Security Analysis of EMV Channel Establishment Protocol in An Enhanced Security Model

Yanfei Guo(✉), Zhenfeng Zhang, Jiang Zhang, and Xuexian Hu

Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing, China
{guoyanfei,zfzhang,zhangjiang,huxuexian}@tca.iscas.ac.cn

Abstract. The EMV chip-and-pin system is one of the most widely used cryptographic system in securing credit card and ATM transactions. As suggested by the EMV consortium, the existing RSA-based EMV system will be upgraded to Elliptic Curve Cryptography (ECC) based system. In CCS 2013, Brzuska et al. made the first step to analyze the security of the ECC-based EMV channel establishment protocol in a channel establishment security model, and showed that a slightly modified version of the protocol meets the intended security goals. In this paper, we continue this strand of research by analyzing the security of the ECC-based EMV protocol in a strong channel establishment security model which allows the adversary to get ephemeral private keys of the involved parties. We find that the original protocol is not secure in our security model because the adversary can impersonate a Card entity. Then we slightly modify the protocol almost with no addition of computation cost and show that the resulting protocol is secure in our security model under standard cryptographic assumptions.

1 Introduction

As an international specification for debit and credit card payments [1,7,25], the EMV system has been widely deployed in more than 1.6 million credit cards [24]. The current EMV system is based on RSA public key cryptography, and symmetric-key cryptography (such as DES and AES) [9–12]. The system is recognized as of great significance in providing secure transaction and reducing card payment fraud [26].

Due to the practical significance of EMV system, many researchers have made efforts to investigate its security [1,5–8,25,26,28]. Most of the works focused on the RSA-based EMV system, while the EMV consortium are planning to upgraded the existing RSA-based EMV system to ECC-based system. In November 2012, the EMV consortium released a Request-For-Comments [13] on a draft specification for the ECC-based EMV channel establishment protocol, which is used for establishing a common secret seed and a channel to protect all subsequent messages between a Card and a Terminal.

© Springer International Publishing Switzerland 2015
L.C.K. Hui et al. (Eds.): ICICS 2014, LNCS 8958, pp. 305–320, 2015.
DOI: 10.1007/978-3-319-21966-0_22

According to the EMV consortium, the protocols are designed to (i) provide authentication of the Card (the authenticated parties) by the Terminal (the unauthenticated parties), (ii) detect modifications to the communications, (iii) and protect against eavesdropping and card tracking [13].

As far as we know, there is only one work [3] by Brzuska et al. which analyzes the security of the protocol. They suggested minor changes to the protocol by choosing the ephemeral secret key of the Card entity from a larger space, and establishing two keys instead of one key for the authenticated encryption scheme. They proved the modified protocol (we will call it ECC-based EMV protocol in our paper) is secure in a carefully designed channel establishment security model. However, in their model, adversaries are not allowed to get the participants' ephemeral secret keys.

1.1 Security Model

Bellare and Rogaway [2] proposed the first formal security model for key establishment protocols, known as the BR model. The BR model captures basic security requirements for authenticated key establishment protocols such as known key security and impersonation resilience. Canetti and Krawczyk [4] consider the leakage of the parties' static secret keys and sessions' state (i.e., CK model). Whereas, both above two models fail to capture several advanced attacks such as key compromise impersonation (given a static secret key, an adversary tries to impersonate some honest party in order to fool the owner of the leaked secret key), the breaking of weak perfect forward secrecy (given the static secret keys of participants of the protocol, the adversary tries to recover a previous session key) and maximal exposure attacks (an adversary tries to distinguish the session key from a random value under the disclosure of any pair of secret static keys and ephemeral secret keys of the participants in the session except for both the secret keys of a single participant) [15,20]. In order to capture the advanced attacks mentioned above, LaMacchia et al. [22] proposed a well known security model, (i.e., eCK model [14,17,27,29]) which allows the adversary to obtain the ephemeral secret keys.

However, the above security models seem not fit for the practical requirement in real word protocols such as the TLS protocol [3,18]. Therefore, researchers started to focus on the study of more accurate portrayal of the widely used channel establishment protocols. In 2012, Jager et al. [18] defined the security model for authenticated and confidential channel establishment (ACCE) protocol in which they proved the TLS-DHE is secure under the assumption that the TLS record layer is a stateful length hiding authenticated encryption (sLHAE) scheme.

Later, Krawczyk et al. [21] and Kohlar et al. [19] proved the security of TLS-RSA, TLS-DHE and TLS-DH in the ACCE model respectively. Giesen et al. [16] extended the ACCE model to give the formal treatment of renegotiation in secure channel establishment protocols and analyzed the security of TLS renegotiation in the extended model. Li et al. [23] introduced the definition of ACCE security for authentication protocols with pre-shared keys and proved the security

of the Pre-Shared Key Ciphersuites of TLS in the model. Following the similar idea, Brzuska et al. [3] analyzed the ECC-based EMV protocol in a channel establishment security model, which captures one-way authentication key agreement followed by composition with a secure channel and unlinkability property. Nevertheless, the model doesn't describe the situations for the leakage of the ephemeral secret keys and leakage of static secret keys for the parties involved in the target session, which seems not meet the practical requirement [14,20,22].

1.2 Our Contribution

In this paper, we propose a strong security model for one-way authentication channel establishment protocols and point that the ECC-based EMV protocol is not secure in our security model. Concretely, if the adversary can get ephemeral keys and session keys of the sessions, he can impersonate a valid Card entity to Terminals. We make slight modification almost without addition of computation cost to the protocol and show that the modified protocol is secure in our security model.

In our security model, we strengthen the adversaries' ability by allowing them to obtain the ephemeral keys of the sessions through $EphemeralKeyReveal$ queries. In particular, we allow the adversary to obtain either the static or the ephemeral secret keys of the authenticated party involved in the target session, but not both the static and ephemeral secrets of that party. This enables us to capture the forward security property in one-way authentication setting which means that the compromise of the authenticated party's static secret key can not help the adversary to recover the party's previously established session keys. We note that this property is not captured in the previous one-way authentication channel establishment security models [3,21].

The security proof is given in the random oracle model under standard cryptographic assumptions, i.e., Gap Diffie-Hellman assumptions, the existence of EUF-CMA digital signatures, and the existence of IND-sfCCA secure and INT-sfPTXT secure authenticated encryption schemes (see Sect. 2).

2 Preliminaries and Definitions

We denote $\mathbb{G} = E(\mathbb{F}_p)$ to be a Diffie-Hellman group defined over an elliptic curve of prime order q, which uses a base point $P \in \mathbb{G}$. The prime q is a function of an implicit security parameter λ. Denote with $\emptyset$ the empty string. Assume that messages in a transcript s are represented as binary strings. Let $|s|$ denote the number of messages of s. $Prefix(s_1, s_2) = true$ if the first $|s_1|$ messages (provided not empty) in transcripts s_1 and s_2 are pairwise equivalent as binary strings, and $false$ otherwise. $\stackrel{r}{\leftarrow}$ means that the value on the left is chosen uniformly at random from the set that on the right of the notion.

Definition 1 (Computational Diffie-Hellman (CDH)). *The CDH problem asks that given $P, rP, sP \in \mathbb{G}$, where $r, s \stackrel{r}{\leftarrow} \mathbb{F}_q$, compute rsP. We say that the*

CDH problem is (t, ϵ_{CDH}) *hard if for any adversary* $\mathcal{A}$ *that runs in time* t *it holds that*

$$Pr[r, s \xleftarrow{r} \mathbb{F}_q : \mathcal{A}(rP, sP) = rsP] \leq \epsilon_{CDH}$$

Definition 2 (Gap-Diffie-Hellman(Gap-DH)). *Let* $\mathcal{O}_{DDH}$ *be an oracle that solves the DDH problem in* $\mathbb{G}$*, i.e. takes as input* $rP, sP, uP \in \mathbb{G}$*, and outputs one if* $uP = rsP$ *and zero otherwise. The Gap Diffie-Hellman problem then asks that given* $P, aP, bP \in \mathbb{G}$ *where* $a, b \xleftarrow{r} \mathbb{F}_q$*, and access to* $\mathcal{O}_{DDH}$*, compute* abP *(i.e. solve CDH). We say that the Gap-DH problem is* $(t, \epsilon_{Gap\text{-}DH})$ *hard if for any adversary* $\mathcal{A}$ *that runs in time* t *it holds that*

$$Pr[a, b \xleftarrow{r} \mathbb{F}_q : \mathcal{A}^{\mathcal{O}_{DDH}}(P, aP, bP) = abP] \leq \epsilon_{Gap\text{-}DH}$$

A digital signature scheme is a triple $SIG = (SIG.Gen, SIG.Sign, SIG.Vfy)$, consisting of a key generation algorithm $(sk, pk) \xleftarrow{\$} SIG.Gen(1^\lambda)$ generating a (public) verification key pk and a secret signing key sk on input of security parameter λ, signing algorithm $\sigma \leftarrow SIG.Sign(sk, m)$ generating a signature for message m, and verification algorithm $SIG.Vfy(pk, \sigma, m)$ returning 1, if σ is a valid signature for m under key pk, and 0 otherwise. Consider the following security experiment played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

1. The challenger generates a public/secret key pair $(sk, pk) \xleftarrow{\$} SIG.Gen(1^\lambda)$, the adversary receives pk as input.
2. The adversary may query arbitrary messages m_i to the challenger. The challenger replies to each query with a signature $\sigma_i = SIG.Sign(sk, m_i)$. Here i is an index, ranging between $1 \leq i \leq q$ for some $q \in \mathbb{N}$. Queries can be made adaptively.
3. Eventually, the adversary outputs a message/signature pair (m, σ).

Definition 3 (EUF-CMA). *We say that* SIG *is* (t, ϵ_{SIG}) *EUF-CMA secure against existential forgeries under adaptive chosen-message attacks, if for all adversaries* $\mathcal{A}$ *that run in time* t *it holds that*

$$Pr[(m, \sigma) \leftarrow \mathcal{A}^{Sign(sk,\cdot)}(1^\lambda, pk) \text{ such that } SIG.Vfy(pk, m, \sigma) = 1 \wedge m \notin \{m_1, \cdots, m_q\}] \leq \epsilon_{SIG}$$

Note that we have $q \leq t$, i.e., the number of allowed queries q is bounded by the running time t of the adversary.

An authenticated encryption (AE) scheme $AE = (\mathcal{K}, enc, dec)$ consists of three algorithms. The randomized key generation algorithm $\mathcal{K}$ returns a key K. The encryption algorithm enc, takes key K and a plaintext and returns a ciphertext. The decryption algorithm dec takes key K and a ciphertext and returns either a plaintext or a special symbol $\perp$ indicating failure. The following two properties are variants of the stateful security models of AE scheme [3]. $enc_\kappa(h, m; st_e)$ is a symmetric encryption oracle for κ, which takes as input a header h, message m, outputs ciphertext c and updated state st_e. $dec_\kappa(h, c; st_d)$ is a symmetric decryption oracle for κ which takes a header h, a ciphertext c as input, outputs message m or $\perp$ and updated state st_d. $LR_b(m_0, m_1)$ for $b \in \{0, 1\}$

outputs m_b. So a left-or-right encryption oracle $enc_\kappa(h, LR_b(m_0, m_1); st_e)$ outputs $enc_\kappa(h, m_b; st_e)$ if $m_0 = m_1$ and $\perp$ otherwise. C-E (C-D) is the set of ciphertexts output by (input to) the left-or-right encryption (decryption) oracle. M-E (M-D) is the set of messages input to (output by) the encryption (decryption) oracle.

Definition 4 (IND-sfCCA [3]). *Consider the authenticated encryption scheme $AE = \{enc_\kappa, dec_\kappa\}$. Let $\mathcal{A}$ be an adversary with access to a left-or-right encryption oracle $enc_\kappa(h, LR_b(m_0, m_1); st_e)$ and a decryption oracle $dec_\kappa(h, c; st_d)$. It is mandated that any two messages queried to $enc_\kappa(h, LR_b(m_0, m_1); st_e)$ have equal length. The ind-sfcca experiment is defined as in Fig. 1. The attacker wins when $b' = b$, and his advantage is defined as*

$$Adv_{AE}^{ind\text{-}sfcca}(\mathcal{A}) = Pr[Exec_{AE}^{ind\text{-}sfcca\text{-}1}(\mathcal{A}) = 1] - Pr[Exec_{AE}^{ind\text{-}sfcca\text{-}0}(\mathcal{A}) = 1].$$

We say that AE is $(t, \epsilon_{ind\text{-}sfcca})$ IND-sfCCA secure, if for all adversaries $\mathcal{A}$ that run in time t it holds that

$$Adv_{AE}^{ind\text{-}sfcca}(\mathcal{A}) \leq \epsilon_{ind\text{-}sfcca}.$$

$$Adv_{AE}^{ind\text{-}sfcca}(\mathcal{A}) \leq \epsilon_{ind\text{-}sfcca}.$$

The *ind-sfcca* experiment

$Exec_{AE}^{ind\text{-}sfcca\text{-}b}(\mathcal{A})$
$\kappa \xleftarrow{r} \{0,1\}^k$, C-$E := \emptyset$, C-$D := \emptyset$, $st_e := \emptyset$ and $st_d := \emptyset$
Run $\mathcal{A}^{enc_\kappa, dec_\kappa}$
  Reply to $enc_\kappa(h, LR_b(m_0, m_1); st_e)$ as follows:
    $(c, st_e) \xleftarrow{r} enc_\kappa(h, m_b, st_e)$
    C-$E \leftarrow C$-$E \cup c; \mathcal{A} \Leftarrow c$
  Reply to $dec_\kappa(h, c; st_d)$ as follows:
    $(m, st_d) \xleftarrow{r} dec_\kappa(h, c; st_d)$
    if $m \neq \perp$ then
      C-$D \leftarrow C$-$D \cup c$
      if $Prefix(C\text{-}D, C\text{-}E) = false$ then $\mathcal{A} \Leftarrow m$
Until $\mathcal{A}$ returns a bit b'
return b'

The *int-sfptxt* experiment

$Exec_{AE}^{int\text{-}sfptxt}(\mathcal{A})$
$\kappa \xleftarrow{r} \{0,1\}^k, d := 0$
M-$E := \emptyset$, M-$D := \emptyset$, $st_e := \emptyset$ and $st_d := \emptyset$
Run $\mathcal{A}^{enc_\kappa, dec_\kappa}$
  Reply to $enc_\kappa(h, m; st_e)$ as follows:
    $(c, st_e) \xleftarrow{r} enc_\kappa(h, m; st_e)$
    M-$E \leftarrow M$-$E \cup m; \mathcal{A} \Leftarrow c$
  Reply to $dec_\kappa(h, c; st_d)$ as follows:
    $(m, st_d) \xleftarrow{r} dec_\kappa(h, c; st_d)$
    if $m \neq \perp$ then
      M-$D \leftarrow M$-$D \cup m; \mathcal{A} \Leftarrow 1$
      if $Prefix(M\text{-}D, M\text{-}E) = false$ then $d := 1$
    else $\mathcal{A} \Leftarrow 0$
Until $\mathcal{A}$ halts
return d

Fig. 1. The *ind-sfcca* (resp. *int-sfptxt*) experiment

Definition 5 (INT-sfPTXT [3]). *Consider the scheme $AE = \{enc_\kappa, dec_\kappa\}$. Let $\mathcal{A}$ be an adversary with oracle access to $enc_\kappa(h, m; st_e)$ and $dec_\kappa(h, c; st_d)$. The int-sfptxt experiment is defined as in Fig. 1. The advantage $Adv_{AE}^{int\text{-}sfptxt}(\mathcal{A})$ of an adversary is defined as*

$$Adv_{AE}^{int\text{-}sfptxt}(\mathcal{A}) = Pr[Exec_{AE}^{int\text{-}sfptxt}(\mathcal{A}) = 1].$$

We say that AE is $(t, \epsilon_{int\text{-}sfptxt})$ INT-sfPTXT secure, if for all adversaries $\mathcal{A}$ that run in time t it holds that

$$Adv_{AE}^{int\text{-}sfptxt}(\mathcal{A}) \leq \epsilon_{int\text{-}sfptxt}.$$

3 EMV Channel Establishment Protocol

The original specification of the EMV channel establishment protocol can be found in [13]. In this section, the EMV channel establishment protocol modified by [3] is presented. There are two kinds of participants in the system: Card and Terminal. Each Card holds a certificate which is a digital signature of its public key $Q_C = dP \in \mathbb{G}$. The secret key of the Card participant is $d \xleftarrow{r} \mathbb{F}_q$. The protocol uses a hash function H that takes elements in the group $\mathbb{G}$ and maps them onto a pair of keys for the authenticated encryption scheme.

After the protocol has established secret keys, it uses them in a secure channel protocol $(SendCh, ReceiveCh)$. On input an application message m and state st_e, $SendCh$ returns a channel message ch. On input a channel message ch and state st_d, $ReceiveCh$ returns an application message m. The secure channel protocol is based on a stateful AE scheme $AE = \{enc, dec\}$. Assume that all plaintext headers used by the secure channel are unauthenticated, implying that no header is sent in clear as part of the AE scheme. The states st_e and st_d here model the fact that in practice sequence numbers are used to ensure that messages are delivered in order, thus the operations are stateful. The protocol is presented in Fig. 2. The static key of the Card is d, the ephemeral key of the Card is a, the ephemeral key of the Terminal is e, and the session keys are $(\kappa_e^C, \kappa_d^C) = (\kappa_d^T, \kappa_e^T) = H(eadP)$.

Card (C)		Terminal (T)
$a \in_R \mathbb{F}_q$		
	$\xrightarrow{A=aQ_C}$	
		$e \in_R \mathbb{F}_q$
	$\xleftarrow{E=eP}$	
$(\kappa_e^C, \kappa_d^C) = H(daE)$		$(\kappa_d^T, \kappa_e^T) = H(eA)$
$(ch, st_e^C) = SendCh_{\kappa_e^C}(cert_C \|\| a \|\| Q_C; st_e^C)$	$\xrightarrow{ch}$	$(cert_C \|\| a \|\| Q_C, st_d^T) = ReceiveCh_{\kappa_d^T}(ch, st_d^T)$
		Check $ver_{pk}(cert_C, Q_C) \stackrel{?}{=} true$
		Check $aQ_C \stackrel{?}{=} A$
$(ch_1, st_e^C) = SendCh_{\kappa_e^C}(m_1; st_e^C)$	$\xrightarrow{ch_1}$	$(m_1, st_d^T) = ReceiveCh_{\kappa_d^T}(ch_1; st_d^T)$
$(m_2, st_d^C) = ReceiveCh_{\kappa_d^C}(ch_2; st_d^C)$	$\xleftarrow{ch_2}$	$(ch_2, st_e^T) = SendCh_{\kappa_e^T}(m_2; st_e^T)$

Fig. 2. ECC-based EMV channel establishment protocol

4 The Enhanced Security Model

In this section we present a stronger security model for one-way authentication channel establishment protocols which is inspired by the security models of [3] and [22]. We enhance the channel establishment security model [3] by considering *EphemeralKeyReveal* queries and using stronger freshness definition.

This enhancement enables us to capture the forward security property in one-way authentication setting which is not captured in the previous one-way authentication channel establishment security models [3,21].

4.1 Preliminaries

Let $n_C, n_T, n_S \in N$ be positive integers. Assume that there are n_C authenticated entities and n_T unauthenticated entities in the system. Each party can establish at most n_S sessions. Each party in the system has a distinct identity i.

The protocol description is defined by two efficiently computable stateful (sub)-protocols $P = \{\Pi, \mathcal{G}\}$. The protocol Π defines how honest parties behave and $\mathcal{G}$ is the key generation algorithm. Each execution of the protocol can be modeled as an oracle Π_i^s, which means that the session is party i's s-th instance of carrying out the protocol with some partner j (which is determined during the protocol execution). The oracle has access to its owner's private key and independently maintains a list of internal state information as follows:

- $\delta \in \{derived, accept, reject, \perp\}$ is current state of the key exchange (initialized to $\perp$). When the session owner derives a session key, he marks the session as *derived*. When the key establishment protocol ends successfully (and stipulates that no further messages are to be received), the session owner marks the session as *accepted*. An *accepted* session must be *derived*. $\delta = reject$ means that the session rejects.
- $\rho \in \{initiator, responder\}$ is the role of the participant.
- *pid* is the partner identifier which is determined during the protocol execution.
- *sid* is the session identifier which can be defined by a transcript of all the messages the session receives and sends.
- $\kappa = (\kappa_e^\rho, \kappa_d^\rho) \in (\{0,1\}^* \cup \{\perp\})^2$ is the agreed pair of keys. The order of the keys depends on the role. It is initialized as $(\perp, \perp)$. κ is set to be the derived session key when $\delta = derived$.
- T_i^s records the transcript of messages sent and received by oracle Π_i^s. Initialized as $\emptyset$.
- $kst_i^s \in \{exposed, fresh\}$ denotes the freshness of the session key. Initialized as $fresh$.
- $st_k \in \{0,1\}^*$ is the session state after the session key/channel is established. Initialized as $\emptyset$.

To distinguish the different types of messages that may occur in an execution, there are three different execution "modes"of protocols: establishing a key, sending, respectively receiving messages from the established channel. Formally, the honest operation of a participant is defined by a triple $\Pi = (KeyExch, SendCh, ReceiveCh)$.

Some of the messages sent during the key-exchange may travel over the channel. So, strictly speaking, $KeyExch$ may make use of the latter algorithms. To facilitate the description of the resulting complex interaction we define the algorithm $EstChannel$ which, essentially , is in charge of establishing the channel. This algorithm may make calls to the algorithms defining Π.

During the execution of a protocol an oracle can receive two types of input, an application message (user input) or a channel message (received from the wire). At any point during its execution, protocol Π takes as input a message m and a message type $type \in \{ap, ch\}$ indicating the message was received from the user's application or the channel, respectively, runs the appropriate algorithm, and returns the output of that algorithm. The execution of protocol Π is summarized in Fig. 3.

After the channel has been established whenever the input message type is ch then $ReceiveCh$ will be called. This models messages that are received from the channel (for decryption). It takes as input a message m and state st_d and outputs a message m' for output to the user's application. $ReceiveCh$ rejects and outputs $\perp$ if the received messages are "out of state" messages (e.g., format error, invalid message).

When the message type is ap then $SendCh$ will be called. This models application messages that are input to be sent (encrypted) on the channel. It takes as input a message m and state st_e and outputs a message m' for output to the channel. Note that if keys have not yet been established ($\delta \neq accept$) then such a call to $SendCh$ will output $\perp$.

Protocol $\Pi(m, type)$:
if $type = ch \wedge \delta \neq accept$ **then**
$(m'; st_\kappa) \leftarrow EstChannel(m; st_\kappa)$
else if $type = ch$ **then**
$(m'; st_d) \leftarrow ReceiveCh_{\kappa_d}(m; st_d)$
else $(type = ap)$
$(m'; st_e) \leftarrow SendCh_{\kappa_e}(m; st_e)$
return m'

Fig. 3. Honest protocol execution

4.2 Matching Conversations

Denote $TEstCha_i^s$ and $TEstCha_j^t$ to be the transcript involved in the execution of $Estchannel$ for oracles Π_i^s and Π_j^t respectively.

Definition 6 (Matching Conversation). *We say that an oracle Π_i^s has a matching conversation to oracle Π_j^t, if $Prefix(TEstCha_j^t, TEstCha_i^s) = true$ or $Prefix(TEstCha_i^s, TEstCha_j^t) = true$.*

To keep the correctness of the protocol, two matching sessions which accept should always establish the same session key.

Definition 7 (Correctness). *For any two oracles Π_i^s and Π_j^t that have matching conversation with $pid_i^s = j$, $pid_j^t = i$, $\delta_i^s = accept$ and $\delta_j^t = accept$ it always holds that $\kappa_i^s = \kappa_j^t$.*

4.3 Adversarial Capabilities

The adversary $\mathcal{A}$ is a probabilistic polynomial Turing machine taking as input the security parameter λ and the public information, and controls the communication and network. $\mathcal{A}$ can issue the following queries to oracles Π_i^s for $i \in [1, n_C + n_T], s \in [1, n_S]$.

- $Newsession(i, \rho)$. Create a session for user i with role ρ.
- $Send(\Pi_i^s, m, type)$. Send a message m to Π_i^s with type *type*. As a result Π_i^s will run Π on input $(m, type)$ (as in Fig. 3) and respond with the outputting message m^* (if there is any) that should be sent according to the protocol specification and its internal states. The state information of Π_i^s will be updated depending on the protocol specification. After the session accepted, this query may initiate $ReceiveCh$ or $SendCh$ algorithms. Note that the session will not send message to the channel when it just invokes the $ReceiveCh$ algorithms.
- $StaticKeyReveal(i)$. $\mathcal{A}$ obtains the long-term private key of i if it is an authenticated entity.
- $EphemeralKeyReveal(\Pi_i^s)$. $\mathcal{A}$ obtains the ephemeral private key of session Π_i^s.
- $SessionKeyReveal(\Pi_i^s)$. $\mathcal{A}$ gets the derived session key of Π_i^s if δ is *derived* or *accepted*, at the same time, kst_i^s is set to be *exposed*. If at the point when this query is issued, there exists another oracle Π_j^t having matching conversation to Π_i^s, then kst_j^t is also set to be *exposed*.

Following the routine of [3], to define the security experiments for message authentication and privacy latter, the following notations for each Π_i^s is maintained:

- Application messages sent $Ap\text{-}S_i^s$, i.e. the list of all messages m input to $Send(\Pi_i^s, m, ap)$.
- Channel messages sent $Ch\text{-}S_i^s$, i.e. the list of all outputs from $Send(\Pi_i^s, m, ap)$.
- Channel messages received $Ch\text{-}R_i^s$, i.e. the list of all messages m input to $Send(\Pi_i^s, m, ch)$.
- Application messages received $Ap\text{-}R_i^s$, i.e. the list of all outputs from $Send$ (Π_i^s, m, ch).

Once a channel is established, whenever an application message is input to $Send$, the protocol Π is executed and a channel message will be output and sent on the channel. Similarly whenever a channel message is input to $Send$, the protocol Π is run and an application message will be output to the user. The above lists help us keep track of these messages and facilitate checking necessary in the following security models.

4.4 Security Definitions

In this subsection, we consider the security of one-way authentication secure channel establishment protocols, which is the scenario of EMV channel establishment protocol. The parties in the system are classified into two sets.

Let C be the set of authenticated participants (the Cards) and let T be the set of unauthenticated participants (the Terminals), where unauthenticated participants do not hold long-term private/public key pairs.

In our model, a Terminal $i \in T$ wishes to authenticate a Card $j \in C$ and establish a key (additionally a secure channel) with this Card. For a session Π_j^t owned by party $j \in C$ with public/secret key pairs, the adversary $\mathcal{A}$ needs to obtain both static and ephemeral secret keys to get the session key. But if the session has a matching conversation with another session Π_i^s with $i \in T$, knowing the session's ephemeral key enables the adversary to get the session key. Since all $i \in T$ have no long-term secret, it would always be possible for an adversary to impersonate an unauthenticated participant and establish a session with a real Card. So, in our model, the target session should always be a session owned by a Terminal.

We give the freshness definition for a session of a Terminal as follows. The session that the adversary attacks should keep fresh to make sense. In our model, the freshness definition enlarges the scope of sessions the adversary can attack compared to the previous security model [3]. That is, we allow the adversary to obtain either static key or ephemeral key of the Card party involved in the target session. While in [3], the adversary can ask neither keys of this party. So, the forward security property in one-way setting is captured in our model.

Definition 8 (One-Sided Freshness). *Let Π_i^s be an* accepted *session held by a party $i \in T$ with other party $j \in C$, and both parties are honest. Session Π_j^t (if it exists) is the matching conversation of Π_i^s. Then the session Π_i^s is said to be fresh if none of the following conditions hold:*

1. *Π_i^s has internal state $kst_i^s = exposed$.*
2. *Π_j^t exists and $\mathcal{A}$ issued one of the following:*
   - *EphemeralKeyReveal(Π_i^s).*
   - *Both StaticKeyReveal(j) and EphemeralKeyReveal(Π_j^t).*
3. *Π_j^t doesn't exist and $\mathcal{A}$ issued one of the following:*
   - *EphemeralKeyReveal(Π_i^s).*
   - *StaticKeyReveal(j) before session Π_i^s accepts.*

As in [3], we formulate three levels of security: Entity Authentication (EA), Message Authentication (MA) and Message Privacy (MP).

Entity Authentication. An adversary violates entity authentication if he can get a session to accept even if there is no unique session of its intended partner that has a matching conversation to it. More formally, the security is defined via an experiment *ent* played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. At the beginning, $\mathcal{C}$ generates the long-term key pairs (pk_U, sk_U) for all the parties $U \in C$ and sends the public keys pk_U for $U \in C$ to $\mathcal{A}$. Then the adversary can issue the oracle queries we defined above to the oracles.

Send query for *auth* game

$Send(\Pi_i^s, m, type)$:
$m' \leftarrow \Pi_i^s(m, type)$
if $\delta = accept$ and $type = ap$ then
  $AP\text{-}S_i^s \leftarrow AP\text{-}S_i^s || m$
else if $\delta = accept$ and $type = ch$ then
  if $m' \neq \perp$ then $AP\text{-}R_i^s \leftarrow AP\text{-}R_i^s || m'$
return m'

SendLR query for *priv* game

$SendLR(\Pi_i^s, m_0, m_1, type)$:
if $\delta = accept$ and $type = ap$ then
  $m' \leftarrow \Pi_i^s(m_{b_i^s}, ap)$
  $Ch\text{-}S_i^s \leftarrow Ch\text{-}S_i^s || m'$
else if $m_0 \neq m_1$ then $m' := \perp$
else
  $m' \leftarrow \Pi_i^s(m_0, type)$
  if $\delta = accept$ and $type = ch$ then
    if $m' \neq \perp$ and Π_i^s has a partner Π_j^t then
      $Ch\text{-}R_i^s \leftarrow Ch\text{-}R_i^s || m_0$
      if $Prefix(Ch\text{-}R_i^s, Ch\text{-}S_j^t) = true$ then $m' := \emptyset$
return m'

Fig. 4. The *Send* (resp. *SendLR*) query for the *auth* (resp. *priv*) games

Definition 9 (EA). *We say that protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ_{EA})-secure EA protocol if for all adversaries $\mathcal{A}$ running in time at most t, when $\mathcal{A}$ terminates, then with probability at most ϵ_{EA} there exists a fresh oracle Π_i^s such that Π_i^s accepts, but there is no unique oracle Π_j^t such that Π_i^s has a matching conversation to Π_j^t for $i \in T, j \in C$.*

Message Authentication. The message authentication property ensures the integrity and authenticity of all messages sent over the channel. For any two partner oracles Π_i^s and Π_j^t, the oracle Π_i^s should only successfully receive messages which were output by Π_j^t and vice versa. That is formalized by requiring that for any fresh oracle Π_i^s with unique partner Π_j^t, $Prefix(Ap\text{-}R_i^s, Ap\text{-}S_j^t) = true$. If this does not hold then the adversary successfully fools Π_i^s into receiving an application message which was not output by the partnered oracle Π_j^t.

The authentication experiment *auth* generates public/private key pairs for each user $i \in C$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$. The adversary is permitted to make the queries $NewSession(i, \rho)$, $SessionKeyReveal(\Pi_i^s)$, $StaticKeyReveal(i)$, $EphemeralKey\ Reveal(\Pi_i^s)$ as well as $Send(\Pi_i^s, m, type)$ with message $type \in \{ap, ch\}$. On querying $Send(\Pi_i^s, m, type)$, the game behaves as in Fig. 4.

The game $Exec_{\Pi}^{auth}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$ is defined as follows:

1. The challenger $\mathcal{C}$ generates public/private key pairs for each user $U \in C$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to make as many $NewSession$, $SessionKeyReveal$, $StaticKey\ Reveal$, $EphemeralKeyReveal$, $Send$ queries as it likes.
3. The adversary stops with no output.

We say that an adversary $\mathcal{A}$ wins the game if there exists Π_i^s with unique partner Π_j^t such that they are matching conversations and the list $Ap\text{-}R_i^s$ is not a prefix of $Ap\text{-}S_j^t$. The adversary's advantage is $Adv_{\Pi}^{auth}(\mathcal{A}) = Pr[\exists \Pi_i^s, \Pi_j^t$ for $i \in T, j \in C : \Pi_i^s$ is fresh $\wedge \Pi_i^s, \Pi_j^t$ are matching $\wedge Prefix(Ap\text{-}R_i^s, Ap\text{-}S_j^t) = false]$.

Definition 10 (MA). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ_{MA})-secure MA protocol if for all adversaries $\mathcal{A}_{auth}$ running in time at most t, $Adv_{\Pi}^{auth}(\mathcal{A}_{auth}) \leq \epsilon_{MA}$.*

Message Privacy. The message privacy property ensures that the adversary should not be able to determine which set of messages $\{m_{01}, m_{02}, m_{03}, \cdots\}$ and $\{m_{11}, m_{12}, m_{13}, \cdots\}$ has been transmitted on the secure channel.

The message privacy experiment *priv* initializes the states as in the authentication experiment *auth*, except that each session now also holds a random secret bit b_i^s. As before, the adversary can make the queries $NewSession$, $SessionKeyReveal$, $StaticKeyReveal$, $EphemeralKeyReveal$. In addition, the adversary can issue a left-right version of $Send(\Pi_i^s, m, type)$ which is used to model message privacy. Specifically, the query $SendLR(\Pi_i^s, m_0, m_1, type)$ takes as input two messages and returns $Send(\Pi_i^s, m_{b_i^s}, type)$. When $type \neq ap$ these two messages are equal, $SendLR(\Pi_i^s, m, m, type) = Send(\Pi_i^s, m, type)$.

As before, two sessions are matching conversations. On the $SendLR(\Pi_i^s, m_0, m_1, type)$ query, the game behaves as in Fig. 4. Once the channel is established, whenever $SendLR(\Pi_i^s, m, m, ch)$ is called, we allow the protocol to run as normal but check the lists $Ch\text{-}R_i^s$ and $Ch\text{-}S_j^t$. If the message m was a channel output from Π_i^s's partner Π_j^t, then $SendLR$ will not return anything. This allows the adversary to progress the state of an oracle but prevents them from trivially winning the game.

Game $Exec_{\Pi}^{priv}(\mathcal{A})$ between an adversary $\mathcal{A}$ and challenger $\mathcal{C}$:

1. The challenger $\mathcal{C}$, generates public/private key pairs for each user $U \in C$ (by running $\mathcal{G}$) and returns the public keys to $\mathcal{A}$.
2. $\mathcal{A}$ is allowed to make as many $NewSession$, $SessionKeyReveal$, $StaticKeyReveal$, $EphemeralKeyReveal$, $SendLR$ queries as it likes.
3. Finally $\mathcal{A}$ outputs a tuple (i, s, b_0) for $i \in T$.

We say the adversary $\mathcal{A}$ wins if its output $b_0 = b_i^s$ and Π_i^s is fresh (and has a unique partner) and the output of $Exec_{\Pi}^{priv}(\mathcal{A})$ is set to 1. Otherwise the output is 0. Formally we define the advantage of $\mathcal{A}$ as $Adv_{\Pi}^{priv}(\mathcal{A}) = |Pr[Exec_{\Pi}^{priv}(\mathcal{A}) = 1] - 1/2|$.

Definition 11 (MP). *A protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ_{MP})-secure MP protocol if for all adversaries $\mathcal{A}_{priv}$ running in time at most t, $Adv_{\Pi}^{priv}(\mathcal{A}_{priv}) \leq \epsilon_{MP}$.*

A channel establishment protocol is secure if it satisfies all of the three notions above.

Definition 12 (eEAMAP). *Protocol $P = \{\Pi, \mathcal{G}\}$ is a (t, ϵ)-secure eEAMAP protocol if it is a (t, ϵ)-secure EA protocol, a (t, ϵ)-secure MA protocol and a (t, ϵ)-secure MP protocol.*

4.5 Unlinkability

In practice, the Card holders may also want to have a property that their two independent transactions can not be linked. Actually, this property is formally captured by a notion called unlinkability. In this paper, we adapt the idea of [3] to define our unlinkability definition (see full version of our paper), which means that it should be hard for an adversary to determine whether two particular sessions are linked with the same Card. Note that this property only holds against an eavesdropper adversary who is not a Terminal.

5 Security Analysis of EMV Channel Establishment Protocol in Our Security Model

In our security model, the adversary controls all the communications and can get the ephemeral keys and session keys of sessions, so he can impersonate a valid Card through the following steps (see Fig. 5):

1. Card entity C chooses $a \in_R \mathbb{F}_q$ and sends out $A = aQ_C$.
2. The adversary $\mathcal{M}$ intercepts the message A, computes $2 \cdot A$ and sends it to Terminal T.
3. The Terminal T selects $e \in_R \mathbb{F}_q$ and sends out $E = eP$.
4. $\mathcal{M}$ intercepts E, computes $2 \cdot E$ and sends it to the Card C.
5. After that, $\mathcal{M}$ issues $EphemeralKeyReveal$ to the Card session and obtains a.
6. After the Card session accepts, $\mathcal{M}$ issues $SessionKeyReveal$ query to the Card's session and obtains its session key $(\kappa_e^C, \kappa_d^C) = H(2adE) = H(2deaP) = (\kappa_d^T, \kappa_e^T)$.
7. $\mathcal{M}$ can obtain $Q_C, cert_C$ by impersonating a Terminal to C in a different session.[1]
8. $\mathcal{M}$ computes and sends $ch' = SendCh_{\kappa_e^C}(cert_C||2a||Q_C; st_e^C)$ which will pass the verification of the Terminal.

So, the adversary successfully impersonates the Card to the Terminal which breaks the EA property.

6 The Enhanced Protocol

The enhanced protocol is presented as follows in Fig. 6. The only difference lies in the computation of the session key. We add the ephemeral public keys of the session to the inputs of the hash function.

[1] Note that in this process the adversary can also make no $EphemeralKeyReveal$ queries and just keep and decrypt value ch using κ_e^C to obtain $(cert_C||a||Q_C)$ and extract the value of $cert_C, a, Q_C$.

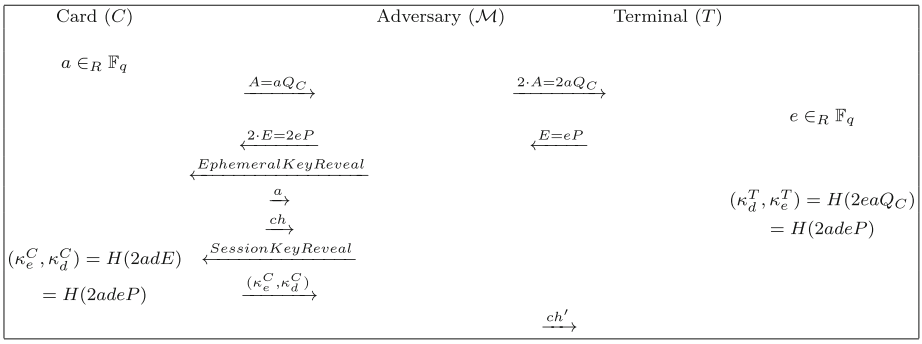

Fig. 5. Security analysis of ECC-based EMV channel establishment protocol in the enhanced security model

7 Security

Theorem 1. *If the Gap-DH problem is* $(\boldsymbol{t}, \epsilon_{Gap\text{-}DH})$ *hard over* $\mathbb{G}$*,* $AE = (enc, dec)$ *is* $(\boldsymbol{t}, \epsilon_{ind\text{-}sfcca})$ *IND-sfCCA secure and* $(\boldsymbol{t}, \epsilon_{int\text{-}sfptxt})$ *INT-sfPTXT secure, and the signature scheme* (sig, ver) *used to produce card certificates is* $(\boldsymbol{t}, \epsilon_{sig})$ *EUF-CMA secure, then the Enhanced EMV protocol* $P = (\Pi, \mathcal{G})$ *in Fig. 6 is secure in the sense of eEAMAP and unlinkability.*

The proof of the theorem is given in the full version of the paper.

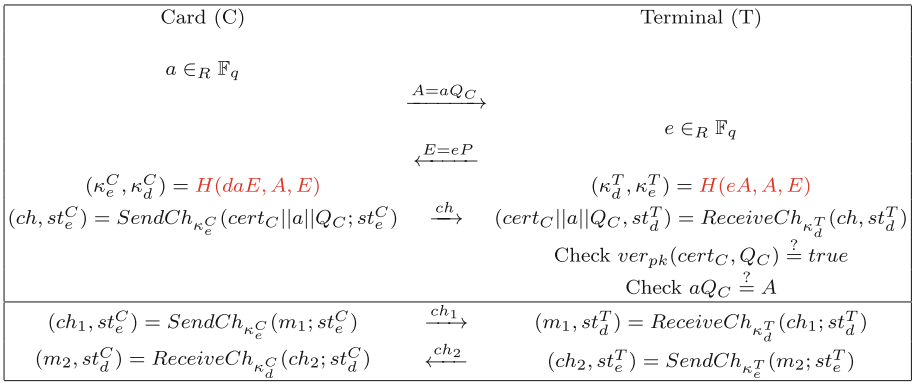

Fig. 6. Enhanced ECC-based EMV channel establishment protocol

Acknowledgement. The work is supported by the National Basic Research Program of China (No. 2013CB338003), the National Natural Science Foundation of China (No. 61170278, 91118006), and the 863 project (No. 2012AA01A403).

References

1. Anderson, R., Bond, M., Choudary, O., Murdoch, S.J., Stajano, F.: Might financial cryptography kill financial innovation? – the curious case of EMV. In: Danezis, G. (ed.) FC 2011. LNCS, vol. 7035, pp. 220–234. Springer, Heidelberg (2012)
2. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (1994)
3. Brzuska, C., Smart, N.P., Warinschi, B., Watson, G.J.: An analysis of the EMV channel establishment protocol. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, pp. 373–386. ACM, New York (2013)
4. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 453. Springer, Heidelberg (2001)
5. Coron, J.-S., Naccache, D., Tibouchi, M.: Fault attacks against EMV signatures. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 208–220. Springer, Heidelberg (2010)
6. Coron, J.-S., Naccache, D., Tibouchi, M., Weinmann, R.-P.: Practical cryptanalysis of ISO/IEC 9796-2 and EMV signatures. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 428–444. Springer, Heidelberg (2009)
7. Degabriele, J.P., Lehmann, A., Paterson, K.G., Smart, N.P., Strefler, M.: On the joint security of encryption and signature in EMV. In: Dunkelman, O. (ed.) CT-RSA 2012. LNCS, vol. 7178, pp. 116–135. Springer, Heidelberg (2012)
8. Drimer, S., Murdoch, S.J., Anderson, R.: Optimised to fail: card readers for online banking. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 184–200. Springer, Heidelberg (2009)
9. EMVCo: EMV-Integrated Circuit Card Specifications for Payment Systems, Book 1: Application Independent ICC to Terminal Interface Requirements (2011)
10. EMVCo: EMV-Integrated Circuit Card Specifications for Payment Systems, Book 2: Security and Key Management (2011)
11. EMVCo: EMV-Integrated Circuit Card Specifications for Payment Systems, Book 3: Application Specification (2011)
12. EMVCo: EMV-Integrated Circuit Card Specifications for Payment Systems, Book 4: Cardholder, Attendant, and Acquirer Interface Requirements (2011)
13. EMVCo: EMV ECC Key Establishment Protocols (2012)
14. Fujioka, A., Suzuki, K.: Designing efficient authenticated key exchange resilient to leakage of ephemeral secret keys. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 121–141. Springer, Heidelberg (2011)
15. Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 467–484. Springer, Heidelberg (2012)
16. Giesen, F., Kohlar, F., Stebila, D.: On the security of TLS renegotiation. In Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS 2013, pp. 387–398. ACM, New York (2013)
17. Huang, H.: Strongly secure one round authenticated key exchange protocol with perfect forward security. In: Boyen, X., Chen, X. (eds.) ProvSec 2011. LNCS, vol. 6980, pp. 389–397. Springer, Heidelberg (2011)

18. Jager, T., Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DHE in the standard model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 273–293. Springer, Heidelberg (2012)
19. Kohlar, F., Schäge, S., Schwenk, J.: On the security of TLS-DH and TLS-RSA in the standard model. Cryptology ePrint Archive, Report 2013/367 (2013). http://eprint.iacr.org/
20. Krawczyk, H.: HMQV: a high-performance secure diffie-hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (2005)
21. Krawczyk, H., Paterson, K.G., Wee, H.: On the security of the TLS protocol: a systematic analysis. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 429–448. Springer, Heidelberg (2013)
22. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Heidelberg (2007)
23. Li, Y., Schäge, S., Yang, Z., Kohlar, F., Schwenk, J.: On the security of the pre-shared key ciphersuites of TLS. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 669–684. Springer, Heidelberg (2014)
24. EMVCo LLC: EMV deployment statistics (2012). http://www.emvco.com/about_emvco.aspx?id=202
25. Murdoch, S., Drimer, S., Anderson, R., Bond, M.: Chip and pin is broken. In: 2010 IEEE Symposium on Security and Privacy (SP), pp. 433–446, May 2010
26. Ogundele, O., Zavarsky, P., Ruhl, R., Lindskog, D.: The implementation of a full EMV smartcard for a point-of-sale transaction. In: 2012 World Congress on Internet Security (WorldCIS), pp. 28–35, June 2012
27. Ustaoglu, B.: Obtaining a secure and efficient key agreement protocol from (H)MQV and NAXOS. Des. Codes Crypt. **46**(3), 329–342 (2008)
28. Van Herreweghen, E., Wille, U.: Risks and potentials of using EMV for internet payments. In: Proceedings of the USENIX Workshop on Smartcard Technology on USENIX Workshop on Smartcard Technology, WOST 1999, p. 18. USENIX Association, Berkeley (1999)
29. Yang, Z.: Efficient eCK-secure authenticated key exchange protocols in the standard model. In: Qing, S., Zhou, J., Liu, D. (eds.) ICICS 2013. LNCS, vol. 8233, pp. 185–193. Springer, Heidelberg (2013)