Secure Contactless Payment

Handan Kılınç and Serge Vaudenay

EPFL, Lausanne, Switzerland

Abstract. A contactless payment lets a card holder execute payment without any interaction (e.g., entering PIN or signing) between the terminal and the card holder. Even though the security is the first priority in a payment system, the formal security model of contactless payment does not exist. Therefore, in this paper, we design an adversarial model and define formally the contactless-payment security against malicious cards and malicious terminals including relay attacks. Accordingly, we design a contactless-payment protocol and show its security in our security model. At the end, we analyze EMV-contactless which is a commonly used specification by most of the mobile contactless-payment systems and credit cards in Europe. We find that it is not secure against malicious cards. We also prove its security against malicious terminals in our model. This type of cryptographic proof has not been done before for the EMV specification.

1 Introduction

A contactless payment (CP) system is a payment method using a card or a device, that allows a user to pay at a point of sale by holding the card/device near a contactless terminal. There are two main ways of performing a contactless transaction: with a card or with a smartphone.

CP technologies advanced quickly in recent years. Therefore, the CP market size is expected to grow from USD 6.70 Billion in 2016 to USD 17.56 Billion by 2021 [1]. One of the reasons of this development is based on the convenience of the payment process (e.g., users do not need to type a PIN code (or sign a bill) and wait for the verification process of the PIN). The first CP was implemented in 1995 by Seoul Bus Transport and since then many leading companies (Apple, Google, Samsung) started to integrate a CP process into smartphones. The first (contactless) payment system launched by a leading company is Google Wallet in 2011. Then, Apple Pay and Samsung Pay followed suit in 2014 and 2015, respectively. Also in 2015, Google announced a new contactless system, Android Pay. Classic CP systems use cards. A majority of them now follow EMV contactless specifications, written by EMVCo [3], a consortium created by payment companies, like Visa and Mastercard. The USA has migrated from old magnetic reader terminals to new EMV compliant ones, already used in Europe.

Despite the big developments in this technology, we realize that some important functionalities such as secure processing of payments have not been considered formally. No standard security model was provided for CP. Some pre-play attacks were detected for EMV because of poor random generation [8, 7]. Roland

and Langer [23] discovered a cloning attack for EMV contactless payment cards since the contactless payment permits an attacker to learn the necessary data for cloning. The cloned cards can then be used to perform EMV Mag-Stripe transactions at any EMV contactless payment terminal. Another type of pre-play attack [8] was discovered which relies on the fact that EMV do not impose any encryption between merchant and acquirer, or between acquirer and issuer.

The most important attack specific for EMV-contactless (and also most of the contactless applications) is relay attack which has shown up for a while ago [22, 28, 19, 17, 18]. A relay attack in an EMV-contactless payment can be run as follows: the man-in-the-middle (MiM) adversary makes payment by relaying messages from a card to a terminal and vice versa, while terminal and the card think that they communicate with each other. Chothia et al. [14] remark that the first version of EMVco is vulnerable to relay attacks and provide a solution for this. The current EMVco [3], therefore, take precaution partly against relay attacks using the solution proposed by Chothia et al. [14]. It is "partly" because the solution they use is software based where the terminal does not require a specific hardware. So, it protects against relatively trivial adversaries but does not protect against the adversaries using a sophisticated hardware [18, 15]. To defend this level of security that they provide against relay attacks, Chothia et al. [14] say that "Considering that contactless payments are limited to small amounts, the cost of the hardware would be a disincentive for criminals". However, limiting to small amounts does not necessarily mean that the relay attack outcome will be also a small amount. An attacker in a crowded area (e.g., metro, concert, museum) can execute many numbers of relay attacks and increase its outcome. In addition, some cards are limited to some small amounts in their issued country currency, but when they are abroad this limit is removed because the conversion from the issued country currency to currency in the current country cannot be computed. Besides this, the solution provided by Chothia et al. [14] for EMV-contactless does not protect against malicious cards who can execute relay attacks in a different way than MiM-adversaries such as:
Distance Fraud (DF): A malicious far-away card tries to prove that he is close enough to the terminal to make the verifier accept the payment.
Distance Hijacking (DH) [16]: A far-away malicious prover takes advantage of some honest and active provers who are close to the verifier to make the verifier grant privileges to the far-away prover.

Preventing against DF and DH in payment protocols is important as well. For example, a DF or DH attack can be harmful to a bank in the following case: A credit card holder makes a payment while he is far-away from a POS machine. Then, he asks for a reimbursement of the payment from his bank by claiming that he did not make the payment and he was probably exposed to relay attack or cloning attack. While doing this, he can prove that he was not at the place where the payment has been executed (e.g., showing that he was in another city).

The most promising solution against MiM, DH or DF is distance bounding (DB) [11]. In DB, a verifier determines the distance of a prover who wants to

authenticate. If the distance of a prover is close enough, the verifier will be sure of the nonexistence of relay attack during the protocol execution. Apparently, it is necessary to utilize a secure distance bounding [20, 26, 25, 27, 12, 6, 24, 9, 10] in contactless payment.

Our Contributions: Considering all these attacks and the missing formalism, we design a new security model for CP protocols and design a secure contactless-payment protocol. In more detail, our contributions are as follows:

- We formally define CP between parties: an issuer, a terminal, a card. Then, we give two security definitions for malicious cards and for malicious terminals in the adversarial and communication model that we define.
- We construct a secure CP protocol (ClessPay) against malicious cards and malicious terminals. ClessPay uses a distance bounding protocol to protect against relay attacks by malicious cards and MiM-adversaries. We proved formally the security of ClessPay in our security model.
- We analyze EMV-contactless protocol in our model. We give some vulnerabilities on this against malicious cards. We prove the security of EMV-contactless protocol against malicious terminal formally. This type of formal cryptographic analysis is the first for EMV-contactless protocol.

2 Definitions

2.1 Contactless Payment

According to the EMV specifications [2], a (contactless) payment system consists of a card holder, a merchant, an acquirer, an issuer, a payment system, a card and a terminal. Our definitions do not include certification by the payment system, communication between merchant-acquirer and terminal-acquirer. We assume that the setup between payment components has been established. For the sake of simplicity, we assume the terminal represents both the terminal and the acquirer in the payment system and all cards are issued by one issuer.

The Issuer: It issues a personalized card to the card holder. The cards may contact with the issuer during the payment process (in online transactions) for the verification of the payment data. It also gives reimbursements of completed transactions to the acquirer. Each issuer has its policy function Policy to approve or disapprove a transaction. We assume that the issuer has a database $DataB$ which stores the card information. $DataB$ consists of tuples (Public Key, Card Information) of each card. Card information (CI) may consist of transaction list, the balance or the card limit.

Cards: They have a technology (e.g. NFC, Bluetooth) to communicate with a payment terminal without any contact. In CP, cards are the components which interact with a payment terminal to execute a payment with a certain amount. They include a unique card number. They also store a secret/public key pair in their tamper-resistant module and the issuer's public key. In this paper, we exclude card numbers in our definitions for simplicity. In our definitions, cards are identified with their public keys.

Terminals: Terminals interact with both cards and their issuers via acquirers.

They receive an order of payment from a card and validate the payment together with the issuer of the card.

Definition 1 (Contactless Payment (CP)). *A CP consists of algorithms for cards, terminals and issuers. They respectively run the algorithms* $C(\mathsf{sk}_C, \mathsf{pk}_C, \mathsf{pk}_I)$, $T(\mathsf{pk}_I, \tau_T)$ *and* $I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)$. *Here,* $(\mathsf{sk}_C, \mathsf{pk}_C)$ *and* $(\mathsf{sk}_I, \mathsf{pk}_I)$ *are the secret/public key pair of* C *and* I, *respectively. They are generated by the algorithms* $\mathcal{G}_C(1^n)$ *and* $\mathcal{G}_I(1^n)$ *where* n *is a security parameter.* $DataB$ *is the database for cards' information.* I *includes a subroutine* $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ *where* CI *represents the card information of a card with* pk_C. *In the end,* I *outputs* $\mathsf{Out}_I \in \{0,1\}$ *and privately outputs* $\mathsf{POut}_I = (\mathsf{pk}_C, id_I, \tau_I)$. *Similarly,* T *outputs* $\mathsf{Out}_T \in \{0,1\}$[1] *and private output* $\mathsf{POut}_T = (\mathsf{pk}_C, id_T, \tau_T)$ *and* C *privately outputs* $\mathsf{POut}_C = (id_C, \tau_C)$. *Here,* τ *is the transaction (*τ_T, τ_I *and* τ_C *are the values seen by the terminal , the issuer and the card), id is the identifier of the transaction (*id_T, id_I *and* id_C *are similarly defined) and* $\phi \in \{0,1\}$ *shows the approval or disapproval of the transaction.*

The algorithm Policy depends on the policy of the transaction approval by the issuer. So, we can consider it as an algorithm which decides if a transaction τ_I is possible for the card with pk_C and CI.

We note that Out_I and $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ can be different. Out_I (similarly Out_T) shows the result of the CP which can be either accepting or canceling the payment. However, $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I)$ shows only if the card with pk_C is able to do the payment. For example, even though the payment is canceled ($\mathsf{Out}_I = 0$) by the issuer, the issuer can approve the payment ($\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_I) = 1$). It means that the card is able to to this payment but the payment process is canceled (e.g., because of malicious behaviors).

Definition 2 (Correctness of CP). *A contactless payment is correct for all* B, *transactions* τ, *database* $DataB$, CI, *and generated key pairs* $(\mathsf{sk}_C, \mathsf{pk}_C)$ *and* $(\mathsf{sk}_I, \mathsf{pk}_I)$ *if*

- *the algorithms* C, T *and* I *are run,*
- T *starts a transaction* τ,
- *there exists a* C *whose distance from* T *is at most* B,
- (pk_C, CI) *is in* $DataB$ *of an issuer* I,

then there exists id such that probability of $(\mathsf{Out}_T = \mathsf{Out}_I = \mathsf{Policy}(\mathsf{pk}_C, CI, \tau)) \wedge (\mathsf{POut}_T = \mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)) \wedge (\mathsf{POut}_C = (id, \tau))$ *is 1.*

The output of T has to depend on the output of I because actually I is in the position to decide if the transaction is possible with the card (in fact an honest card cannot know if the transaction is possible).

Adversarial and Communication Model: In contactless payment, we consider the similar adversarial and communication model with the access control (AC) security model by Kılınç and Vaudenay [21]. The parties in AC: a controller, a

[1] $\mathsf{Out}_I = 0$ or $\mathsf{Out}_T = 0$ mean canceling and $\mathsf{Out}_I = 1$ or $\mathsf{Out}_T = 1$ mean accepting.

reader, a tag correspond to the parties contactless payment: an issuer, a terminal, a card, respectively. Differently than AC, in the contactless-payment adversarial model, terminals can be malicious. In a nutshell, the model is as follows:

- The communication between T and I is secure and authenticated. The adversary cannot attack this part of the communication.
- The communication between the parties is limited by the speed of light.
- All parties have polynomially many instances. An instance of a party is an execution of its corresponding algorithm at a given location. Instances of honest parties cannot be run in parallel.
- The adversaries can change the location of honest instances (but they move at a limited speed) or can activate them (See [21] for details).
- Adversaries can create the database.
- Adversaries can change the destination of messages.

Definition 3 (Security in Contactless Payment with Malicious Cards). *The security game is as follows:*

- *Run* $\mathcal{G}_I(1^n) \rightarrow (\mathsf{sk}_I, \mathsf{pk}_I)$ *and* $\mathcal{G}_C(1^n) \rightarrow (\mathsf{sk}_{C_i}, \mathsf{pk}_{C_i})$ *for the issuer and each card* C_i *and give the public keys to the adversary.*
- *The adversary creates instances of cards* (C_i*'s) and the terminals at some locations of his choice. There is a distinguished terminal* T *(*T *is honest).*
- *The adversary sets a database* $DataB$ *of the issuer. The issuer instance* I *which communicates with* T *is the distinguished issuer.*
- *The adversary creates the instances of himself (malicious cards or terminals) which can run independently and communicate together.*

We denote $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$ *and* $\mathsf{POut}_T = (\mathsf{pk}''_C, id_T, \tau_T)$ *the private outputs of* I *and* T*. Following our communication model, the game ends when* T *outputs* Out_T*. A contactless payment is secure, if the adversary wins this game with negligible probability. The adversary wins the game if* $\mathsf{Out}_T = 1$ *and at least one of the following conditions are satisfied:*

1. $(\mathsf{pk}'_C, .) \notin DataB$,
2. $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and the distance between any* C *holding* pk *and* T *is more than* B *during the execution of the protocol with* id_T,
3. $\mathsf{pk}'_C \notin \{\mathsf{pk}_{C_i}\}$ *and no instance of the adversary is close to* T *during the execution of the contactless payment protocol with* T *and* I.
4. $(\mathsf{pk}'_C, id_I, \tau_I) \neq (\mathsf{pk}'_C, id_T, \tau_T)$,
5. $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ *and there exists no card with* pk'_C *and* $\mathsf{POut}_C = (id_I, \tau_I)$.

Remarks: The first winning condition shows that a card which does not belong $DataB$ should not authenticate. The second and the third conditions are to protect against MiM and DH (DF as well), respectively. Finally, the last two conditions are to be sure that the transaction that I and T approve and complete, and the transaction that I and an honest C approve and complete are the same.

Definition 4 (Security in Contactless Payment with Malicious Terminals). *The security game is as follows:*

- *Run the key generation algorithms* $\mathcal{G}_I(1^n) \rightarrow (\mathsf{sk}_I, \mathsf{pk}_I)$ *and* $\mathcal{G}_C(1^n) \rightarrow (\mathsf{sk}_{C_i}, \mathsf{pk}_{C_i})$ *for the issuer* I *and each card* C_i *and give away public keys.*

– *The adversary creates instances of C_i and the terminals at some locations of his choice. There is a distinguished instance I.*
– *The adversary sets a database $DataB$.*
– *The adversary creates the instances of himself which can run independently and communicate together (as malicious cards or malicious terminals).*

At the end of the game I outputs Out_I and $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$. A contactless payment is secure, if the adversary wins this game with negligible probability. The adversary wins the game:

1. *if $\mathsf{Out}_I = 1$ and if at least one of the following conditions are satisfied:*
   (a) $(\mathsf{pk}'_C, .) \notin DataB$,
   (b) $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and there exists no card with pk'_C which outputs (id_I, τ_I),
   (c) $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and the instance of this card with pk'_C having (id_I, τ_I) has distance from the adversary and any honest terminal more than B.
2. *or if there exists an honest card instance with $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$ which privately outputs $\mathsf{POut}_C = (id_C, \tau_C)$ and there exists an issuer instance which has $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau_C) = 0$ and id_C.*

The proximity condition ($1c$) has not been considered by any of the payment systems before. Actually, even though we make sure that the payment can be approved only when the terminal is close to the card, we still cannot prevent a malicious terminal to execute a payment unbeknown to a card holder. For example, a malicious terminal can be moved close to a card while the card is not at the shop. This means $1c$ does not prevent the malicious intention of the terminals. If we can be sure that the terminals can be run in a certain location, then we can guarantee the security against malicious terminals with the proximity condition. This can be possible by using position-based cryptography [13], but current terminals do not support this. Therefore, in our protocol, we eliminate $1c$. We call **almost-secure against malicious terminals** if a protocol is secure without the condition $1c$ in Definition 4. The condition 2 is to prevent honest cards to make payment even though the issuer does not approve it. For example, this condition prevents attacks where malicious terminals make a card pay (maybe without the knowledge of the honest card) for a big amount of money where normally the issuer would not let this amount of payment.

2.2 Preliminaries about Public Key Distance Bounding

We give security definitions (MiM, DF, DH) of public-key distance bounding. In CP, the terminal represents the verifier in DB because the issuer is not at the position to determine the distance of cards and the card represents the prover.

Definition 5 (Public key DB Protocol [26, 20]). *A public key DB protocol is a two-party probabilistic polynomial-time (PPT) protocol and it consists of a tuple $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$. Here, $\mathcal{K}_P$ is the key generation algorithm of the prover algorithm P and outputs secret/public key pair $(\mathsf{sk}_P, \mathsf{pk}_P)$. $\mathcal{K}_V$ is the key generation algorithm of the verifier algorithm V and outputs secret/public key pair $(\mathsf{sk}_V, \mathsf{pk}_V)$. B is the distance bound. $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ and $V(\mathsf{sk}_V, \mathsf{pk}_V)$ are interactive algorithms. At the end of the protocol, $V(\mathsf{sk}_V, \mathsf{pk}_V)$ outputs Out_V and*

privately outputs $\mathsf{POut}_V = \mathsf{pk}_P$. *If* $\mathsf{Out}_V = 1$, *then* V *accepts. If* $\mathsf{Out}_V = 0$, *then* V *rejects. A public-key DB protocol is correct if and only if under honest execution, whenever a verifier* V *and a close (to* V*) prover* P *run the distance bounding protocol, then* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* $\mathsf{POut}_V = \mathsf{pk}_P$.

We use the same adversarial and communication model as in contactless-payment where the provers are cards and the verifiers are terminals.

Definition 6 (MiM security [26]). *The game begins by running the key generations algorithms* $\mathcal{K}_V$ *and* $\mathcal{K}_P$. *They output* $(\mathsf{sk}_V, \mathsf{pk}_V)$ *and* $(\mathsf{sk}_P, \mathsf{pk}_P)$, *respectively. The public keys* pk_V *and* pk_P *are given to the adversary. In the game, we have polynomially many verifier instances where one of them is the distinguished one* $\mathcal{V}$ *and polynomially many honest prover instances which are far away from* $\mathcal{V}$. *The adversary with its instances can be at any location. The adversary wins if* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* $\mathsf{POut}_V = \mathsf{pk}_P$. *A DB protocol is MiM-secure if for any such game, the probability of an adversary to win is negligible.*

Definition 7 (Distance fraud [26]). *The game begins by running the key generation algorithm* $\mathcal{K}_V$. *It outputs* $(\mathsf{sk}_V, \mathsf{pk}_V)$. *The public key* pk_V *is given to the adversary. The adversary generates its secret/public key pair* $(\mathsf{sk}_P, \mathsf{pk}_P)$ *with using an arbitrary algorithm* $\mathcal{K}_P^*$. *In the game, we have polynomially many verifier instances including the distinguished one* $\mathcal{V}$ *and instances of an adversary (prover instances). The adversary wins if* $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* $\mathsf{POut}_V = \mathsf{pk}_P$ *when there is no close party to* $\mathcal{V}$. *A DB protocol is DF-secure, if for any such game, the adversary wins with negligible probability.*

Definition 8 (Distance hijacking [26]). *The game includes polynomially many verifier instances* $\mathcal{V}, V_1, V_2, ...$, *a far away adversary* P, *and honest prover instances* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2$.... *In this game, we consider a DB protocol* $(\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ *with phases: initialization, a challenge and a verification phases. A DB protocol is DH-secure if for all PPT algorithms* $\mathcal{K}_P^*$ *and* $\mathcal{A}$, *the probability of* P *to win the following game is negligible.*

- *The game runs* $\mathcal{K}_V \rightarrow (\mathsf{sk}_V, \mathsf{pk}_V)$ *for the verifier and* $\mathcal{K}_{P'} \rightarrow (\mathsf{sk}_{P'}, \mathsf{pk}_{P'})$ *for the honest prover.*
- *The adversary runs* $\mathcal{K}_P^*(\mathsf{pk}_{P'}, \mathsf{pk}_V) \rightarrow (\mathsf{sk}_P, \mathsf{pk}_P)$.
- *The game aborts, if* $\mathsf{pk}_P = \mathsf{pk}_{P'}$. *Otherwise, instances of* P *run the adversarial algorithm* $\mathcal{A}$, *the honest prover instances* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *run* $P(\mathsf{sk}_{p'}, \mathsf{pk}_V)$, *the verifier instances* $\mathcal{V}, V_1, V_2, ...$ *run* $V(\mathsf{sk}_V, \mathsf{pk}_V)$.
- P *interacts with* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *and* $\mathcal{V}, V_1, V_2, ...$ *during the initialization phase of* $\mathcal{V}$ *and* P' *concurrently.*
- P' *and* $\mathcal{V}$ *continue interacting with each other in their challenge phase and* P *remains passive but it sees the exchanged messages.*
- P *interacts with* $\mathsf{P}', \mathsf{P}'_1, \mathsf{P}'_2, ...$ *and* $\mathcal{V}, V_1, V_2, ...$ *in the verification phase.*

The adversary wins if $\mathcal{V}$ *outputs* $\mathsf{Out}_V = 1$ *and* $\mathsf{POut}_V = \mathsf{pk}_P$.

The initialization and verification phase do not have any specific definition but the challenge phase corresponds to the phase where the challenge/response

exchanges occur. It is the time critical phase meaning that the verifier determines the proximity of the responses by checking the response time (i.e., If the responses arrived on time, the prover is accepted. Otherwise, it is rejected.).

3 Contactless Payment Protocol

In this section, we construct a secure CP protocol from a public-key distance bounding $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$, an encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and a signature scheme $(\mathsf{Sign}, \mathsf{Verify})$.

3.1 ClessPay

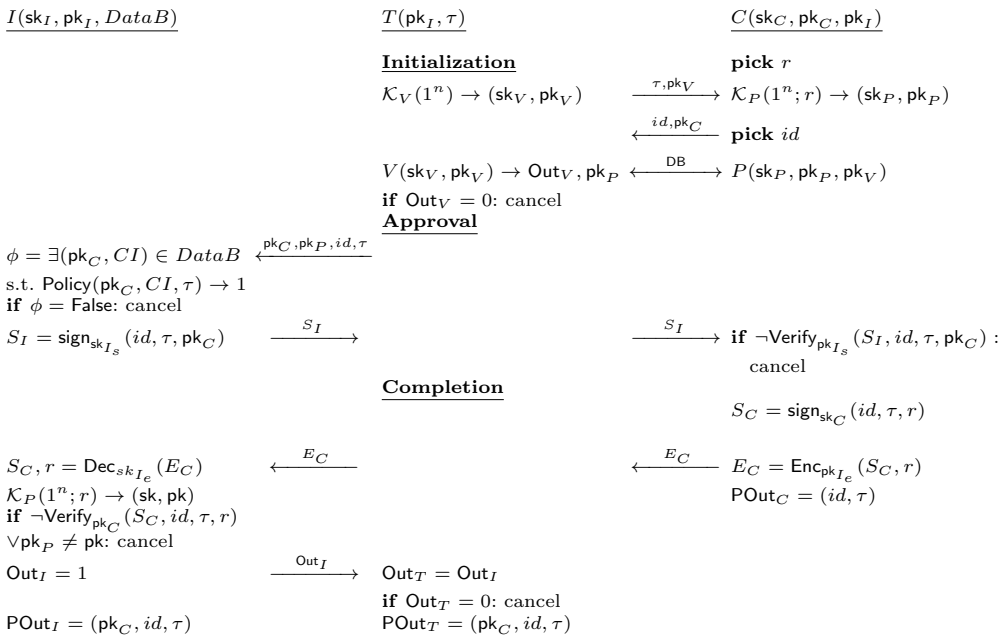

Fig. 1. The ClessPay Protocol.

The protocol ClessPay (See Figure 1) starts after the terminal T creates a transaction τ and connects with a card C. We do not give the details of τ since it depends on the payment system.

In our protocol, we use signature schemes and an encryption scheme. Therefore, some secret/public key pairs are generated by using their key generation algorithms. More specifically, the key generation algorithm $\mathcal{G}_I$ generates a secret/public key pair $(\mathsf{sk}_I, \mathsf{pk}_I) = ((\mathsf{sk}_{I_s}, \mathsf{sk}_{I_e}), (\mathsf{pk}_{I_s}, \mathsf{pk}_{I_e}))$ where $(\mathsf{sk}_{I_s}, \mathsf{pk}_{I_s})$ is generated by the key generation algorithm of the signature scheme used by issuers and $(\mathsf{sk}_{I_e}, \mathsf{pk}_{I_e})$ is generated by the key generation algorithm of the encryption scheme. The key generation algorithm $\mathcal{G}_C$ generates a secret/public key

pair $(\mathsf{sk}_C, \mathsf{pk}_C)$ using the key generation algorithm of the signature scheme used by cards. ClessPay consists of the following phases:

Initialization Phase: This phase is executed by T and C. If this phase cannot be completed successfully, then T cancels the transaction.

T and C generate ephemeral secret/public key pairs for the distance bounding protocol $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$. So, C first picks the random coins r and runs the deterministic algorithm $\mathcal{K}_P(1^n; r)$ to generate $(\mathsf{sk}_P, \mathsf{pk}_P)$. Here, what C does is equivalent to running $\mathcal{K}_P(1^n)$. C needs to generate the random coins used in $\mathcal{K}_P(1^n)$ because they will be needed in the last phase as a one-time proof for having generated pk_P. Then, T runs $\mathcal{K}_V(1^n)$ to obtain $(\mathsf{sk}_V, \mathsf{pk}_V)$ used for DB. T sends τ and pk_V to C. After receiving them, C picks an identifier id and replies with id and pk_C to introduce itself.

T and C start the distance bounding protocol so that T determines the distance of C. Therefore, T runs the verifier algorithm $V(\mathsf{sk}_V, \mathsf{pk}_V)$ of DB and C runs the prover algorithm $P(\mathsf{sk}_P, \mathsf{pk}_P, \mathsf{pk}_V)$ of DB. At the end, V outputs Out_V which shows if C is close or not and private output $\mathsf{POut}_P = \mathsf{pk}_P$. If $\mathsf{Out}_V = 0$, then T cancels the transaction. Otherwise, they continue with the next phase. Remark that, T still does not know if the card whose distance is determined is an authorized card because C has not authenticated itself with its (static) public key pk_C yet.

Approval Phase: This phase aims to check with the issuer whether the card can execute the transaction. T first sends $\mathsf{pk}_C, \mathsf{pk}_P, id, \tau$ to I. I checks if the card with pk_C is in $DataB$. If it is in $DataB$, it retrieves the card information of the card (CI) and runs the algorithm $\mathsf{Policy}(\mathsf{pk}_C, CI, \tau)$ which outputs 1 if the card has the privilege to execute τ[2]. If this algorithm returns 0, the transaction is canceled. Otherwise, I approves the transaction.

If it is approved, I signs with sk_{I_s} the message $(id, \tau, \mathsf{pk}_C)$. This signature is necessary for cards to be sure that they are approved for the payment. Then, it sends this signature S_I to T and T relays it to C. C runs the verification algorithm of the signature scheme $\mathsf{Verify}_{\mathsf{pk}_{I_s}}(S_I, id, \tau, \mathsf{pk}_C)$ to be sure that C and I have the same id, τ, pk_C. If C verifies S_I, then the next phase begins. Otherwise, C cancels.

Completion Phase: In this phase, the execution of the transaction τ with id is completed by I, T and C. First, C signs the message id, τ, r with sk_C as a proof of execution of the payment. The reason of signing r is showing that C took part in the DB protocol. Then, it encrypts the signature S_C and r by using the key pk_{I_e}. The reason of the encryption is to hide r. At the end, C sends the encryption (E_C) to T. T relays it to I. At this point, the transaction is completed for C and it privately outputs (id, τ).

In order to obtain S_C and r, I first decrypts E_C with sk_{I_e}. I verifies that r generates pk_P by running $\mathcal{K}_P(1^n; r)$. If it is verified, it also verifies S_C with $\mathsf{Verify}_{\mathsf{pk}_C}(S_C, id, \tau, r)$. If the signature is valid, then it sends $\mathsf{Out}_I = 1$ to T and privately outputs $(\mathsf{pk}_C, id, \tau)$. Otherwise, I cancels the transaction.

[2] The Policy checks the execution right of a card depending on the bank policy. So, we do not discuss about how this verification happens.

Cancel the transaction: As it can be seen in the protocol, the cancellation can be done by I, T or C. In the case of timeout, parties cancel as well. When I cancels, it sets $\mathsf{Out}_I = 0$ and sends Out_I to T. Then, T cancels as well. When T cancels, it sets $\mathsf{Out}_T = 0$ and terminates. When C cancels, it sends a cancel message to T and terminates with $\mathsf{POut}_C = \perp$.

3.2 Security

Theorem 1. *Assuming that $DB = (\mathcal{K}_P, \mathcal{K}_V, V, P, B)$ is DF secure (Definition 7), DH-secure (Definition 8) and MiM-secure (Definition 6), the encryption scheme is IND-CCA secure and the signature scheme used by cards is secure against the existential forgery under no message attacks (EF-0MA), ClessPay is secure against malicious cards (Definition 3).*

Proof. We define a sequence of games Γ_i where we denote p_i as a success probability of winning Γ_i. We assume that we have honest cards $\{C_1, C_2, ..., C_k\}$ and their public keys are in a set $\{pk_{C_i}\}$.

Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 3. There is a distinguished terminal instance $\mathcal{T}$ which privately outputs $\mathsf{POut}_T = (\mathsf{pk}''_C, id_T, \tau_T)$ and in which the V protocol outputs $\mathsf{POut}_V = \mathsf{pk}'_P$, and a distinguished issuer $\mathcal{I}$ which communicates with $\mathcal{T}$ and privately outputs $\mathsf{POut}_I = (\mathsf{pk}'_C, id_I, \tau_I)$. In Γ_0, the adversary cannot win with **the first condition** in Definition 3 because I always cancels the transaction if $(\mathsf{pk}'_C, .) \notin DataB$.

Γ_1 : It is the same game as Γ_0 except that $(\mathsf{pk}'_C, id_I, \tau_I)$ is always equal $(\mathsf{pk}''_C, id_T, \tau_T)$. Because of our secure and authenticated channel assumption between T and I and because of the honesty of T, they have the same public-key, identifier and the transaction. Besides, T outputs 1, if I outputs 1. So, $p_1 = p_0$. In Γ_1, the adversary cannot win with **the fourth condition** in Definition 3.

Γ_2 : It is the same game as in Γ_1 except that instances of honest cards do not sign and they encrypt a random message. Basically, each stores the ciphertext together with the identifier, transaction and static/ephemeral public keys to a table. $\mathcal{I}$ does not decrypt such random ciphertexts and retrieves their data from the table. More specifically, we simulate them as follows:

$C(\mathsf{sk}_C, \mathsf{pk}_C, \mathsf{pk}_I)$
... (unchanged until sign)
pick R
$E_C = \mathsf{Enc}_{\mathsf{pk}_{I_e}}(R)$
store $(E_C, id, \tau, \mathsf{pk}_C, \mathsf{pk}_P)$ **in** TableE
send E_C
$\mathsf{POut}_C = (id, \tau)$

$\mathcal{I}(\mathsf{sk}_I, \mathsf{pk}_I, DataB)$
... (unchanged until the reception of E_C)
if $(E_C, id, \tau, \mathsf{pk}_C, .) \in$ TableE:
  retrieve pk **s.t.** $(E_C, id, \tau, \mathsf{pk}_C, \mathsf{pk}) \in$ TableE
  if $\mathsf{pk} \neq \mathsf{pk}_P$: cancel
  $\mathsf{Out}_I = 1, \mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)$
else: the same as after receiving E_C

We can easily show Γ_1 and Γ_2 are indistinguishable by using the IND-CCA security of the encryption scheme. So, $|p_2 - p_1|$ is negligible. Remark that the random coins of the honest cards are not used in Γ_2.

Γ_3 : It is the same game as Γ_2 except that $\mathsf{Out}_V = 0$ after the execution of $V(\mathsf{sk}_V, \mathsf{pk}_V)$ if one of the situations happens:

1. no party is close to $\mathcal{T}$,
2. pk'_P is generated by no honest card and there is no adversary close to $\mathcal{T}$,
3. pk'_P is generated by an honest card but it has no instance close to $\mathcal{T}$.

Γ_3 and Γ_2 are indistinguishable because the probability that $\mathsf{Out}_V = 1$ if one of the situations above happens is negligible. $\mathsf{Out}_V = 1$ when the 1^{st} situation happens with negligible probability due to the the DF-security of DB. $\mathsf{Out}_V = 1$ when the 2^{nd} situation happens with negligible probability due to the DH-security of DB. $\mathsf{Out}_V = 1$ when the 3^{rd} situation happens with negligible probability due to the MiM-security of DB. Note that we can simulate an honest card instance in Γ_3 by using a prover instance in the MiM-game because r is not used by honest card instances. Therefore, $|p_3 - p_2|$ is negligible.

Γ_4 : It is the same game as in Γ_3 except that $\mathcal{I}$ cancels after decrypting and obtaining the random coins r where $\mathcal{K}_P(1^n; r) \rightarrow (\mathsf{sk}_P, \mathsf{pk}_P)$ and $(\mathsf{sk}_P, \mathsf{pk}_P)$ is generated by an honest card instance.

$\boldsymbol{\mathcal{I}}(\mathsf{sk}_I, \mathsf{pk}_I, DataB)$
... (unchanged until the reception of E_C)
if $(E_C, id, \tau, \mathsf{pk}_C, .) \in$ TableE: **retrieve** pk **s.t.** $(E_C, id, \tau, \mathsf{pk}_C, \mathsf{pk}) \in$ TableE
  if pk $\neq \mathsf{pk}_P$: cancel
  $\mathsf{Out}_I = 1, \mathsf{POut}_I = (\mathsf{pk}_C, id_T, \tau_T)$
else: $S_C, r = \mathsf{Dec}_{\mathsf{sk}_I}(E_C)$, $\mathcal{K}_P(1^n; r) \rightarrow (\mathsf{sk}, \mathsf{pk})$
  if $(\mathsf{sk}, \mathsf{pk})$ is generated by an honest instance: cancel
  else if $\neg\mathsf{Verify}(S_C, id, \tau, r) \vee \mathsf{pk}_P \neq \mathsf{pk}$: cancel
  $\mathsf{Out}_I = 1, \mathsf{POut}_I = (\mathsf{pk}_C, id, \tau)$

We can easily prove that if there exists an adversary with pk_C in Γ_3 which obtains a randomness r generating the secret/public key pair used by an honest instance, then we can construct another adversary which breaks the MiM-security of DB. Clearly, during the simulation of Γ_3, if I gets r, then it generates the corresponding secret key of the prover in MiM-game and breaks the MiM-security. Since receiving such r in Γ_4 is negligible, $|p_4 - p_3|$ is negligible.

Now, we show that the adversary cannot win with **the third condition** in Γ_4. If the adversary wins with this in Γ_4, it means that $\mathsf{pk}'_C \notin \{\mathsf{pk}_{C_i}\}$ and no instance of the adversary is close to $\mathcal{T}$ during the execution of the CP protocol with $\mathcal{T}$ and $\mathcal{I}$. Due to the condition 2 in the reduction of Γ_3, pk_P must be generated by an honest card (otherwise, $\mathcal{T}$ cancels) . However, in Γ_4, it is not possible to have $\mathsf{Out}_I = 1$ while $\mathsf{pk}_C \notin \{\mathsf{pk}_{C_i}\}$ and pk_P is generated by an honest card instance (check the dashed underlined parts in the simulation of $\mathcal{I}$). So, it is not possible that $\mathsf{Out}_I = 1$, if the game is in the third condition.

Since only condition 2 and 5 of Definition 3 remain to win in Γ_3, we can assume that $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$.

Γ_5 : It is the same game as Γ_4 except we simulate Verify algorithm with Verify′ such that it only accepts the signature of malicious cards. It does not accept the signatures of honest cards' instances.

The only difference in Verify and Verify′ is in the case of $\mathsf{pk}_C \in \{\mathsf{pk}_{C_i}\}$. In this case, while Verify returns the output of the verification of the signature, Verify′ returns 0. In Γ_5 and Γ_4, no honest cards' instances generate a signature. So, the only difference between Γ_4 and Γ_5 happens when $\mathcal{I}$ obtains a forged signature of an honest card instance.

Thanks to EF-0MA security of the signature, we can easily show that forging a signature of any honest cards happens with a negligible probability to prove that Γ_5 and Γ_4 are indistinguishable.

Remark that in Γ_5, $\mathcal{I}$ have $\mathsf{Out}_I = 1$, if and only if $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P)$ is in TableE. So, we can assume that $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P) \in \mathsf{TableE}$.

If the adversary wins with the condition 2 in Γ_5, then $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and the distance between any C holding pk'_C and $\mathcal{T}$ is more than B during the execution of the protocol with id_T. Due to condition 3 in Γ_3, pk'_P must not been generated by C. So, $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P, .)$ cannot be in TableE which contradicts with our assumption. Hence, the adversary cannot win with **the second** condition.

If the adversary wins with the fifth condition, then it means that $\mathsf{pk}'_C \in \{\mathsf{pk}_{C_i}\}$ and there exists no card with pk'_C which privately outputs id_I, τ_I. Then, it means that $(E_C, id_T, \tau_T, \mathsf{pk}'_C, \mathsf{pk}'_P, .) \notin \mathsf{TableE}$ since no honest card instance has (id_T, τ_T). This contradicts with our assumption. Therefore, the adversary cannot win with **the fifth condition**. Remark that in Γ_5, the adversary cannot win the game So, p_5 is negligible meaning that p_0 is negligible. □

Theorem 2. *Assuming that the signature schemes used are existential forgery chosen message attack (EF-CMA) secure then ClessPay is* ***almost-secure*** *against malicious terminal (Definition 4).*

Proof. We recall that in almost-security, we do not need to consider condition 1*c* of Definition 4 .

Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 4. We have a distinguished issuer instance $\mathcal{I}$ which outputs $(\mathsf{pk}'_C, id_I, \tau_I)$. Remark that in Γ_0, the adversary cannot win **with condition 1*a*** $((\mathsf{pk}'_C, .) \notin DataB)$ because $\mathcal{I}$ rejects the cards which are not in $DataB$.

Γ_1 : It is the same game as Γ_2 except that no id selected by an honest card instance repeats. Clearly, $|p_1 - p_0|$ is negligible.

Γ_2 : It is the same game as Γ_1 except that we simulate $\mathcal{I}$ and its instances while generating the signature and honest cards' instances in the verification of this signature as follows:

$\boldsymbol{I(\mathsf{sk}_I, \mathsf{pk}_I, DataB)}$
$S_I = \mathsf{sign}_{\mathsf{sk}_{I_s}}(id, \tau, \mathsf{pk}_C)$
store $(S_I, id, \tau, \mathsf{pk}_C)$ **in** Table1
send S_I

$\mathsf{Verify}'_{\mathsf{pk}_{I_s}}(S, id, \tau, \mathsf{pk}_C)$
if $(S, id, \tau, \mathsf{pk}_C)$ **in** Table1
  return 1
else: return 0

$|p_2 - p_1|$ is negligible.

The output of issuer instance is the same as issuer instances in Γ_1. Therefore, we have a perfect simulation for it. The only difference happens when honest cards' instances in Γ_1 receive a valid signature verified by pk_{I_s} and not in $\mathsf{Table1}$. In this case, honest cards in Γ_1 verify the signature but they do not in Γ_2. Otherwise, the simulations of them are perfect. We can easily show that the probability of generating a valid signature which is not in the $\mathsf{Table1}$ is negligible in Γ_2 thanks to EF-CMA security of the signature scheme. We can use the public key received from the signing game as a public key of the issuer and simulate signatures of issuer instances by using the signing game. Note that sk_{I_s} is not used in the simulation but the signature generation, so we can simulate the rest of the protocol perfectly. Therefore, $|p_2 - p_1|$ is negligible.

The adversary cannot win the game **with condition 2** in Definition 4. Assume that the adversary wins with this. It implies that $(., id_C, \tau_C, \mathsf{pk}_C) \notin \mathsf{Table1}$ since id_C is unique. So, no honest card instance outputs (id_C, τ_C) in this case.

Γ_3 : It is the same game as Γ_2 except we simulate honest cards' instances while generating the signature and $\mathcal{I}$ in the verification of it as follows:

$\underline{\boldsymbol{C}(\mathsf{sk}_C, \mathsf{pk}_C, DataB)}$
$S_C = \mathsf{sign}_{\mathsf{sk}_C}(id, \tau, r)$
store $(S_C, \mathsf{pk}_C, id, \tau, r)$ **in** Table2
$E_C = \mathsf{Enc}_{\mathsf{pk}_{I_e}}(S_C, r)$
send E_C

$\underline{\mathsf{Verify}''_{\mathsf{pk}_C}(S, id_I, \tau_I, \mathsf{pk}_C, r)}$
if $(S, \mathsf{pk}_C, id, \tau, r)$ **in** Table2
return 1
else: return 0

The only difference is the output of Verify'' and Verify when a forged signature received. To show the indistinguishability of Γ_2 and Γ_3, we can EF-CMA security of the signature scheme. So, $|p_3 - p_2|$ is negligible.

Remark that in this game, the adversary cannot win with **the condition 1*b***. If $\mathcal{I}$ outputs $(\mathsf{pk}'_C, id_I, \tau_I)$, it means that an honest card instance with pk'_C added $(S, \mathsf{pk}'_C, id_I, \tau_I, .)$ in Table2 and outputted (id_I, τ_I). Hence, in Γ_3, the adversary cannot win. So, p_0 is negligible. □

We recommend using Eff-pkDB [20] as a public-key DB in ClessPay since it is shown that it is the most efficient public-key DB protocol having the necessary security requirements for ClessPay. It requires one exponentiation and hashing.

The assumption on the signature scheme used by cards differ in Theorem 1 (EF-0MA) and Theorem 2 (EF-CMA). Hence, it looks like to have security against both terminals and cards we need DF, DH, MiM-secure DB protocol, IND-CCA secure encryption scheme, and EF-CMA secure signature schemes. However, we could have the almost security against malicious terminal if we have the following assumptions in Theorem 2: the encryption scheme is IND-CCA secure and the signature scheme used by cards is EF-0MA secure. In this case, the proof of Theorem 2 would need the same games Γ_2 and Γ_5 in the proof of Theorem 1 instead of Γ_3 in the proof of Theorem 2. So, actually, to have full security in ClessPay, we need DF, DH, MiM-secure DB protocol, IND-CCA secure encryption scheme, EF-CMA secure signature for issuers, and EF-0MA secure signature for cards.

4 EMV Analysis

EMV key setting is different than our contactless payment key setting because it has a symmetric key shared between the card and its issuer as well as asymmetric keys. An issuer I has secret/public key pair S_I/P_I. It also has a master symmetric key MK_{AC}. A card C shares MK_{AC} with its issuer I. It has public/secret key pair P_{IC} and S_{IC}. P_{IC} is signed by I's private key S_I. C stores certified P_I. We assume that the terminal T knows the public key of the certificate authority (CA) to verify P_I and so P_{IC}. We also assume that the channel between I and T is authenticated.

For the sake of simplicity, in our description, we assume that C knows all terminal related information and the authentication method. T also knows the card related information.

EMV contactless session consists of four phases without card holder (user) verification method:
Contact Establishment with NFC card: T detects C.

Transaction Initialization: T sends the transaction τ to C. Then, C responds with its public key P_{IC} and card information such as PAN and expiration date (ED). If T verifies P_{IC}, it continues.
Relay Resistance Protocol [3]: This protocol is executed if C and T support it. Here, we assume that they support this feature. T picks a random number R_1 and sends this to C. C responses with another random number R_2. It also sends timing estimates (*timings*): Min and Max Time For Processing, Device Estimated Transmission Time. Then, T checks if the total time passed after sending R_1 exceeds the limit (let's call it B). If the total time does not exceed B, then the next phase begins. Otherwise, the transaction is canceled.
Data Authentication: There are three type of authentication methods in EMV: Static Data Authentication (SDA), Dynamic Data Authentication (DDA) and Combined Data Authentication (CDA). Because of some weaknesses in SDA and DDA (replay attacks and wedge attacks), we consider CDA which is combined with the next phase.
Transaction: T sends a random number UN_T to request a cryptogram generation from C. In EMV, three type of cryptograms exist: Transaction Certificate (TC), Authorization Request Cryptogram (ARQC), Application Authentication Cryptogram (AAC). Here, we consider the online verification where T requests ARQC. TC is used for the offline verification by the issuer and AAC is used to cancel the transaction.

Online Verification: C increases its counter ATC and generates a secret key SK_{AC} by using ATC and MK_{AC}. Then, it generates the cryptogram $ARQC$: a MAC of UN_T, ATC, τ with using SK_{AC}. C sends the cryptogram AC to T and T relays it to I with the card information. I verifies $ARQC$ and possibly validate the information of C. If $ARQC$ passes verification and card is validated for the transaction, then $ARC = 1$ and I generates a MAC of $ARQC$ and ARC with the secret key SK_{AC}. This MAC is called as $ARPC$. I sends $ARPC$ with the message to T and T relays it to C if $ARC = 1$. Otherwise, it cancels. C verifies $ARPC$. If the verification and ARC is 1 then C generates the second cryptogram TC. TC is a MAC of CDOL2's objects with SK_{AC} (See [4], Table 26) to show transaction is complete. Also, it picks a random number UN_C and signs UN_C, UN_T, ATC, TC, $timings, R_1, R_2$ with S_{IC}. C At the end, C sends the signature and TC to T.

Terminal checks if the signature and the data signed are valid. Later, the terminal contacts with the issuer to receive the reimbursement and gives TC as a proof of transaction completion by the card. In this case, the issuer verifies TC to execute the reimbursement.

EMV in Our Model: The EMV protocol can have the following maps to have the same structure as in Definition 1: $(\mathsf{sk}_C, \mathsf{pk}_C) = ((MK_{AC}, S_{IC}), P_{IC})$, $(\mathsf{sk}_I, \mathsf{pk}_I) = ((MK_{AC}, S_I), P_I)$, $id = ATC$, $\mathsf{Policy}(\mathsf{pk}_C, CI, id, \tau) = ARC$, Out_T = approval/ decline, $\mathsf{Out}_I = \mathsf{Verify}(TC, UN_T, ATC, \tau)$, $\mathsf{POut}_I = (P_{IC}, ATC, \tau)$, $\mathsf{POut}_T = (P_{IC}, ATC, \tau)$ and $\mathsf{POut}_C = (ATC, \tau)$.

Security Against Malicious Terminal in EMV: Clearly, the EMV protocol is not secure according to Definition 4 since the malicious terminal can approve

relay resistance protocol without considering the distance of C. However, it is almost-secure against malicious terminals. We prove this as follows:

Theorem 3. *Assuming that* MAC *is EF-CMA secure and* Gen *is a pseudo-random permutation, then EMV protocol is almost-secure against malicious terminals (Definition 4).*

Proof. Γ_0 : The instances of the issuer, terminals and cards play the game in Definition 4. We have a distinguished issuer instance $\mathcal{I}$ which outputs (P_{IC}, ATC, τ_I). In Γ_0, there exists at most one card instance with P_{IC} seeing ATC as ATC is a counter and incremented by each new instance. Let's call this instance as $\mathcal{C}$.

Γ_1 : It is the same game as Γ_0 except that the honest card instances picks a random SK'_{AC} instead of generating it with $\mathsf{Gen}(MK', ATC)$ and stores the random SK'_{AC} in Table1 as (MK', ATC', SK'_{AC}). If an issuer instance receives card information belongs to an honest card then it retrieves SK'_{AC} from Table1. Since Gen is pseudo-random permutation, $|p_1 - p_0|$ is negligible.

Γ_2 : It is the same game as Γ_1 except that we simulate MAC generation of honest cards and verification of MACs of honest cards' instances by the issuer as follows:

$I(P'_{IC}, S'_{IC}, P'_I, MK'_{AC})$

$ATC' = ATC' + 1$
pick SK'_{AC}
store $(MK'_{AC}, ATC', SK'_{AC})$
$ARQC = \mathsf{MAC}_{SK'_{AC}}(UN_T, ATC', \tau)$
store $(SK'_{AC}, UN'_T, ATC', \tau', ARQC)$ in Table_{ARQC}
rest is the same until TC/AAC generation
if ARC $= 1$ **and** $\mathsf{Verify}(ARPC', SK'_{AC})$:
  $TC = \mathsf{MAC}_{SK'_{AC}}(UN_T, ATC', \tau)$
  store $(SK'_{AC}, UN'_T, ATC', \tau', TC)$ in Table_{TC}
else:
  $AAC = \mathsf{MAC}_{SK'_{AC}}(UN_T, ATC', \tau)$
  store $(SK'_{AC}, UN'_T, ATC', \tau', AAC)$ in Table_{AAC}

$\mathsf{Verify}'(AC, SK_{AC})$

if $(SK_{AC}, UN_T, ATC, \tau, AC) \in \mathsf{Table}_{AC}$
  return 1
else: return 0

Γ_2 is indistinguishable from Γ_1 because of the security of MAC. The similar reduction in the proof of Theorem 1 from Γ_4 to Γ_5 can be used to prove the indistinguishably. So, $|p_2 - p_1|$ is negligible.

Γ_3 : It is the same game with Γ_2 except that $\mathcal{I}$ generates $ARPC$ and then stores it to Table_{ARPC} (similar storing as in Γ_2). Then, the honest cards verify $ARPC$ by checking if it is in the Table_{ARPC}. Γ_3 is indistinguishable from Γ_2 because of the security of MAC. So, $|p_3 - p_2|$ is negligible.

Clearly, in Γ_3, the adversary cannot win with the condition $1b$ because $\mathcal{I}$ privately outputs (P_{IC}, ATC, τ) if and only if the card with P_{IC} outputs ATC, τ.

In addition, it cannot win with the condition 2 because if $ARC \neq 1$, then no honest card outputs ATC, τ and if an honest card receives a valid $ARPC$ having $ARC = 1$, then it means that $ARPC$ is in Table_{ARPC}. So, $\mathcal{I}$ has (P_{IC}, ATC, τ). Since the adversary cannot win in Γ_3, p_0 is negligible. □

However, there exists another problem in EMV related to ATC which is not considered in our model. It can be explained as follows: ATC is 16-bit number and incremented at the beginning of each session. If ATC reaches the limit which

65535, then the card is not valid anymore because EMV specification does not let rotating the counter due to the security reasons. According to EMV specification [4] if cards are used normally, it will approach the limit (65,535) transaction limit not so fast (60 per day every day for a 3-year card). However, an attacker who does not aim to make a payment but aims to invalidate the card can trigger the card at most 65,535 times. Then, the card cannot be used anymore.

Security Against Malicious Card in EMV: Unfortunately, EMV is not secure against malicious cards. In the followings, we show that an adversary can win with the second, third and fourth condition in Definition 3.
Fake Transaction Attack: This attack comes from the fact that T cannot validate TC in the signature $SDAD$ because it does not have SK_{AC}. Therefore, a malicious card can generate an invalid TC' in the last cryptogram generation process and use this cryptogram while generating this signature. Then, the terminal will approve the payment because the signature is correct. However, TC' is not valid. So, when T contacts with I, I cannot validate TC'. In this case, the malicious card succeeds to break the security of EMV with breaking **the fourth condition** of Definition 3 because I cancels while T does not.
Distance Fraud Attack: A malicious card can initiate a payment process with T, while it is not close T. In this case, it can send R_2 before seeing R_1 in order to reply early enough. In this case, T thinks that the card is close. Here, the malicious card succeeds to break the security of EMV with breaking **the third condition** of Definition 3. This type of attack is dangerous for an EMV payment because the malicious card can claim later that it does not do the payment by showing that it was in somewhere else.
MiM Attack: The relay resistance protocol in EMV constructed to prevent relay attacks by a MiM-adversary. In this attack scenario, a MiM-adversary relays the messages between the card and the terminal to do the payment without the card's consent. The relay resistance protocol aims to prevent it by checking the distance of the card. The assumption on its security based on the fact that the adversary cannot relay the messages faster than the speed of light. Therefore, the adversary cannot succeed to pass the relay resistance protocol because it cannot guess R_2 before R_2 is picked by the card. However, it has been shown that with guessing attacks [15] the security against relay attacks is breakable for the protocols with single challenge/response bit strings exchanges. In addition, Chothia et al. [14] have already explained this vulnerability.

5 Conclusion

In this paper, we concentrated on formalism of CP system. In this direction, we formally define contactless payment by defining the inputs and outputs of the algorithms of issuers, terminals and cards. Based on this definition, we gave two security definitions against malicious cards and malicious terminals. We also considered relay attacks which are very common attacks in CP.

We also designed a contactless-payment protocol ClessPay in our model. In this protocol, the terminal determines the distance of the card by using a

secure public-key distance bounding protocol to prevent the relay attack and then the rest of the protocol continues with the authentication of the card and the issuer. We proved the security of ClessPay against malicious cards and malicious terminals formally.

Finally, we analyzed current EMV-contactless protocol [5] in our model. We realized that it is not secure against malicious cards because MiM-attack and DF-attack which are based on relay attacks. In addition to this, we formally proved that EMV-contactless protocol is secure against malicious terminals. Our analysis is the first formal cryptographic analysis of EMV-contactless protocol.

If we compare ClessPay and EMVCo in regard to cryptographic computations executed by the cards, we see that EMVCo is slightly more efficient since public-key operations are less in EMVCo. A card in EMVCo has to compute two MAC, verify one MAC and generate one signature. While a card in ClessPay has to compute one public-key encryption, generate one signature and verify one signature. However, to have the highest level of the security, it is the price to pay and with a dedicated hardware on smart cards, this price is not so high. As a future work, assuming that changing completely EMV specification is very hard, we can recommend some adaptations on EMVCo to have full security without not so much change in the basic structure of the protocol.

References

1. Contactless payment market by solution (payment terminal, mobile payment, transaction and data management, security and fraud management), service (professional, managed), payment mode (mobile handsets, smart cards), vertical - global forecast to 2021. Available at https://www.marketsandmarkets.com/Market-Reports/contactless-payments-market-1313.html.
2. EMV acquirer and terminal security guidelines.
3. EMV contactless specifications for payment systems, Book C-2: Kernel 2 specification.
4. EMV integrated circuit card specifications for payment systems, Book 2: Security and key management.
5. EMVCo: EMV contactless specifications for payment systems, version 2.4 (2014).
6. G. Avoine, X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A terrorist-fraud resistant and extractor-free anonymous distance-bounding protocol. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 800–814. ACM, 2017.
7. M. Bond, M. O. Choudary, S. J. Murdoch, S. Skorobogatov, and R. Anderson. Be prepared: The emv preplay attack. *IEEE Security & Privacy*, 13(2):56–64, 2015.
8. M. Bond, O. Choudary, S. J. Murdoch, S. Skorobogatov, and R. Anderson. Chip and skim: cloning emv cards with the pre-play attack. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 49–64. IEEE, 2014.
9. I. Boureanu, A. Mitrokotsa, and S. Vaudenay. Secure and lightweight distance-bounding. In *Lightweight Cryptography for Security and Privacy*, LNCS 8162, pages 97–113. Springer, 2013.
10. I. Boureanu and S. Vaudenay. Optimal proximity proofs. In *Inscrypt*, LNCS 8957, pages 170–190. Springer, 2014.

11. S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT*, LNCS 765, pages 344–359. Springer-Verlag, 1993.
12. X. Bultel, S. Gambs, D. Gérault, P. Lafourcade, C. Onete, and J.-M. Robert. A prover-anonymous and terrorist-fraud resistant distance-bounding protocol. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 121–133. ACM, 2016.
13. N. Chandran, V. Goyal, R. Moriarty, and R. Ostrovsky. Position based cryptography. In *CRYPTO*, volume 9, pages 391–407. Springer, 2009.
14. T. Chothia, F. D. Garcia, J. De Ruiter, J. Van Den Breekel, and M. Thompson. Relay cost bounding for contactless EMV payments. In *International Conference on Financial Cryptography and Data Security*, pages 189–206. Springer, 2015.
15. J. Clulow, G. Hancke, M. Kuhn, and T. Moore. So near and yet so far: Distance-bounding attacks in wireless networks. *Security and Privacy in Ad-hoc and Sensor Networks*, pages 83–97, 2006.
16. C. Cremers, K. B. Rasmussen, B. Schmidt, and S. Capkun. Distance hijacking attacks on distance bounding protocols. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 113–127. IEEE, 2012.
17. S. Drimer, S. J. Murdoch, et al. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX security symposium*, volume 312, 2007.
18. A. Francillon, B. Danev, and S. Capkun. Relay attacks on passive keyless entry and start systems in modern cars. In *NDSS*, 2011.
19. L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *International Workshop on Radio Frequency Identification: Security and Privacy Issues*, LNCS 6370, pages 35–49. Springer, 2010.
20. H. Kılınç and S. Vaudenay. Efficient public-key distance bounding protocol. In *ASIACRYPT*, 2016.
21. H. Kılınç and S. Vaudenay. Contactless access control based on distance bounding. 10599:195–213, 2017.
22. K. Markantonakis, L. Francis, G. Hancke, and K. Mayes. Practical relay attack on contactless transactions by using nfc mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012.
23. M. Roland and J. Langer. Cloning credit cards: A combined pre-play and downgrade attack on emv contactless. In *WOOT*, 2013.
24. S. Vaudenay. On modeling terrorist frauds. In *Provable Security*, LNCS 8209, pages 1–20. Springer, 2013.
25. S. Vaudenay. On privacy for RFID. In *Provable Security*, pages 3–20. Springer, 2015.
26. S. Vaudenay. Private and secure public-key distance bounding application to NFC payment. In *Financial Cryptography*, LNCS 8975, pages 207–216, 2015.
27. S. Vaudenay. Sound proof of proximity of knowledge. In *Provable Security*, LNCS 9451, pages 105–126. Springer, 2015.
28. M. Weiß. Performing relay attacks on ISO 14443 contactless smart cards using nfc mobile equipment. *Master's Thesis in Computer Science, University of Munich*, 2010.