the one depicted in Figure 6. This (new) rule models the terminal's acceptance of an online-authorized transaction and produces the corresponding Commit and TerminalAccepts facts.