A Cryptographic Study of Tokenization Systems

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 theft of this data is a serious threat to any organization. Any organization that stores such data needs to achieve payment card industry (PCI) compliance, which is an intricate process where the organization needs to demonstrate that the data it stores is safe. Recently there has been a paradigm shift in treatment of the problem of storage of payment card information. The problem is solved by a new paradigm called "tokenization", where instead of the real credit card data a token is stored. The token resembles the credit/debit card number but is in no way related to it. This solution relieves the merchant from the burden of PCI compliance in several ways. Though tokenization systems are heavily in use, but to our knowledge, a formal cryptographic study of this problem has not yet been done. In this paper we initiate a study in this direction. We formally define the syntax of a tokenization system, and try to define 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 popular 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 is considered to be one of the most serious threats to any business. Such a breach not only amounts to a serious financial loss to the business but also a critical damage to the "brand image" of the company in question.

The Payment Card Industry Security Standard Council (PCI SSC), which was founded by the major payment card brands, 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) which specifies security mechanisms required to secure payment card data. PCI DSS dictates that organizations, which process card payments, must protect cardholder data when they store, transmit and process them. In summary it mandates that the credit card number must remain unreadable anywhere it is stored, and it proposes to use "strong cryptography" as a possible solution. The actual requirements specified by PCI DSS are elaborate and complex. To obtain PCI compliance, a merchant needs to provide documentation on the usage and security policies regarding *all* sensitive information stored in its environment. PCI compliance is considered to be necessary for any business to acquire the confidence of its customers, moreover a business which has suffered theft of sensitive information while not being compliant can be subject to hefty amounts of fines from the government in some countries.

Traditionally credit card numbers have been used as a primary identifier in many business processes in the merchant sites. We quote from a document by Securosis (Securosis White Paper, 2011b):

> *As the standard reference key, credit card numbers are stored in billing, order management, shipping, customer care, business intelligence, and even fraud detection systems. Large retail organizations typically store credit card data in every critical business processing system.*

Thus, in merchant sites, credit card numbers are scattered across their environment. This makes it very difficult for a merchant to formulate security policies and provide necessary documentation to obtain PCI compliance.

But, in most systems where credit card numbers are stored, the data itself is not required, and the system would function as well as before if the credit card numbers are replaced by some other information which would "look like" credit card numbers. This

observation has lead to a paradigm shift in the way security of credit card numbers are viewed: instead of securing sensitive data wherever it is present it is easier to remove sensitive data from where it is not required. This basic paradigm has been implemented using *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 try to popularize tokenization and discuss about the possible solutions to the tokenization problem (Securosis White Paper, 2011a; Securosis White Paper, 2011b; Voltage Security White paper, 2012; RSA White paper, 2012). PCI SSC has also formulated its guidelines regarding *tokenization* (PCI Security Standards Council, 2011). But to our knowledge a formal cryptographic analysis of the problem has not been done. Even it is not clear what basic cryptographic objects should be used and in what way, to achieve the goals of tokenization.

SMALL DOMAIN ENCRYPTION. One obvious solution for securing credit card numbers in a merchant site is to encrypt them. But as we stated, a typical merchant site heavily depends on the credit card numbers for its functioning, even it index data based on credit card numbers. Hence, a strict requirement for applying encryption is that the cipher should look like a credit card number, so that for using encryption one does not require to change the database fields where these numbers are stored. 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. This is much less than the block size of a typical block cipher (say AES). Thus, direct application of a block cipher to encrypt would result in a considerable length expansion, and it would not be possible to encode the cipher into sixteen decimal digits.

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. Formally, if we consider $\mathcal{X}$ to be a message space which contains strings from an arbitrary alphabet satisfying certain format, $\mathcal{D}$ and $\mathcal{K}$ be finite sets called the tweak space and key space respectively, then a format preserving encryption scheme is formally defined to be a function $\mathsf{FP} : \mathcal{K} \times \mathcal{D} \times \mathcal{X} \to \mathcal{X}$, such that for every $d \in \mathcal{D}$ and $K \in \mathcal{K}$, $\mathsf{FP}_K(d,\cdot) : \mathcal{X} \to \mathcal{X}$ is a permutation. And FP should provide security as that of a tweakable strong pseudorandom permutation (SPRP). Designing such schemes for arbitrary $\mathcal{X}$ is a challenging and interesting problem. In particular given a SPRP on $\{0,1\}^n$, designing a SPRP for a message space $\{0,1\}^t$, where $t < n$ is difficult. There have been some interesting solutions to this problem, but none of them can be considered to be efficient (Bellare et al., 2009; Bellare et al., 2010; Brier et al., 2010; Hoang et al., 2012; Morris et al., 2009; Stefanov and Shi, 2012).

A credit card number encrypted by a FPE scheme can act as a token. Such a solution is also provided by Voltage Security (Voltage Security White paper, 2012). To the best of our knowledge, this is the only solution to the tokenization problem with known cryptographic guarantees. But again, there does not exist a formal security model for tokenization, and it has been contested that a token which is an encryption of the credit card data may not be considered as a safe token as there exists a possibility that the token can be inverted to get the original data (Securosis White Paper, 2011b).

OUR CONTRIBUTION. We study the problem of tokenization from a cryptographic viewpoint, the main contributions of this work can be summarized as follows. We point out the basic needs for a tokenization system, and develop a syntax for the problem. The syntax follow closely the recommendation of the PCI SSC, and is general enough to accommodate various implementation options. Further, we develop a security model for the problem 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. We amply discuss the adequacy of these new notions of security in practical scenarios.

Finally, we propose some constructions of tokenization systems, and prove their security in the proposed security models. We propose three generic constructions namely TKR1, TKR2 and TKR2a and discuss how these constructions can be instantiated with available cryptographic primitives. TKR1 is a construction which just uses a format preserving encryption to generate tokens. TKR2 and TKR2a are similar but both are very different from TKR1. In the constructions TKR2 and TKR2a we demonstrate how the problem of tokenization can be solved both securely and efficiently without using FPE. Both TKR2 and TKR2a uses off the shelf cryptographic primitives, in particular we show how to instantiate them using ordinary block ciphers, stream ciphers supporting initialization vectors (IV) and physical random number generators. We also prove security of our constructions in the proposed security models. For lack of space, in this version we do not provide the proofs of the stated theorems and propositions. The proofs are mostly straightforward and follows from standard reductionist arguments, but as is always the case for such proofs, they are lenghty. We will provide the full

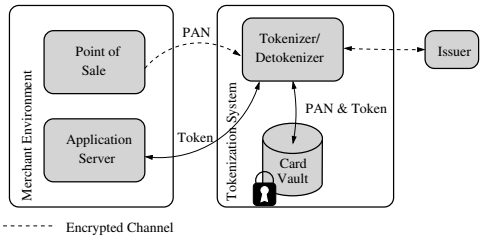

Figure 1: Architecture of the tokenization system

proofs in the extended version which would be made available 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, this is true in most situations today, as the merchants receive the service of tokenization from a third party. But it is also possible that the merchant itself implements its tokenization 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). In (PCI Security Standards Council, 2011) there is no specific recommendation regarding how this process should be implemented. Some of the mentioned options are encryption functions, cryptographic hash functions and random number generators.
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 data involved in token generation.

The PCI guidelines for tokenization are quite vague (this has been pointed out before in many places including (Securosis White Paper, 2011a)), and it is difficult to make out what properties tokens and tokenization systems should posses for functionality and security. 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. It has been said that in some scenarios it may be important that the tokens can be easily distinguished from that of the PANs. For example, most credit card numbers have a Luhn checksum (ISO/IEC 7812-1, 2006) of zero. One can make tokens containing same number of digits as that of the PAN but the Luhn checksum should be 1. Such a distinguishing criteria may make audits easier.
2. **Uniqueness:** The token generation method should be deterministic. As stated before, the application software in the merchant side uses the PAN for indexing, 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.

3 Cryptographic Preliminaries and Notations

GENERAL NOTATIONS. The set of all n bit strings would be denoted by $\{0,1\}^n$. We shall sometimes consider strings over an arbitrary alphabet AL, for $Y \in \mathsf{AL}^*$, by $|Y|_{\mathsf{AL}}$ we will denote the number of characters in the string Y. If $\mathsf{AL} = \{0,1\}$, and X is a string

over AL, then we will use $|X|$ to denote the length of X in bits. If A is a finite set, then $\#A$ will denote the cardinality of A. If X, Y are strings, $X||Y$ will denote the concatenation of X and Y. For $A \in \{0,1\}^*$, $\mathsf{format}_n(X) = X_1||X_2||\ldots||X_m$, where $|X_i| = n$, for $1 \leq i \leq m-1$ and $0 \leq |X_m| < n$. If $X \in \{0,1\}^*$ is such that $|X| \geq \ell$, then $\mathsf{take}_\ell(X)$ will denote the ℓ most significant bits of X. For a non negative integer $i \leq 2^n - 1$, $\mathsf{bin}_n(i)$ will denote the n bit binary representation of i, and for any n-bit string X, $\mathsf{toInt}(X)$ will denote the integer represented by the string X.

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 bit b. $\mathcal{A}^O \Rightarrow b$, will denote the fact that an adversary $\mathcal{A}$ has access to an oracle O and outputs b. In general an adversary would have other sorts of interactions, maybe with other adversaries and/or algorithms before it outputs, these would be clear from the context. Whenever we refer to resources of an adversary we would mean: the number of oracle queries made by it and its running time.

PSEUDORANDOM FUNCTIONS AND PERMUTATIONS. For finite sets A and B, by $\mathsf{Func}(A,B)$ we would mean the set of all functions mapping A to B, and $\mathsf{Perm}(A)$ would denote the set of all permutations on A (i.e., all bijective functions mapping A to A). If $A = \{0,1\}^n$ and $B = \{0,1\}^\ell$, then we would denote $\mathsf{Func}(A,B)$ by $\mathsf{Func}(n,\ell)$ and $\mathsf{Perm}(A)$ by $\mathsf{Perm}(n)$.

Consider the map $F : \mathbb{K} \times \mathbb{D} \to \mathbb{R}$ where $\mathbb{K}, \mathbb{D}, \mathbb{R}$ (commonly called keys, domain and range respectively) are all non-empty and $\mathbb{K}$ and $\mathbb{R}$ are finite. We view this map as representing a *family of functions* $F = \{F_K\}_{K \in \mathbb{K}}$, i.e., for each $K \in \mathbb{K}$, F_K is a function from $\mathbb{D}$ to $\mathbb{R}$ defined as $F_K(X) = F(K,X)$. For every $K \in \mathbb{K}$, we call F_K to be an instance of the family F.

Let $F : \mathbb{K} \times \mathbb{D} \to \mathbb{R}$ be a family of functions. We define the prf-advantage of an adversary $\mathcal{A}$ in breaking F as

$$\mathbf{Adv}^{\mathrm{prf}}_F(\mathcal{A}) = \Big|\Pr[K \xleftarrow{\$} \mathbb{K} : \mathcal{A}^{F_k(\cdot)} \Rightarrow 1] - \Pr[\rho \xleftarrow{\$} \mathsf{Func}(\mathbb{D},\mathbb{R}) : \mathcal{A}^{\rho(\cdot)} \Rightarrow 1]\Big|.$$

Similarly, if $E : \mathbb{K} \times \mathbb{D} \to \mathbb{D}$ is a family of permutations, we define the prp-advantage of an adversary $\mathcal{A}$ in breaking E as

$$\mathbf{Adv}^{\mathrm{prp}}_E(\mathcal{A}) = \Big|\Pr[K \xleftarrow{\$} \mathbb{K} : \mathcal{A}^{E_K(\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}(\mathbb{D}) : \mathcal{A}^{\pi(\cdot)} \Rightarrow 1]\Big|.$$

A *tweakable enciphering scheme (TES)* is a function $\mathcal{E} : \mathbb{K} \times \mathbb{T} \times \mathcal{M} \to \mathcal{M}$ where $\mathbb{K}$ is the key space, $\mathbb{T}$ is the tweak set, and $\mathcal{M}$ is the message space and for every $K \in \mathbb{K}$ and $T \in \mathbb{T}$ we have that $\mathcal{E}(K,T,\cdot) = \mathcal{E}^T_K(\cdot)$ is a length preserving permutation. We define the $\widetilde{\mathrm{prp}}$ advantage of an adversary $\mathcal{A}$ as

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathcal{E}}(\mathcal{A}) = \Big|\Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] - \Pr[\pi \xleftarrow{\$} \mathsf{Perm}^{\mathbb{T}}(n) : \mathcal{A}^{\pi(\cdot,\cdot)} \Rightarrow 1]\Big|,$$

where $\mathsf{Perm}^{\mathbb{T}}(\mathcal{M})$ is the set of length preserving tweak indexed permutations on $\mathcal{M}$. If the message space $\mathcal{M} = \{0,1\}^n$, then $\mathcal{E}$ is called a tweakable block cipher.

DETERMINISTIC CPA SECURE ENCRYPTION. Let $\mathbf{E} : \mathbb{K} \times \mathbb{T} \times \mathcal{M} \to \mathbb{C}$ be a deterministic encryption scheme with key space $\mathbb{K}$, tweak space $\mathbb{T}$, message space $\mathcal{M}$ and cipher space $\mathbb{C}$. We define the det-cpa advantage of any adversary $\mathcal{A}$, which does not repeat any query as

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

where $\$(.,.)$ is an oracle, which on input $(d,x) \in \mathbb{T} \times \mathbb{M}$ returns a random string of the size of the ciphertext of x.

4 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 [1].
4. CV, the card vault. The card vault is a repository where PAN's and tokens are stored, which may have a special structure for the ease of implementation of the token mapping procedure. In our syntax we shall use the CV to represent a state of the

[1] In our view, irrespective of other possible identifiers, the associated data should contain an identifier of the merchant. Thus if $d, d' \in \mathcal{D}$ are two associated data related to two different merchants, it should be that $d \neq d'$. For our notion of correctness this requirement for the associated data would be required.

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 is the procedure responsible for generating tokens from the PANs. We consider the tokenizer 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 t and also changes the CV. Thus, other than t, TKR also produces an element from $\mathbb{C}$ which is used to update the CV. We use the square brackets to denote this interaction. We formally see TKR as a function $\mathsf{TKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{X} \times \mathcal{D} \rightarrow \mathcal{T} \times \mathbb{C}$. 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. As in case of tokenizer, we denote a detokenizer as a function $\mathsf{DTKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{T} \times \mathcal{D} \rightarrow \mathcal{X} \cup \{\perp\}$. 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, we do not disallow the case where two different PANs with different associated data have the same tokens. This requirement is clear if we consider the associated data d to be an identifier for a merchant. 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.

5 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 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.

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

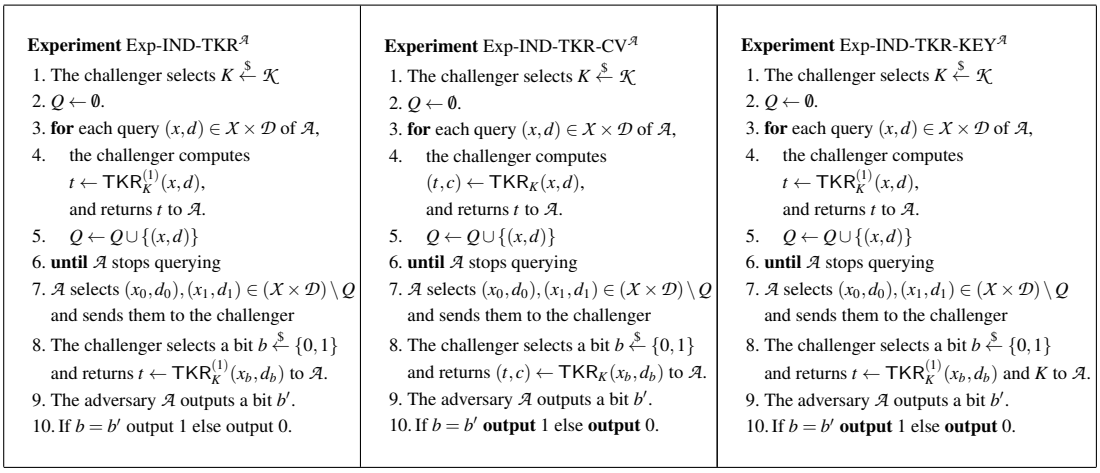

Figure 2: Experiments used in the security definitions: IND-TKR, IND-TKR-CV and IND-TKR-KEY

KEY *are defined as*

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

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

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

respectively.

From the definitions, it is obvious that IND-TKR-CV $\Longrightarrow$ IND-TKR and IND-TKR-KEY$\Longrightarrow$IND-TKR, but IND-TKR $\not\Longrightarrow$ IND-TKR-CV and IND-TKR$\not\Longrightarrow$IND-TKR-KEY. Thus IND-TKR-CV and IND-TKR-KEY are strictly stronger than IND-TKR.

ADEQUACY OF THE NOTIONS. We discuss some of the characteristics and limitations of the proposed definitions next.

1. IND-TKR refers to the basic security requirement for tokens. It adheres to the informal security notion for tokens as stated in the PCI DSS guideline for tokenization. It models the fact that tokens and PANs are un-linkable in a computational sense, if the key and card-vaults are kept secret. Thus, if a merchant adopts a tokenization scheme provided by a third party, which is secure in the IND-TKR sense then this will probably relieve it from PCI compliance. As in this case the merchant does not own the card-vault or the keys, and the burden of security involved with the keys and the card vault lies with the provider who offers the tokenization service.
2. The IND-TKR-CV is a stronger notion. If a tokenization system achieves this security, then it implies that tokens and PANs are un-linkable even with the knowledge of the card-vault. This in turn implies that the contents of the card-vault are not useful (in a computational sense) to derive a relation between PANs and tokens. Thus, it provides security both to the tokenization service provider and the merchant who use this service.
3. IND-TKR-KEY is a stronger form of the IND-TKR notion. Some public documents like (Securosis White Paper, 2011b) it has been stressed that encryption is not a good option for tokenization, as in theory there exists the possibility that a token can be inverted to obtain the PAN. If tokens are generated using a "secure" encryption scheme, then it is infeasible for any "reasonably efficient" adversary to invert the token without the knowledge of the key. But, this computational guarantee does not seem to be enough for users. The IND-TKR-KEY definition aims to model this paranoid situation, where linking the PANs with tokens becomes infeasible even with the knowledge of the key. Note in IND-TKR-KEY we still assume that the card vault is inaccessible to an adversary.
4. All the definitions follow the style of a chosen plaintext attack. The definitions may be made stronger by giving the adversary additional power of obtaining PANs corresponding to tokens of its choice. But in this application, we think such stronger notions are not applicable.

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. The other constructions (TKR2 and a variant TKR2a) presented in Section 7 are very different. For the later constructions our main aim is to bypass the use of FPE schemes and use standard cryptographic schemes along with some encoding mecha-

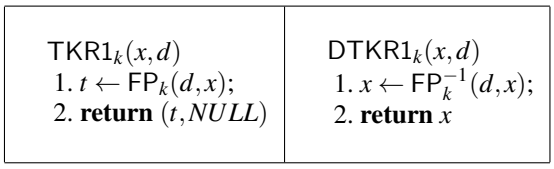

Figure 3: The TKR1 tokenization scheme using a format preserving encryption scheme FP.

nism to achieve both security and the format requirements for arbitrarily formatted PANs/tokens.

6 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, assuming 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 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 prp adversary $\mathcal{B}$ such that*

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

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 prp adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that*

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

The first claim of the Theorem is an easy reduction where we design a 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 do not have any additional information compared to an IND-TKR adversary. The proofs would be provided in the full version.

This scheme can be instantiated using any format preserving encryption scheme as described in (Bellare et al., 2009; Brier et al., 2010; Bellare et al., 2010; Morris et al., 2009; Hoang et al., 2012; Stefanov and Shi, 2012). We discuss more on the impact of security and efficiency for specific instantiations in Section 8.

7 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 $\mathcal{T}$ which we use 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, which we will denote by $\mathsf{RN}^{\mathcal{T}}[k]()$, where k is a uniform random element of a pre-defined finite key space $\mathcal{K}$. $\mathsf{RN}^{\mathcal{T}}()$ can also be realized by using a keyed cryptographic primitive f_k, such instantiations would be more specifically denoted by $\mathsf{RN}^{\mathcal{T}}[f_k]()$. 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 $\mathsf{RN}^{\mathcal{T}}()$. For the description we consider that the card-vault CV is a collection of tuples, where each tuple has 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 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.

Correctness: A limitation of the TKR2 scheme is that it may violate the property of uniqueness. It is not guaranteed that $\mathsf{TKR2}_k(x,d) \neq \mathsf{TKR2}_k(x',d')$ when $(x,d) \neq (x',d')$. As discussed before, for practical purposes what is required is a weak form of uniqueness, i.e., for $x \neq x'$, for any $d \in \mathcal{D}$, $\mathsf{TKR2}(x,d) \neq \mathsf{TKR2}(x',d)$. This requirement stems from the fact that a specific merchant with associated data d, may use the tokens as a primary key in its databases. Thus if $d \neq d'$, it can be tolerated that $\mathsf{TKR2}(x,d) = \mathsf{TKR2}(x',d')$, for any $x, x' \in \mathcal{X}$.

$\mathsf{TKR2}_k(x,d)$	$\mathsf{DTKR2}_k(t,d)$
1. $S_1 \leftarrow \mathsf{SrchCV}(2,x)$;	1. $S_1 \leftarrow \mathsf{SrchCV}(1,t)$;
2. $S_2 \leftarrow \mathsf{SrchCV}(3,d)$;	2. $S_2 \leftarrow \mathsf{SrchCV}(3,d)$;
3. **if** $S_1 \cap S_2 = \emptyset$	3. **if** $S_1 \cap S_2 = \emptyset$
4. $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k]()$;	4. **return** $\perp$;
5. $c \leftarrow (t,x,d)$;	5. **else** let $\mathsf{tup} \in S$
6. $\mathsf{InsertCV}(c)$;	6. $x \leftarrow \mathsf{tup}^{(i)}$;
7. **else** let $\mathsf{tup} \in S$	7. **end if**
8. $t \leftarrow \mathsf{tup}^{(1)}$	8. **return** x
9. $c \leftarrow (t,x,d)$	
10. **end if**	
11.**return** (t,c)	

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

Let us assume that $\mathsf{RN}^{\mathcal{T}}()$ behaves ideally. If q unique tokens have been already generated with a specific associated data d, the probability that the $(q+1)^{th}$ token (generated with associated data d) is equal to any of the q previously generated tokens is given by $q/\#\mathcal{T}$. Thus, this probability of collision increases with the number of tokens already generated. If the total number of tokens generated by the tokenizer for a specific associated data is much smaller than the size of the token space (which will be the case in a practical scenario) this probability of collision would be insignificant [2]. But, still the uniqueness can be guaranteed by an additional search as shown in Figure 5. Where $\mathsf{RN}^{\mathcal{T}}()$ is repeatedly invoked unless a token different from one already produced is obtained. Following the previous discussion if q is small compared to $\#\mathcal{T}$, the expected number of repetitions required until an unique token is obtained would be small. The detokenization corresponding to the modified tokenization scheme described in Figure 5 remains the same as described in Figure 4.

We formally specify the security of TKR2 later in this section, but 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, hence if the card vault is revealed then no security remains. This can be fixed by encrypting the tokens in the card vault. To achieve security in terms of IND-TKR-CV, any CPA secure encryption can be used to encrypt the PANs stored in

[2]According to (CardHub, 2012) the total number of credit cards in 2012 from the four primary credit card networks (i.e. VISA, MasterCard, American Express, and Discover) was 546 millions ($\approx 2^{30}$). This can be considered as a reasonable upper bound for q. Assuming credit card numbers to be of 16 decimal digits, $\#\mathcal{T} = 10^{16} \approx 2^{53}$. These numbers leads to a collision probability of $1/2^{23}$ which is insignificant

$\mathsf{TKR2}_k(x,d)$
1. $S_1 \leftarrow \mathsf{SrchCV}(2,x)$;
2. $S_2 \leftarrow \mathsf{SrchCV}(3,d)$;
3. **if** $S_1 \cap S_2 = \emptyset$
4. $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k]()$;
5. **if** $t \in S_2^{(1)}$ **go to** 4;
6. $c \leftarrow (t,x,d)$;
7. $\mathsf{InsertCV}(c)$;
8. **else** let $\mathsf{tup} \in S$
9. $t \leftarrow \mathsf{tup}^{(1)}$
10. $c \leftarrow (t,x,d)$
11. **end if**
12.**return** (t,c)

Figure 5: Modified TKR2 to ensure uniqueness

$\mathsf{TKR2a}_{k_1,k_2}(x,d)$	$\mathsf{DTKR2a}_{k_2}(t,d)$
1. $z \leftarrow \mathbf{E}_{k_1}(d,x)$;	1. $S_1 \leftarrow \mathsf{SrchCV}(1,t)$;
2. $S_1 \leftarrow \mathsf{SrchCV}(2,z)$;	2. $S_2 \leftarrow \mathsf{SrchCV}(3,d)$;
3. $S_2 \leftarrow \mathsf{SrchCV}(3,d)$;	3. **if** $S_1 \cap S_2 = \emptyset$
4. **if** $S_1 \cap S_2 = \emptyset$	4. **return** $\perp$;
5. $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;	5. **else** let $\mathsf{tup} \in S$
6. **if** $t \in S_2^{(1)}$ **go to** 5;	6. $z \leftarrow \mathsf{tup}^{(i)}$;
7. $c \leftarrow (t,z,d)$;	7. $x \leftarrow \mathbf{E}_{k_2}^{-1}(d,z)$;
8. $\mathsf{InsertCV}(c)$;	8. **end if**
9. **else** let $\mathsf{tup} \in S$	9. **return** x
10. $t \leftarrow \mathsf{tup}^{(1)}$	
11. $c \leftarrow (t,z,d)$	
12. **end if**	
13.**return** (t,c)	

Figure 6: The TKR2a tokenization scheme.

the card vault, note that for the encrypted PAN to be stored in the card vault the format preserving requirement is not required. We modify TKR2 to TKR2a to achieve this. We discuss the details of TKR2 next.

Modifying TKR2 to TKR2a: For this modification, the structure of the card vault remains same as for TKR2, but in the first component of the tuples in the card vault, encrypted tokens are stored. We additionally use a deterministic CPA secure encryption (supporting associated data) scheme $\mathbf{E} : \mathbb{K} \times \mathcal{D} \times \mathcal{X} \rightarrow \mathbb{C}$, with key space $\mathcal{K}$, tweak (associated data) space $\mathcal{D}$ and message space $\mathcal{X}$. Note that the cipher space $\mathbb{C}$ can be arbitrary, i.e., it is not required that $\mathbb{C} = \mathcal{X}$, as the ciphers here would not be tokens but would be stored in the card vault. The tokenization scheme TKR2a described in Figure 6 uses the objects described above.

7.1 Realizing $\mathsf{RN}^{\mathcal{T}}[k]$

The keyed primitive $\mathsf{RN}^{\mathcal{T}}[k]$ can be realized by standard cryptographic primitives. We discuss here a specific realization which uses a pseudorandom function $f : \mathcal{K} \times \mathbb{Z}_N \to \{0,1\}^L$, where L and N are sufficiently "large", the exact requirements for N and L will become clear later. We call the construction $\mathsf{RN}[f_k]()$ and is shown in Figure 7.

For the construction shown in Figure 7, we assume that $\mathcal{T}$ contains strings of fixed length μ from an arbitrary alphabet AL. Let $\#\mathsf{AL} = \ell$, and $\lambda = \lceil \lg \ell \rceil$. Let $\sigma : \mathsf{AL} \to \{0, 2, \ldots, \ell - 1\}$ be a fixed bijection. The variable *cnt* can be considered as a state of the algorithm and it maintains its values across invocations. The basic idea behind the algorithm is to generate

$\mathsf{RN}[f_k]()$
1. $X \leftarrow f_k(cnt)$;
2. $X_1 || X_2 || \ldots || X_m \leftarrow \mathsf{format}_\lambda(X)$;
3. $Y \leftarrow \varepsilon$; (empty string)
4. $i \leftarrow 1$;
5. **while** $|Y|_{\mathsf{AL}} \neq \mu$,
6. **if** $\mathsf{toInt}(X_i) < \ell$,
7. $Y \leftarrow Y || \sigma^{-1}(\mathsf{toInt}(X_i))$;
8. $i \leftarrow i + 1$;
9. **end while**
10. $cnt \leftarrow cnt + 1$;
11. **return** Y;

Figure 7: Construction of $\mathsf{RN}()$ using a pseudorandom function $f_k()$.

a "long" binary string using $f_k(cnt)$ and divide the string into blocks of λ bits. If a block is less than ℓ then it is discarded otherwise it is accepted. The accepted blocks are encoded as elements in AL.

Choosing L and N: Let us define, $p = \Pr[y \xleftarrow{\$} \{0,1\}^\lambda : \mathsf{toInt}(y) < \ell] = \frac{\ell}{2^\lambda} > \frac{1}{2}$. Thus, if we assume that the output of $f_k()$ is uniformly distributed then an X_i passes the test in line 6 (of Figure 7) with probability p. Thus the expected number of times the while loop will run is at most 2μ. Thus, $L = 3\mu\lambda$, will be sufficient for all practical purposes.

Note that each invocation of $\mathsf{RN}[f_k]()$ increases the value of *cnt* by 1. Thus the value of N should be a conservative upper bound on the number of times $\mathsf{RN}[f_k]()$ needs to be invoked. $N = 2^{80} - 1$, should be sufficient for all practical purposes.

If f_k is a PRF then $\mathsf{RN}^{\mathcal{T}}[f_k]$ is secure in the RND sense. We formally state this security property in the following theorem.

Theorem 2. *Let $\mathcal{A}$ be an arbitrary adversary attacking $\mathsf{RN}[f_k]$ (as described in Figure 7) in the rnd sense. Then there exists a prf adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\mathrm{rnd}}_{\mathsf{RN}[f_k]()}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{prf}}_{f_k}(\mathcal{B}). \qquad (2)$$

This theorem asserts that as long as $f_k()$ is a PRF, the construction achieves the desired security in the rnd sense.

7.2 Candidates for $f_k()$:

$f_k()$ can be instantiated through standard symmetric key primitives. We discuss three options below:

1. *Stream cipher*: Modern stream ciphers, such as those in the eStream (Robshaw and Billet, 2008) portfolio, take as input a short secret key K and a short initialization vector (IV) and produce a "long" and random looking string of bits. Let $\mathsf{SC}_K : \{0,1\}^\ell \to \{0,1\}^L$ be a stream cipher with IV, i.e., for every choice of K from a certain pre-defined key space $\mathcal{K}$, SC_K maps an ℓ-bit IV to an output string of length L bits. The basic idea of security is that for a uniform random K and for distinct inputs $IV_1, \ldots, IV_q$, the strings $\mathsf{SC}_K(IV_1), \ldots, \mathsf{SC}_K(IV_q)$ should appear to be independent and uniform random to an adversary. This is formalized by requiring a stream cipher to be a PRF. See (Berbain and Gilbert, 2007) for further discussion on this issue. Thus, a stream cipher with the above security guarantees can be used to instantiate f_k.
2. *Block cipher*: A block cipher $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ can also be used to construct f_k as follows.

$f_k(cnt)$
1. $m \leftarrow \lceil L/n \rceil$;
2. $Y \leftarrow \mathsf{bin}_n(cnt)$;
3. $W \leftarrow E_k(Y)$;
4. $Z \leftarrow E_k(W) || E_k(W \oplus \mathsf{bin}_n(1)) || \cdots$
$\cdots || E_k(W \oplus \mathsf{bin}_n(m-1))$;
5. **return** $\mathsf{take}_L(Z)$

The above construction is same as the counter mode of operation, and if E_k is assumed to be a PRF then f_k as constructed above is also a PRF, in particular it is easy to verify the following holds

Proposition 1. *Let $\mathcal{B}$ be an arbitrary prf adversary attacking $f_k()$ who asks at most q queries, then one can construct a prf adversary $\mathcal{B}'$ for $E_K()$ such that, $\mathcal{B}'$ asks at most mq queries and*

$$\mathbf{Adv}^{\mathrm{prf}}_{f}(\mathcal{B}) \leq \mathbf{Adv}^{\mathrm{prf}}_{E}(\mathcal{B}') + \frac{m^2 q^2}{2^n}.$$

3. *True random number generator:* We end this discussion with another possible interesting instantiation of $\mathsf{RN}()$. The specific construction that we depicted in Figure 7 basically uses a stream of random bits generated through a pseudorandom function. Currently there has been a lot of interest in designing physical true random number generators. Such generators harvests entropy from its "environment" and generates random streams with some post processing. It has been claimed that such generators are "true random number generators" (TRNG). Such a generator can be used to design $\mathsf{RN}()$ as in Figure 7 by replacing $f_k()$ with a TRNG, and by selecting suitable blocks from the generated stream according to the format requirements of $\mathcal{T}$. As a TRNG is key-less, thus this would lead to a key-less construction of RN, we call such an instantiation as $\mathsf{RN}[\mathsf{TR}]$. As such a generator gives us "true randomness", hence for any adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{rnd}}_{\mathsf{RN}[\mathsf{TR}]} = 0$.

From now onwards, where it is necessary, we will denote $\mathsf{TKR2}$ instantiated with $\mathsf{RN}[f_k]$ and $\mathsf{RN}[\mathsf{TR}]$ by $\mathsf{TKR2}[f_k]$ and $\mathsf{TKR2}[\mathsf{TR}]$ respectively. Similar convention would be followed for $\mathsf{TKR2a}$.

7.3 Realizing $\mathbf{E}_k(d,x)$

As, discussed we require $\mathbf{E_k}(\cdot,\cdot)$ is used to encrypt the PAN, and the encryption is stored in the card vault within the tokenization system. We do not require this encryption to be format preserving. Here we discuss two instantiations of $\mathbf{E}$ using a secure block cipher E. If the block length of E is n, then both the proposed constructions have $\{0,1\}^n$ as their cipher space, and $\mathcal{X}$ and $\mathcal{D}$ as their message space and tweak space, respectively. For the constructions we assume some restrictions on $\mathcal{X}$ and $\mathcal{D}$, but these restrictions would be satisfied in most practical scenarios.

Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a block cipher. As we defined before, let $\mathcal{X}$ contain strings of fixed length μ from an arbitrary alphabet AL where $\#\mathsf{AL} = \ell$ and $\lambda = \lceil \lg \ell \rceil$. Let $\#\mathcal{D} = \ell_1$ and $\lambda_1 = \lceil \lg \ell_1 \rceil$. Let n_1 and n_2 be positive integers such that $n_1 \geq \mu\lambda$, $n_2 \geq \lambda_1$ and $n_1 + n_2 = n$. Note that for practical choice of AL, $\mathcal{D}$, μ and n, such n_1, n_2 can be selected. Let $\mathsf{pad}_{\mathcal{X}} : \mathcal{X} \to \{0,1\}^n$, $\mathsf{pad}_{\mathcal{D}} : \mathcal{D} \to \{0,1\}^n$, $\mathsf{pad}_1 : \mathsf{AL}^{\mu} \to \{0,1\}^{n_1}$ and $\mathsf{pad}_2 : \mathcal{D} \to \{0,1\}^{n_2}$ be injective functions.

The two different proposed instantiations of $\mathbf{E}$ are shown in Figure 8. Both the constructions uses a block cipher with a block length of n, and the padding functions defined above. In **E1**, the message x and the associated data d are suitably formatted to a n bit string and this formatted string is encrypted using the block cipher. **E2** is same as the construction of a tweakable block cipher proposed in (Liskov et al., 2002). If E_K is a secure block cipher in the prf sense then both $\mathbf{E1}_K$ and $\mathbf{E2}_K$ are det-cpa secure, we state this formally next.

Proposition 2. *Let $\mathcal{A}$ be an arbitrary det-cpa adversary attacking* **E1***, who asks at most q queries, never repeats a query, and runs for time at most T, then there exists a prf adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{det-cpa}}_{\mathbf{E1}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{E}(\mathcal{B}),$$

and $\mathcal{B}$ also asks exactly q queries and runs for time $O(T)$.

Proposition 3. *Let $\mathcal{A}$ be an arbitrary det-cpa adversary attacking* **E2***, who asks at most q queries, never repeats a query, and runs for time at most T, then there exists a prf adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{det-cpa}}_{\mathbf{E2}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{E}(\mathcal{B}) + \frac{q^2}{2^n},$$

and $\mathcal{B}$ also asks exactly q queries and runs for time $O(T)$.

The above propositions suggests that **E1** has a better security bound compared to **E2**, and for **E2** two block cipher calls are required for each encryption, whereas only a single block cipher call is required for **E1**. The formatting requirements are more stringent for **E1**, where as **E2** can be applied to any message space $\mathcal{X}$ and tweak space $\mathcal{D}$, where $\#\mathcal{X} \leq 2^n$ and $\#\mathcal{D} \leq 2^n$.

7.4 Security of $\mathsf{TKR2}$ and $\mathsf{TKR2a}$

The following three theorems specify the security of $\mathsf{TKR2}$ and $\mathsf{TKR2a}$.

Theorem 3. *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 4. *Let $\Psi = \mathsf{TKR2a}$ and $\mathcal{A}$ be an adversary attacking Ψ in the* IND-TKR *sense. Then there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$ (which use almost the same resources as of $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\text{ind-tkr-cv}}_{\Psi}(\mathcal{A}) \leq \mathbf{Adv}^{\text{rnd}}_{\mathsf{RN}}(\mathcal{B}) + \mathbf{Adv}^{\text{det-cpa}}_{\mathbf{E}}(\mathcal{B}')$$

Theorem 5. *Let $\Psi \in \{\mathsf{TKR2}[\mathsf{TR}], \mathsf{TKR2a}[\mathsf{TR}]\}$ and $\mathcal{A}$ be an arbitrary adversary attacking Ψ in the* IND-TKR-KEY *sense. Then*

$$\mathbf{Adv}^{\text{ind-tkr-key}}_{\Psi}(\mathcal{A}) = 0$$

The proofs use standard reductionist arguments, for lack of space we do not include them here.

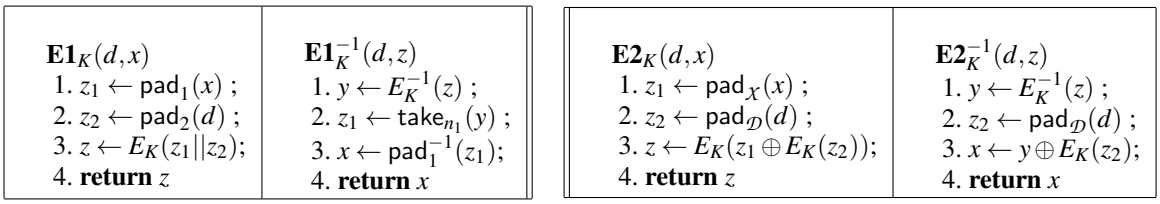

Figure 8: The two instantiations of $\mathbf{E}_K$.

	IND-TKR	IND-TKR-CV	IND-TKR-KEY
TKR1	√	√	
TKR2[f_k]	√		√
TKR2[TR]	√		√
TKR2a[f_k,**E**]	√	√	
TKR2a[TR,**E**]	√	√	√

Table 1: Summary of Security

8 Discussions

SECURITY. The security properties of the various schemes as stated in the previous security theorems are summarized in Table 1. The security theorems in all cases are to be interpreted carefully. We note down some relevant issues below.

In TKR1 the security is gained from the security of the format preserving encryption. The scheme FP used in TKR1 is required to be a tweakable pseudo-random permutation with the message/cipher space $\mathcal{T}$ and the tweak space $\mathcal{D}$. It is important to note that various instantiations of FP can give different security guarantees. Most of the known FPE schemes can only ensure security (in provable terms) when the number of queries made by an adversary is highly restricted. For example, the security claim of the scheme based on Feistel networks discussed in (Black and Rogaway, 2002) becomes vacuous when the number of queries exceeds $2^{\#\mathcal{T}/4}$, whereas the scheme in (Morris et al., 2009) can tolerate up to $2^{\#\mathcal{T}-\varepsilon}$ queries where ε is inversely related to the number of rounds in the construction. Some recent constructions in (Hoang et al., 2012; Ristenpart and Yilek, 2013) achieve much better bounds, specially in (Ristenpart and Yilek, 2013) almost $\#\mathcal{T}$ queries can be tolerated for the bound to be meaningful. As $\#\mathcal{T}$ can be much smaller than the typical domain of a block cipher (2^n, for $n = 128$), thus the exact security guarantees are important in this context. Note, that for a typical scenario if we consider credit card numbers of sixteen decimal digits then $\#\mathcal{T} \approx 2^{53}$.

In the construction of TKR2 and TKR2a the security bounds are better. If RN[f_k] is instantiated as in Figure 7, and in turn f_k is constructed using a block cipher, then using Proposition 1 and Theorem 3, for any IND-TKR adversary $\mathcal{A}$ who asks at most q queries, we have

$$\mathbf{Adv}^{\text{ind-tkr}}_{\Psi}(\mathcal{A}) \leq \frac{m^2q^2}{2^n} + \varepsilon_q,$$

where $\Psi \in \{\mathsf{TKR2}, \mathsf{TKR2a}\}$ and ε_q is the maximum prf advantage of any adversary (who asks at most q queries) in attacking the block cipher E. Note that, n is the block length of the block cipher used to construct f_k. And m depends on $\#\mathcal{T}$, as per the description of the block cipher based construction in Section 7.2, $m = L/n$, and we discussed that it would be enough if we take $L = 3\mu\lambda$, where μ is the length of each token where the tokens are treated as strings in AL and $\lambda = \lceil \lg \#\mathsf{AL} \rceil$. Thus, the security bound is less sensitive on $\#\mathcal{T}$. The bound only becomes vacuous when mq is of the order of $2^{n/2}$. A similar bound holds for $\mathbf{Adv}^{\text{ind-tkr-cv}}_{\mathsf{TKR2a}[f_k]}(\mathcal{A})$, when a block cipher based construction for f_k is used.

The IND-TKR-KEY definition is meant to model the property of independence of the tokens with the keys, and this represents a quite strong notion of security. The constructions TKR2[f_k] and TKR2a[f_k] do not achieve this security. But TKR2[TR] and TKR2a[TR] achieve security in the IND-TKR-KEY sense as here we are assuming an instantiation by a "true" random number generator.

EFFICIENCY. The efficiency of TKR1 depends on the efficiency of the FP scheme. As discussed there are various ways to instantiate FP with varying amount of security and efficiency. Also, most schemes with provable guarantees are far inefficient than standard block ciphers.

The efficiency of TKR2 and TKR2a would be dominated by the search procedure. Asymptotically, if $\#\mathcal{T} = N$, then tokenization and detokenization would take $O(\lg N)$ time. But the hidden constant would depend on how efficiently the search has been implemented and how powerful the machine is (mainly in terms of memory).

We performed some preliminary experiments to determine the efficiency of TKR2 instantiated using RN[f_k] (described in Figure 7), where f_k was instantiated with block cipher based construction described in Section 7.2. We implemented the card-vault in a PostgresSQL data-base, and implemented the AES using the intel AES-NI instructions. We considered

a card-vault which already contains 1000000 tokens, and then generated 1000000 more tokens using the algorithm in Figure 5. The average time taken for generating a token was 1.106 milli-seconds [3]. This time includes all the database searches that are to be performed and the time for inserting the relevant data in the card-vault.

9 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 TKR2a. The constructions TKR2 and TKR2a are particularly interesting, as they demonstrate that tokenization can be achieved without the use of format preserving encryption. We analyzed all the constructions in light of our security definitions and also provided some preliminary experimental results. We plan to perform a more rigourous efficiency comparison of the proposed schemes with the existing FPE schemes in near future.

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.

Bellare, M., Rogaway, P., and Spies, T. (2010). The FFX mode of operation for format-preserving encryption. NIST submission. http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/ffx/ffx-spec.pdf.

Berbain, C. and Gilbert, H. (2007). On the security of IV dependent stream ciphers. In Biryukov, A., editor, *FSE*, volume 4593 of *Lecture Notes in Computer Science*, pages 254–273. Springer.

Black, J. and Rogaway, P. (2002). Ciphers with arbitrary finite domains. In Preneel, B., editor, *CT-RSA*, volume 2271 of *Lecture Notes in Computer Science*, pages 114–130. 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.

[3]The experiments were performed on an Intel i5-2400 @3.1GHz with 4GB RAM

CardHub (2012). Number of credit cards and credit card holders. Available at http://www.cardhub.com/edu/number-of-credit-cards/.

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.

ISO/IEC 7812-1 (2006). Identification cards-identification of issuers-part 1: Numbering system.

Liskov, M., Rivest, R. L., and Wagner, D. (2002). Tweakable block ciphers. In Yung, M., editor, *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 31–46. 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.

Ristenpart, T. and Yilek, S. (2013). The mix-and-cut shuffle: Small-domain encryption secure against n queries. In Canetti, R. and Garay, J. A., editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 392–409. Springer.

Robshaw, M. J. B. and Billet, O., editors (2008). *New Stream Cipher Designs - The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*. Springer.

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

Securosis White Paper (2011b). Tokenization vs. encryption: Options for compliance. Available at https://securosis.com/research/publication/tokenization-vs.-encryption-options-for-compliance.

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.