A Cryptographic Study of Tokenization Systems

Sandra Díaz-Santiago*, Lil Maria Rodriguez-Henriquez and Debrup Chakraborty
Department of Computer Science, CINVESTAV-IPN,
Av. IPN 2508, San Pedro Zacatenco, Mexico City 07360, Mexico

Keywords: Payment Card Industry Standard, Tokenization, Symmetric Encryption, Format Preserving Encryption, Provable Security.

Abstract: Payments through cards have become very popular in today's world. All businesses now have options to receive payments through this instrument, moreover most organizations store card information of its customers in some way to enable easy payments in future. Credit card data is a very sensitive information and its theft is a serious threat to any company. Any organization that stores such data needs to achieve payment card industry (PCI) compliance, which is an intricate process. Recently a new paradigm called "tokenization" has been proposed to solve the problem of storage of payment card information. In this paradigm instead of the real credit card data a token is stored. To our knowledge, a formal cryptographic study of this new paradigm has not yet been done. In this paper we formally define the syntax of a tokenization system, and several notions of security for such systems. Finally, we provide some constructions of tokenizers and analyze their security in the light of our definitions.

1 INTRODUCTION

In our digital age, credit cards have become a usual payment instrument. With increasing popularity of business through internet, every business requires to maintain credit card information of its clients in some form. Credit card data theft causes a serious financial loss and a critical damage to the "brand image" of the company in question.

The Payment Card Industry Security Standard Council (PCI SSC) is an organization responsible for the development and deployment of various best practices in ensuring security of credit card data. In particular PCI SSC has developed a standard called the PCI Data Security Standard (PCI DSS) (PCI Security Standards Council, 2008). This standard dictates that organizations, which process card payments, must protect cardholder data when they store, transmit and process them. Actually it proposes to use "strong cryptography" as a possible solution.

Traditionally credit card numbers have been used as a primary identifier in many business processes and are scattered across the environment of merchant sites. But, in most systems where credit card numbers are stored, the data itself is not required, it can be replaced by some other information which would "look like" credit card numbers. This observation has lead to a paradigm called "tokenization", where credit card numbers are replaced by *tokens*. Tokens are numbers which represent credit cards but are unrelated to the real credit card numbers.

There have been numerous industry white papers and similar documents which discuss about the possible solutions to the tokenization problem (Securosis White Paper, 2011; Voltage Security White paper, 2012; RSA White paper, 2012). PCI SSC has also formulated its guidelines regarding *tokenization* (PCI Security Standards Council, 2011). However to our knowledge a formal cryptographic analysis of the problem has not been done.

SMALL DOMAIN ENCRYPTION. One obvious solution for securing credit card numbers in a merchant site is to encrypt them. But a strict requirement for applying encryption is that the cipher should look like a credit card number. This necessity opened up an interesting problem. A typical credit card number consists of sixteen (or less) decimal digits, if this is treated as a binary string, is about 53 bits long. Thus, direct application of a block cipher (say AES) to encrypt would result in a considerable length expansion, and it would not be possible to encode the cipher into sixteen decimal digits.

*Sandra Díaz-Santiago is on academic leave from Escuela Superior de Cómputo (ESCOM-IPN), Av. Juan de Dios Bátiz, Col. Lindavista, México D.F. 07738, México

Díaz-Santiago S., Maria Rodriguez-Henriquez L. and Chakraborty D..
A Cryptographic Study of Tokenization Systems.
DOI: 10.5220/0005062803930398
In *Proceedings of the 11th International Conference on Security and Cryptography* (SECRYPT-2014), pages 393-398
ISBN: 978-989-758-045-1

The general problem was named by Voltage Security as *format preserving encryption* (FPE), which refers to an encryption algorithm which preserves the format of the message. There have been some interesting solutions to this problem, but none of them can be considered to be efficient (Bellare et al., 2009; Brier et al., 2010; Hoang et al., 2012; Morris et al., 2009; Stefanov and Shi, 2012). A credit card number encrypted by an FPE scheme can act as a token. Such a solution is also provided by Voltage Security (Voltage Security White paper, 2012). But to the best of our knowledge, this is the only solution to the tokenization problem with known cryptographic guarantees.

OUR CONTRIBUTION. We study the paradigm of tokenization from a cryptographic viewpoint. We point out the basic needs for a tokenization system, and define a syntax for the problem. Further, we develop the corresponding security model, in line with concrete provable security. We propose three different security notions IND-TKR, IND-TKR-CV, and IND-TKR-KEY, which depend on three different threat models. Finally, we propose three generic constructions of tokenization systems, namely TKR1, TKR2 and TKR2a. We also prove security of our constructions in the proposed security models. For lack of space, in this version we provide only the basic ideas. A more comprehensive version of this work can be found in the IACR eprint archive.

2 TOKENIZATION SYSTEMS: REQUIREMENTS AND PCI DDS GUIDELINES

The basic architecture of a tokenization system is described in Fig.1. In the diagram we show three separate environments: the merchant site, the tokenization system and the card issuer. The basic data objects of interest are the primary account number (PAN), which is basically the credit card number and the token which represents the PAN. A customer communicates with the merchant environment through the "point of sale", where the customer provides its PAN. The merchant sends the PAN to the tokenizer and gets back the corresponding token. The merchant may store the token in its environment. At the request of the merchant the tokenizer can *detokenize* a token and send the corresponding PAN to the card issuer for payments.

We show the tokenization system to be separated from the merchant environment, but it is also possible that the merchant itself implements its tokenization

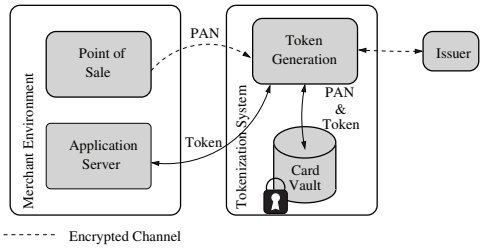

Figure 1: Architecture of the tokenization system.

solution, and in that case, the tokenization system is a part of the merchant environment.

As described in (PCI Security Standards Council, 2011), a *tokenization system* has the following components:

1. **A Method for Token Generation:** A process to create a token corresponding to a *primary account number* (PAN).
2. **A Token Mapping Procedure:** It refers to the method used to associate a token with a PAN. Such a method would allow the system to recover a PAN, given a token.
3. **Card Vault:** It is a repository which usually will store pairs of PANs and tokens and maybe some other information required for the token mapping. Since it may contain PANs, it must be specially protected according the PCI DSS requirements.
4. **Cryptographic Key Management:** This module is a set of mechanisms to create, use, manage, store and protect keys used for the protection of PAN data and also the information involved in token generation.

Here we state two basic requirements for tokens and tokenization systems. We assume that tokenization is provided as a service, thus multiple merchants utilize the same system for their tokenization needs.

1. **Format Preserving:** The token should have the same format as that of the PANs, so that the stored PANs can be easily replaced by the tokens in the merchant environment.
2. **Uniqueness:** The token generation method should be deterministic. Thus the tokens for a specific PAN should be unique, i.e., if the same PAN is tokenized twice by the same merchant then the same token should be obtained. Moreover, in a specific merchant environment two different PANs should be represented by different tokens.

In what follows we will adopt the following notations. For a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ will denote x to be an element chosen uniformly at random from $\mathcal{S}$. We consider an adversary as a probabilistic algorithm that outputs a

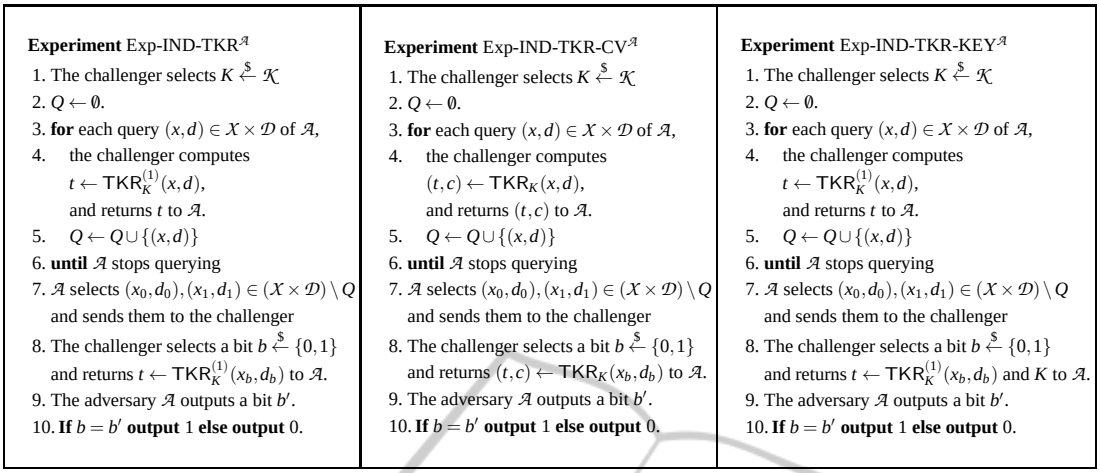

Figure 2: Experiments used in the security definitions: IND-TKR, IND-TKR-CV and IND-TKR-KEY.

bit b. $\mathcal{A}^{\mathcal{O}} \Rightarrow b$, will denote the fact that an adversary $\mathcal{A}$ has access to an oracle $\mathcal{O}$ and outputs b.

If $\mathbf{E} : \mathcal{K} \times \mathcal{D} \times \mathcal{X} \to \mathcal{X}$ be a tweakable permutation (Halevi and Rogaway, 2004) with message/cipher space $\mathcal{X}$ and tweak space $\mathcal{D}$, we define the $\widetilde{\text{prp}}$ advantage of an adversary $\mathcal{A}$ as

$$\mathbf{Adv}_{\mathbf{E}}^{\widetilde{\text{prp}}}(\mathcal{A}) = \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathbf{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}^{\mathcal{D}}(\mathcal{X}) : \mathcal{A}^{\pi(\cdot,\cdot)} \Rightarrow 1]\right|,$$

where $\mathsf{Perm}^{\mathcal{D}}(\mathcal{X})$, is the set of all tweak indexed length preserving permutations on $\mathcal{X}$.

Let $\mathbf{E} : \mathcal{K} \times \mathcal{D} \times \mathcal{X} \to \mathbb{C}$ be a deterministic encryption scheme with key space $\mathcal{K}$, tweak space $\mathcal{D}$, message space $\mathcal{X}$ and cipher space $\mathbb{C}$. For all $K \in \mathcal{K}$, $\mathbf{E}_K(\cdot,\cdot)$ must be an injection from $\mathcal{D} \times \mathcal{X} \to \mathbb{C}$. We define the det-cpa advantage of any adversary $\mathcal{A}$, which does not repeat any query as

$$\mathbf{Adv}_{\mathbf{E}}^{\text{det-cpa}}(\mathcal{A}) = \left|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathbf{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1]\right|,$$

where $\$(.,.)$ is an oracle, which on input $(d,x) \in \mathcal{D} \times \mathcal{X}$ returns a random string of the size of the cipher-text of x.

3 A GENERIC SYNTAX

A tokenization system has the following components:

1. $\mathcal{X}$, a finite set of *primary account numbers* or PAN's. $\mathcal{X}$ contains strings from a suitable alphabet with a specific format.
2. $\mathcal{T}$, a finite set of tokens. $\mathcal{T}$ also contains strings from a suitable alphabet with a specific format. It may be the case that $\mathcal{T} = \mathcal{X}$.
3. $\mathcal{D}$, a finite set of associated data. The associated data can be any data related to the business process.
4. CV, the card vault. The card vault is a repository where PAN's and tokens are stored. In our syntax we shall use the CV to represent a state of the tokenization system. Whenever a new PAN is tokenized, possibly both the PAN and the generated token are stored in the CV, along with some additional data. Disregarding the structure of the CV, we consider that "basic" elements of CV comes from a set $\mathbb{C}$.
5. $\mathcal{K}$, a key generation algorithm. A tokenization system may require multiple keys, all these keys are generated through the key generation algorithm.
6. TKR, the tokenizer. It generates tokens from the PANs. It receives as input: the CV as a state, a key K generated by $\mathcal{K}$, some associated data d which comes from a set $\mathcal{D}$, and a PAN $x \in \mathcal{X}$. An invocation of TKR outputs a token and also produces an element from $\mathbb{C}$ used to update the CV. We use the square brackets to denote this interaction. We formally see TKR as a function $\mathsf{TKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{X} \times \mathcal{D} \to \mathcal{T} \times \mathbb{C}$. For convenience, we shall implicitly assume the interaction of TKR with CV, and we will use $\mathsf{TKR}_K^{(1)}(x,d)$ and $\mathsf{TKR}_K^{(2)}(x,d)$ to denote the two outputs (in $\mathcal{T}$ and $\mathbb{C}$, respectively) of TKR.
7. DTKR, the detokenizer which inverts a token to a PAN. We denote a detokenizer as a function $\mathsf{DTKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{T} \times \mathcal{D} \to \mathcal{X} \cup \{\perp\}$. For detokenization also, we shall implicitly assume its interaction with CV and for $K \in \mathcal{K}$, $d \in \mathcal{D}$ and $t \in \mathcal{T}$, we shall write $\mathsf{DTKR}_K(t,d)$ instead of

$\mathsf{DTKR}[\mathsf{CV}](K,t,d)$.

A tokenization procedure TKR_K should satisfy the following:

- For every $x \in \mathcal{X}$, $d \in \mathcal{D}$ and $K \in \mathcal{K}$, $\mathsf{DTKR}_K(\mathsf{TKR}_K^{(1)}(x,d),d) = x$.
- For every $d \in \mathcal{D}$, and $x,x' \in \mathcal{X}$, such that $x \neq x'$, $\mathsf{TKR}_K^{(1)}(x,d) \neq \mathsf{TKR}_K^{(1)}(x',d)$.

The second criteria focuses on a weak form of uniqueness. We want that two different PANs with the same associated data should produce different tokens. This makes sense if we consider the associated data to be a merchant identifier. We do not want that a single merchant obtains the same token for two different PANs, but we do not care if two different merchants obtain the same token for two different PANs.

4 SECURITY NOTIONS

We define three different security notions, which consider three different attack scenarios:

1. IND-TKR: Tokens are only public. This represents the most realistic scenario where an adversary has access to the tokens only, and the card vault data remains in-accessible.
2. IND-TKR-CV : The tokens and the contents of the card vault are public. This represents an extreme scenario where the adversary gets access to the card vault data also.
3. IND-TKR-KEY : This represents another extreme scenario where the tokens and the keys are public.

We formally define the above three security notions based on the notion of indistinguishability, as is usually done for encryption schemes. Three experiments corresponding to the three attack scenarios discussed above are described in Figure 2. Each experiment represents an interaction between a challenger and an adversary $\mathcal{A}$. The challenger can be seen as the tokenization system, which in the beginning selects a random key from the key space and instantiates the tokenizer with the selected key. Then (in lines 3 to 6 of the experiments), the challenger responds to the queries of the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ in each case queries with $(x,d) \in \mathcal{X} \times \mathcal{D}$, i.e., it asks for the outputs of the tokenizer for pairs of PAN and associated data of its choice. Finally, $\mathcal{A}$ submits two pairs of PANs and associated data (different from the ones already queried) to the challenger. The challenger selects one of the pairs uniformly at random and provides $\mathcal{A}$ with the tokenizer output for the selected pair. The task of $\mathcal{A}$ is to tell which pair was selected by the challenger. If $\mathcal{A}$ can correctly guess the selection of the challenger then the experiment outputs a 1 otherwise it outputs a 0. This setting is very similar to the way in which security of encryption schemes are defined for a chosen plaintext adversary.

$\mathsf{TKR1}_k(x,d)$	$\mathsf{DTKR1}_k(x,d)$
1. $t \leftarrow \mathsf{FP}_k(d,x)$;	1. $x \leftarrow \mathsf{FP}_k^{-1}(d,x)$;
2. **return** $(t, NULL)$	2. **return** x

Figure 3: The TKR1 tokenization scheme using a format preserving encryption scheme FP.

The three experiments differ in what the adversary gets to see. In experiment Exp-IND-TKR$^{\mathcal{A}}$, $\mathcal{A}$, in response to its queries gets only the tokens and in Exp-IND-TKR-CV$^{\mathcal{A}}$ it gets both the tokens and the data that is stored in the card vault. In Exp-IND-TKR-KEY$^{\mathcal{A}}$, $\mathcal{A}$ gets the tokens corresponding to its queries, and the challenger reveals the key to $\mathcal{A}$ after the query phase.

Definition 1. *Let* $\mathsf{TKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{X} \times \mathcal{D} \to \mathcal{T} \times \mathbb{C}$ *be a tokenizer. Then the advantage of an adversary* $\mathcal{A}$ *in the sense of* IND-TKR, IND-TKR-CV *and* IND-TKR-KEY *are defined as*

$$\mathbf{Adv}^{\text{ind-tkr}}_{\mathsf{TKR}}(\mathcal{A}) = \left|\Pr[\text{Exp-IND-TKR}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|,$$

$$\mathbf{Adv}^{\text{ind-tkr-cv}}_{\mathsf{TKR}}(\mathcal{A}) = \left|\Pr[\text{Exp-IND-TKR-CV}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|,$$

$$\mathbf{Adv}^{\text{ind-tkr-key}}_{\mathsf{TKR}}(\mathcal{A}) = \left|\Pr[\text{Exp-IND-TKR-KEY}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|.$$

In the following two sections we discuss two class of constructions for tokenizers. The first construction TKR1, is the trivial way to do tokenization using FPE. In the constructions TKR2 and a variant TKR2a, presented in Section 6, our main aim is to by-pass the use of FPE schemes and use standard cryptographic schemes to achieve both security and the format requirements for arbitrarily formatted PANs/tokens.

5 CONSTRUCTION TKR1: TOKENIZATION USING FPE

The construction TKR1 is described in Figure 3. TKR1 uses an FPE scheme $\mathsf{FP} : \mathcal{K} \times \mathcal{D} \times \mathcal{X} \to \mathcal{T}$ in an obvious way to generate tokens. As FP is a permutation on $\mathcal{X}$, hence we assume that $\mathcal{T} = \mathcal{X}$.

For security we assume that $\mathsf{FP}_k()$ is a tweakable pseudorandom permutation with a tweak space $\mathcal{D}$ and message space $\mathcal{T}$. Note, that this scheme does not utilize a card vault and thus is stateless. The scheme is

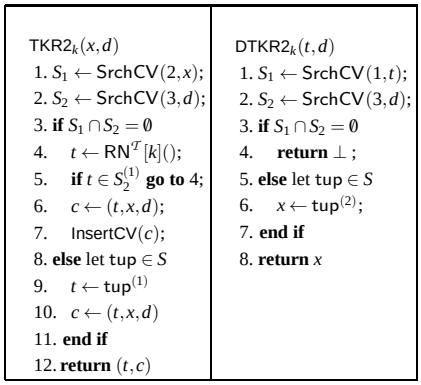

Figure 4: The TKR2 tokenization scheme using a random number generator $\mathsf{RN}^{\mathcal{T}}()$.

secure both in terms of IND-TKR and IND-TKR-CV. We formally state the security in the following theorem.

Theorem 1. *1. Let $\Psi = \mathsf{TKR1}$ be defined as in figure 3, and $\mathcal{A}$ be an adversary attacking Ψ in the* IND-TKR *sense. Then there exists a $\widetilde{\text{prp}}$ adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{ind-tkr}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\text{prp}}}_{\mathsf{FP}}(\mathcal{B}),$$

where $\mathcal{B}$ uses almost the same resources as of $\mathcal{A}$.

2. Let $\Psi = \mathsf{TKR1}$ be defined as in figure 3, and $\mathcal{A}$ be an adversary attacking Ψ in the IND-TKR-CV *sense. Then there exists a $\widetilde{\text{prp}}$ adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{ind-tkr-cv}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\text{prp}}}_{\mathsf{FP}}(\mathcal{B}).$$

The first claim of the Theorem is an easy reduction where we design a $\widetilde{\text{prp}}$ adversary $\mathcal{B}$ which runs $\mathcal{A}$ and finally relate the advantages of the adversaries $\mathcal{A}$ and $\mathcal{B}$. The second claim directly follows from the first, as in the construction TKR1, there is no card vault, thus an IND-TKR-CV adversary for TKR1 does not have any additional information compared to an IND-TKR adversary. The proofs can be found in the full version.

6 CONSTRUCTION TKR2: TOKENIZATION WITHOUT USING FPE

Here we propose a class of constructions which avoids the use of format preserving encryption. Instead of a permutation on $\mathcal{T}$ as we did for the previous construction, we assume a primitive $\mathsf{RN}^{\mathcal{T}}()$, which when invoked (ideally) outputs a uniform random element in $\mathcal{T}$. This primitive can be keyed, we will

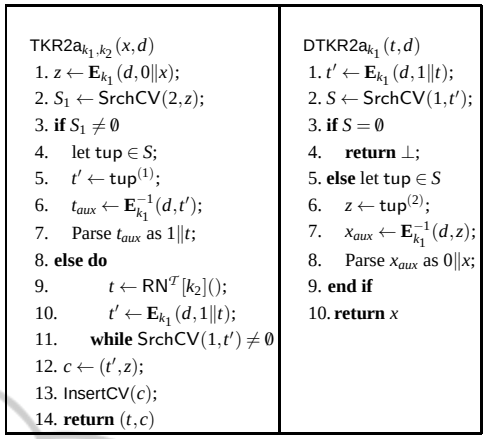

Figure 5: The TKR2a tokenization scheme.

denote this by $\mathsf{RN}^{\mathcal{T}}[k]()$, where k is a uniform random element of a pre-defined finite key space $\mathcal{K}$. We define the rnd advantage of an adversary $\mathcal{A}$ attacking $\mathsf{RN}^{\mathcal{T}}()$ as

$$\mathbf{Adv}^{\text{rnd}}_{\mathsf{RN}}(\mathcal{A}) = \left|\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{RN}^{\mathcal{T}}[k]()} \Rightarrow 1] - \Pr[\mathcal{A}^{\$^{\mathcal{T}}()} \Rightarrow 1]\right|. \quad (1)$$

Where $\$^{\mathcal{T}}()$ is an oracle which returns uniform random strings from $\mathcal{T}$. The task of a rnd adversary $\mathcal{A}$ is to distinguish between $\mathsf{RN}^{\mathcal{T}}[k]()$ and its ideal counterpart when oracle access to these schemes are given to $\mathcal{A}$.

We describe a generic scheme for tokenization in Figure 4, which we call as TKR2 that uses $\mathsf{RN}^{\mathcal{T}}()$. We consider that the card-vault CV is a collection of tuples with 3 components (x_1,x_2,x_3), where x_1,x_2,x_3 are the token, the PAN and associated data respectively. For a tuple $\mathsf{tup} = (x_1,x_2,x_3)$, we would use $\mathsf{tup}^{(i)}$ to denote x_i. Given a card-vault CV we also assume procedures to search for tuples in the CV. $\mathsf{SrchCV}(i,x)$ returns those tuples tup in CV such that $\mathsf{tup}^{(i)} = x$. If S be a set of tuples, then by $S^{(i)}$ we will denote the set of the i-th components of the tuples in S. As it is evident from the description in Figure 4, the detokenization operation is made possible through the data stored in the card vault, and the detokenization is just a search procedure. Also, the determinism is assured by search.

It is easy to see that TKR2 is not secure in the IND-TKR-CV sense, as in the card vault the PANs are stored in clear. To achieve security in terms of IND-TKR-CV, any CPA secure encryption can be used to encrypt the PANs stored in the card vault. We modify TKR2 to TKR2a to achieve this.

Modifying TKR2 to TKR2a: For this modification, we consider that the CV is a collection of tuples with

two components (x_1, x_2), where x_1 is the encrypted token and x_2 is the encrypted PAN. We additionally use a deterministic CPA secure encryption (supporting associated data) scheme $\mathbf{E} : \mathbb{K} \times \mathcal{D} \times \mathcal{X} \to \mathbb{C}$, with key space $\mathcal{K}$, tweak (associated data) space $\mathcal{D}$ and message space $\mathcal{X}$. Note that it is not required that $\mathbb{C} = \mathcal{X}$, as the ciphertexts would only be stored in the card vault. The tokenization scheme TKR2a described in Figure 5 uses the objects described above.

Security of TKR2 and TKR2a. The following two theorems specify the security of TKR2 and TKR2a.

Theorem 2. *Let $\Psi \in \{\mathsf{TKR2}, \mathsf{TKR2a}\}$ and $\mathcal{A}$ be an adversary attacking Ψ in the* IND-TKR *sense. Then there exists a RND adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{ind-tkr}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{rnd}}_{\mathsf{RN}}(\mathcal{B})$$

Theorem 3. *Let $\Psi = \mathsf{TKR2a}$ and $\mathcal{A}$ be an adversary attacking Ψ in the* IND-TKR-CV *sense, who asks at most q queries. Then there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$ (which use almost the same resources as of $\mathcal{A}$) such that*

$$\begin{aligned}\mathbf{Adv}^{\text{ind-tkr-cv}}_{\Psi}(\mathcal{A}) \quad &\leq \quad \mathbf{Adv}^{\text{rnd}}_{\mathsf{RN}}(\mathcal{B}) \\ &\quad + \mathbf{Adv}^{\text{det-cpa}}_{\mathbf{E}}(\mathcal{B}') + \frac{(2q+1)^2}{2^s}\end{aligned}$$

where s is the size of the smallest element in $\mathbb{C}$.

Finally it is important to note that TKR2 and TKR2a achieve security in the IND-TKR-KEY sense, when we instantiate $\mathsf{RN}^{\mathcal{T}}()$ with a true random number generator (TRNG). We can easily see this, considering that a TRNG is keyless, thus we have the property of independence between the tokens and the keys.

7 CONCLUSION

We studied the problem of tokenization from a cryptographic viewpoint. We proposed a syntax for the problem and also formulated three different security definitions. These new definitions may help in analyzing existing tokenization systems. We also proposed three constructions for tokenization: TKR1, TKR2 and a TKR2a. The last two constructions are particularly interesting, as they demonstrate that tokenization can be achieved without the use of format preserving encryption. Also we analyzed all the constructions in light of our security definitions.

More details about instantiations of the schemes along with their security, efficiency properties and experimental results can be found in the extended version.

ACKNOWLEDGEMENTS

The authors acknowledge the support from CONACYT project 166763.

REFERENCES

Bellare, M., Ristenpart, T., Rogaway, P., and Stegers, T. (2009). Format-preserving encryption. In Jr., M. J. J., Rijmen, V., and Safavi-Naini, R., editors, *Selected Areas in Cryptography*, volume 5867 of *Lecture Notes in Computer Science*, pages 295–312. Springer.

Brier, E., Peyrin, T., and Stern, J. (2010). BPS: a format-preserving encryption proposal. NIST submission. Available at http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf.

Halevi, S. and Rogaway, P. (2004). A parallelizable enciphering mode. In Okamoto, T., editor, *CT-RSA*, volume 2964 of *Lecture Notes in Computer Science*, pages 292–304. Springer.

Hoang, V. T., Morris, B., and Rogaway, P. (2012). An enciphering scheme based on a card shuffle. In Safavi-Naini, R. and Canetti, R., editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 1–13. Springer.

Morris, B., Rogaway, P., and Stegers, T. (2009). How to encipher messages on a small domain. In Halevi, S., editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 286–302. Springer.

PCI Security Standards Council (2008). Payment card industry data security standard version 1.2. Available at https://www.pcisecuritystandards.org/security_standards/pci_dss.shtml.

PCI Security Standards Council (2011). Information supplement: PCI DSS tokenization guidelines. Available at https://www.pcisecuritystandards.org/documents/Tokenization_Guidelines_Info_Supplement.pdf.

RSA White paper (2012). Tokenization: What next after PCI. Available at http://www.emc.com/collateral/white-papers/h11918-wp-tokenization-rsa-dpm.pdf.

Securosis White Paper (2011). Tokenization guidance: How to reduce pci compliance costs. Available at http://gateway.elavon.com/documents/Tokenization_Guidelines_White_Paper.pdf.

Stefanov, E. and Shi, E. (2012). Fastprp: Fast pseudo-random permutations for small domains. *IACR Cryptology ePrint Archive*, 2012:254.

Voltage Security White paper (2012). Payment security solution - processor edition. Available at http://www.voltage.com/wp-content/uploads/Voltage_White_Paper_SecureData_PaymentsProcessorEdition.pdf.