CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Studies on Some Weak Notions of Security

A dissertation submitted by

Sandra Díaz Santiago

For the degree of

Doctor of Computer Science

Advisor
Debrup Chakraborty Ph.D.

México, D. F. December, 2014

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL
INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO
DEPARTAMENTO DE COMPUTACIÓN

Estudios sobre algunas Nociones Débiles de Seguridad

Tesis que presenta

Sandra Díaz Santiago

para obtener el grado de

Doctor en Ciencias de la Computación

Director de Tesis:
Dr. Debrup Chakraborty

México, D. F. Diciembre, 2014

Abstract

In our modern world, security of the information has become a growing concern in our society. Cryptography is one of the tools which has been extensively used to achieve various security goals. In general, cryptographic tools are developed to achieve the highest levels of security, and there are applications where if strong cryptography is used in the usual way then it seriously compromises the efficiency and/or usability of the system. In this thesis we explore two such practical problems where this is the case. We explore the right security requirements for these applications and construct some non-standard primitives which are weaker than the traditional schemes but still provides adequate levels of security required for the application. The two problems that we study in this thesis are: (1) How to protect communications against profilers and (2) How to protect credit card numbers.

A profiling adversary is a computer program whose aim is to classify messages sent and/or received by a user and classify them into some pre-defined classes. This classification can reveal important information about the user, for example the user's interests and preferences. This information, though seems harmless, can severely compromise the privacy of an individual. We argue that applying strong encryption to solve this problem is not necessary to obtain security from profilers. We fix an exact security definition for an encryption scheme to be secure against profilers. Further, we propose several concrete schemes and prove that they achieve security in accordance to our definition.

The second problem that we study in this thesis, is related with the financial world. With the popularization of the paradigm of e-commerce it is now possible to buy products and services using a credit card over the internet. Credit card numbers are typically stored in the merchant environments for a variety of purposes. Credit card numbers are very sensitive information, and a merchant who stores or uses credit card data is required to take special security measures. One popular solution which provides security to credit card information is called *tokenization*. In this paradigm instead of storing the credit card number, it is substituted by another value called *token*, which apparently must not reveal any information about the credit card number. To our knowledge there has been no formal study of this useful paradigm from the cryptographic viewpoint. We aim to fill this gap in this thesis. Our principal contributions in this problem are three security definitions for tokenization systems and also several cryptographic constructions of tokenizers, which we analyze in light of our security definitions. Finally, we provide some initial experimental data on the performance of our proposed tokenization procedures.

Resumen

En la actualidad, la seguridad de la información se ha convertido en un área cada vez más importante para nuestra sociedad. La Criptografía es una de las herramientas que ha sido ampliamente usada para satisfacer distintos servicios de seguridad. En general, las herramientas criptográficas se desarrollan para tener el más alto nivel de seguridad y existen aplicaciones donde ésta característica puede comprometer seriamente la eficiencia y/o usabilidad del sistema. En esta tesis, analizamos dos problemas prácticos, que caen en este caso. Hemos estudiado los requerimientos de seguridad apropiados para estos problemas y también hemos construido algunas primitivas criptográficas no estandarizadas, con una seguridad menor que la de los esquemas tradicionales, pero que aún satisfacen un nivel adecuado de seguridad para la aplicación. Los dos problemas que estudiamos en esta tesis son: (1) cómo proteger las comunicaciones contra *adversarios clasificadores* y (2) cómo proteger el número de una tarjeta de crédito.

Un adversario clasificador es un programa de cómputo, cuyo objetivo es separar los mensajes enviados y/o recibidos por un usuario, en clases predefinidas. Esta clasificación puede revelar información importante de un usuario, por ejemplo sus intereses o preferencias. En este contexto, es evidente que un adversario clasificador puede comprometer severamente la privacidad de una persona. Aquí argumentamos que aplicar un algoritmo de cifrado, con un alto nivel de seguridad, es inadecuado para garantizar la seguridad contra este tipo de adversarios. Por tal motivo, damos una definición de seguridad conveniente para que un esquema de cifrado sea seguro contra adversarios clasificadores. Asimismo proponemos varios esquemas y mostramos que satisfacen la definición de seguridad dada.

El segundo problema que estudiamos en esta tesis, esta relacionado con el mundo financiero. Debido a que el comercio electrónico se ha vuelto cada vez más popular, ahora es posible comprar bienes y servicios en internet, usando una tarjeta de crédito. Cuando lo hacemos, los datos de la tarjeta son almacenados en la base de datos del comercio donde adquirimos estos bienes y servicios, con diversos propósitos. En este escenario los datos de la tarjeta de crédito constituyen información vulnerable y los vendedores están obligados a protegerlos. Una solución para hacerlo, que se ha vuelto popular, es un nuevo paradigma conocido en inglés como *tokenization*. Este paradigma propone que en lugar de almacenar el número de tarjeta de crédito, éste sea sustituido por un *token*, el cual es un valor alfanumérico, que aparentemente no proporciona información alguna sobre el número de tarjeta de crédito. Hasta donde sabemos aún no se ha llevado a cabo un análisis formal sobre la seguridad que ofrece este paradigma. Uno de los objetivos de esta tesis es cubrir esta brecha. Nuestras principales contribuciones para resolver este problema son: tres nociones de seguridad para este nuevo paradigma y varias construcciones criptográficas para generar tokens, respaldadas con un análisis de seguridad formal. Finalmente, se muestran los resultados de algunos experimentos preliminares que se llevaron a cabo, para medir el desempeño de los procedimientos propuestos.

Contents

Chapter 1

Introduction

Nowadays communication technologies have become an indispensable part of our lives. Today we are able to transmit a huge amount of information around the world in seconds. And with increased connectivity, the data that we own is scattered across several servers which are physically located all over the world. In many cases this information that we own and transmit is very sensitive and can become a target for different kinds of malicious activities such as unauthorized access, use, modification, destruction, among others. Thus, information security has become a major concern of today, not only for organizations but also for common people. In addition, everyday new security threats appear, this increases our paranoia and we look for solutions which offer the highest level of security.

Cryptography offers several tools which can be used for providing important security services such as privacy, authentication, non-repudiation, integrity, etc. Modern cryptographic primitives sometimes come with provable guarantees of security. For such a guarantee, given an application one needs to formally define the threat model and thereof come up with a formal security definition for a cryptographic primitive or application. Then the goal becomes to construct cryptographic schemes conform to the security definitions. The security of the scheme is argued by means of a "security proof". The proofs of security are based on well studied assumptions which are generally based on hardness of some computational problems or the existence of certain mathematical objects like one-way functions, pseudo-random permutations, etc.

Cryptographic security definitions for a given application generally assume a very powerful adversary with a weak goal, thus, these definitions aim to provide the maximum possible security. Given that we live in a paranoid world, such strong definitions are popular and socially acceptable. The maximum possible security may give us a sense of peace but in many day to day applications such levels of security may be an over-kill or may lead to numerous problems related to usability and availability of a system. This situation is well characterized by a quote of Andrew Odlyzko:

> *Even if we could build a secure system, it would be difficult, if not impossible, to use. For example, a system where secretaries cannot commit "forgery", sign for their bosses, will not work in the real world.*[1]

Though this statement is figurative, it does reveal a reality. Hence, in most practical applications it is important to achieve the right security/usability tradeoff.

In this thesis we aim to study some security applications where some "weak" notions of security are applicable. Although there are several scenarios, which require a weak notion of security, in this thesis we study two problems: protecting communications against profilers and encryption of credit card numbers. As we will see later, if we apply standard strong cryptography in the usual way to solve these problems, then we must deal with other issues. The main goal of this research is to provide suitable security notions for these two scenarios, and to analyze these applications in light of these notions.

In the next section we informally discuss about the two problems that we studied in this work. In the last section we describe the contents of this thesis.

1.1 Problems Addressed

In this section we give a brief and informal account of the two basic problems that we handle in the thesis.

Security against Profiling Adversaries

Every time we use the internet to write and send an email or post something in a social network or simply browse web-sites, we are revealing information about our preferences and interests. This data can be extremely valuable to some entities such as internet publishers, internet providers and government agencies among others. These entities may use this kind of data for different purposes, such as, directed spamming,

[1]quoted from RJ Lipton's blog at http://rjlipton.wordpress.com/2010/05/10/breaking-all-the-security-rules/

displaying advertisement specific to user preferences or simply selling this data to advertisement companies. To obtain this information, there are computer programs that quietly analyze an enormous amount of information about internet users and obtain their profiles.

To formally define the problem, we assume that given a message space, an adversary aims to map each message in the message space into certain classes of his interest. Using this classification of messages the adversary can try to conclude which user is associated with which class and this is expected to reveal information regarding the profile of a given user. Thus, in the scenario of our interest we consider an adversary that classifies messages into pre-defined classes. Such an adversary would be further called as a *profiler*, and the goal is to design encryption schemes which are secure against such profilers.

Using standard encryption which is secure in terms of indistinguishability against chosen plaintext adversaries (IND-CPA secure)[2] one can achieve the above mentioned goal. But it is clear that here we are interested in an adversary whose goal is very different compared to an IND-CPA adversary. Hence, it may be possible to relax the security of the encryption scheme used for data where the threat comes only from profiling adversaries. Hence some of the interesting questions that can be asked in this context are the following:

1. Can a weaker security definition be formulated for an encryption scheme which is meant to be secure only against profiling adversaries?

2. Can one design an encryption scheme satisfying the above notion which would be superior to existing strong encryption schemes in terms of efficiency and/or usability?

3. Can a full protocol be developed in such a way that it relieves the user from the requirement of key exchange or a public key infrastructure?

Fortunately, we were able to find positive answers to all these questions, and we provide details of our solutions in Chapters 3 and 4.

[2]Informally an encryption scheme is considered to be IND-CPA secure if it is difficult for any computationally bounded adversary to distinguish between the ciphertexts produced from two different messages of its choice. We discuss this notion formally in Chapter 2.

Small Domain Encryption and Tokenization

The second problem that we examine comes from the financial world. Today it is well known that credit card numbers are sensitive information and the organizations that process card payments must adopt security measures to protect this information. Although the most obvious solution to this problem is encrypting this data using standard encryption algorithms, there are some problems involved in doing so. Usually a credit card number is 16 decimal digits long, if this is treated as a binary string it is about 53 bits long. This is much less than the block size of a typical block cipher (usually 128 bits). Thus, encrypting in this way not only increases the length of the original data, but in fact transforms the data into a binary string which is not always possible to encode as a number with a fixed number of decimal digits. Although there exist some block ciphers with small block lengths, say 32 bits, 64 bits, etc. But, it is evident that none of them fulfill the specific purpose here.

This problem has given rise to the interesting area of *format preserving encryption (FPE)*. Where the goal is to design encryption algorithms where the domain has an arbitrary format, as in the example above the domain consists of all 16 digit decimal numbers, and the ciphertext is also required to follow the same format. Additionally, the domain can be arbitrarily small. Designing efficient format preserving encryption schemes from available cryptographic primitives (say a block cipher) is surprisingly difficult. Though there are some such schemes reported in the literature [8, 10, 14, 36, 53, 75], which use some very interesting ideas, but none of them can be considered to be efficient enough for practical purposes.

Another problem that arises in designing encryption schemes in small domain is regarding the provably secure bounds that can be proved for them. In general, for most schemes, if the domain contains strings of size n, then the upper bound on the advantage of an adversary[3] is of the order of $q^2/2^n$, assuming that the adversary has seen q ciphertexts produced by the scheme. Note that when n is of the order of 128, then this bound is meaningful for any reasonable value of q, but when n becomes very small this bound becomes vacuous. Though, recently there have been some schemes which guarantee an adversarial upper bound of $q/2^n$, but still in applications where multiple ciphertexts are produced and n is small, this security bound is not sufficient. Thus the field of format preserving encryption in the small domain is largely open, and some of the important problems related to this topic are:

1. Designing FPE schemes for small domains with increased security guarantees.

[3]The advantage of an adversary can be informally seen as its success probability in breaking a scheme. For details see Chapter 2 in Section 2.4.1.

2. Designing FPE schemes which are practically efficient.

3. Study the applications where FPE schemes are required and provide formal security definitions and provably secure solutions for those applications.

In our work we have mostly addressed the last point by taking the application of encrypting credit card numbers. In the current days, organizations have adopted a new solution which is called “tokenization”. Tokenization is a philosophy where instead of storing credit card numbers, a token is stored. To our knowledge a cryptographic study of tokenization systems has not appeared in the literature. We do an extensive and formal study of tokenization in Chapters 5 and 6.

1.2 Organization of this Document

This thesis is divided into seven chapters which intend to explain the results that we obtained throughout the research.

In Chapter 2 we discuss some preliminaries required to understand this work. We start by describing some basic concepts of cryptography and the general notations used throughout this document. Given that all our protocols and security notions are in the symmetric cryptography setting, hence we provide definitions of some basic cryptographic objects used in this setting. We also discuss some tools related to provable security, which are extensively used to show the security of our protocols.

The first problem, that we study in this thesis, is the problem of securing communications from profiling adversaries. As discussed in Section 1.1, a profiling adversary is a specific type of adversary who aims to classify messages sent or received by a user. To our knowledge, the only previous solution to this problem, was proposed by Golle and Farahat [35]. The work in [35] does not delve into the formal aspects of the problem, they do not even formally define the security requirements for an encryption scheme to be secure against profilers. Thus, our first step in Chapter 3, is to introduce a security notion for encryption schemes meant to provide security against profiling adversaries. We call this new security notion as PROF-EAV security. Next we develop a protocol, which we prove to be secure against profilers in accordance to the proposed security definition. There are several curious properties of this protocol. It neither requires a key exchange nor a public key infrastructure. To achieve this, we encapsulate some data required to generate the key in a CAPTCHA. A CAPTCHA is a mechanism to differentiate between a human and a computer program. Nowadays this tool is amply used in several applications, to prevent automated programs from using computing services or collecting sensitive information. To our knowledge, CAPTCHAs have not been previously used for encryption. The most common type of CAPTCHA is an image of a distorted word, which is supposed to be easy to read for a human user, but very hard to read for a computer program. However, this mechanism can fail, sometimes it happens that a human user cannot solve a CAPTCHA. In our protocol if this occurs then the user will be unable to recover the key. To remedy this situation we provide a variant to our original protocol, which solves this problem to a large extent. Finally, in Chapter 3 we also provide a formal analysis of the scheme proposed by Golle and Farahat in [35], according to our security notion. Most parts of this chapter have been already published in [23, 24].

In Chapter 4 we continue with the study of encryption schemes secure against profilers. In this Chapter we focus on developing a new encryption scheme, which is secure against profilers. Our scheme is inspired by a previous “encryption” scheme introduced by Rivest [65], known as *Chaffing and Winnowing* (CW). These terms allude to a procedure used in agriculture, denominated “winnowing”, to separate the grain from the chaff. The purpose of this encryption scheme is to achieve privacy without really encrypting the message. The basic idea is to hide the original message with garbage (or chaff), such that a valid user would be able to filter (winnow) out the real message from the chaff, but to an invalid user the message remains hidden. We use the idea of CW to develop two encryption schemes whose main purpose is to deceive a profiler. Our scheme works for messages in some natural language, and given a message our scheme adds fake words in random positions of the message. The way we add these fake words is designed to achieve the goal that a document classifier would be unable to know the real class of the message given the chaffed message. Our scheme uses some novel techniques from document classification. We also provide a preliminary theoretical analysis and some experimental results for the proposed schemes.

In Chapters 5 and 6 we deal with the problem of encrypting credit card numbers. As stated earlier, there are several practical issues which prevent encrypting credit card numbers by using off the shelf encryption algorithms. A well accepted solution for achieving security of credit card numbers is through the paradigm of tokenization. In Chapter 5 we begin with a formal cryptographic study of tokenization systems, which to our knowledge has not been done before. Firstly, we give a formal syntax, which encompasses all components of a typical tokenization system. Further, we propose three different security notions for tokenizers. These three notions are meant for three different practical threat scenarios. Finally, we propose three schemes to generate tokens and analyze their security in light of our definitions. The proposed schemes are all designed using generic, off the shelf cryptographic objects.

In Chapter 6 we focus on practical and efficient instantiations of the tokenizers proposed in Chapter 5. We also provide some initial experimental performance data of the proposed tokenizers in a practical tokenization environment. The contents of Chapters 5 and 6 have been previously published in [25, 26].

Finally in Chapter 7 we summarize our contributions and discuss some directions for improvements to our work. Furthermore, we discuss some other application areas where weak notions of security are applicable but were not addressed in this thesis.

Chapter 2

Preliminaries

The main goal of this research is to develop adequate security notions and cryptographic protocols for different scenarios, which we already described in the Introduction. Our security notions and protocols have been developed in the setting of symmetric cryptography. Thus in this Chapter we will explain the main concepts and techniques used in this setting to reach our goals. Here we will talk about cryptographic objects such as block ciphers, pseudorandom functions and pseudorandom permutations. We will also discuss about the *provable security* paradigm, whose techniques have been extensively used in this research.

2.1 General Notations

The set of all binary strings is denoted by $\{0,1\}^*$ and the set of all n bit strings would be denoted by $\{0,1\}^n$. For a binary string x, $|x|$ will denote the length of x and for a finite set A, $\#A$ will denote its cardinality. If X, Y are strings, $X||Y$ will denote the concatenation of X and Y. For a non-negative integer $i < 2^n$, $\mathsf{bin}_n(i)$ would denote the n-bit binary representation of i. For a finite set $\mathcal{S}$, $x \xleftarrow{\$} \mathcal{S}$ will denote x to be an element selected uniformly at random from $\mathcal{S}$.

In what follows, by an adversary we shall mean a probabilistic algorithm which outputs an integer or a bit. $\mathcal{A}(x, y) \Rightarrow b$, will denote the fact that an adversary $\mathcal{A}$ given inputs x, y outputs a bit b. By $\mathcal{A}^{\mathcal{O}} \Rightarrow b$ we will mean an adversary $\mathcal{A}$, has an access to an oracle $\mathcal{O}$ and after interacting with its oracle $\mathcal{O}$ it outputs a bit b. In general an adversary would have other sorts of interactions, maybe with other adversaries and/or algorithms before it outputs, these interactions would be clear from the context.

2.2 Block Ciphers

Block ciphers are fundamental objects in modern cryptography. Using them it is possible to construct cryptographic protocols which provide confidentiality, integrity and authentication. We define a block cipher as follows. Let $\mathcal{M}$ be the *message space*, let $\mathcal{C}$ be the *cipher space* and let $\mathcal{K}$ be the *key space*. We define a block cipher as a function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, where $\mathcal{M} = \mathcal{C} = \{0,1\}^n$ and $\mathcal{K} = \{0,1\}^k$, i.e., E takes as input a string of length k and a string of length n and it returns a string of length n. In this context n is called the *block length* and k the key length. For $K \in \mathcal{K}$ and $M \in \mathcal{M}$, we will use $E_K(M)$ instead of $E(K, M)$. E_K must be a permutation for all $K \in \mathcal{K}$, in other words E_K must be a bijection from $\{0,1\}^n$ to $\{0,1\}^n$, which means that for every $C \in \{0,1\}^n$ there is exactly one $M \in \{0,1\}^n$ such that $E_K(M) = C$. Naturally, E_K has an inverse, denoted by E_K^{-1}. It is clear that $E_K^{-1}(E_K(M)) = M$ and $E_K(E_K^{-1}(C)) = C$ for all $M \in \mathcal{M}$ and $C \in \mathcal{C}$. Both E_K and E_K^{-1} must be public and easy to compute.

2.3 Pseudorandom Functions and Pseudorandom Permutations

In this section we will introduce two cryptographic objects, which will be helpful to establish an abstract notion of security for block ciphers. These objects, namely pseudorandom permutations and pseudorandom functions are fundamental primitives of symmetric key cryptography. The discussions here closely follows [49].

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

Given a function family F where the sets keys, domain and range are not specified we shall often write $\mathsf{Keys}(F)$, $\mathsf{Dom}(F)$, $\mathsf{Range}(F)$ to specify them.

Let $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be a function family where $\mathcal{D} = \mathcal{R}$, and for every $K \in \mathcal{K}$, let $F_K : \mathcal{D} \to \mathcal{D}$ be a bijection, then we say that F is a permutation family. So, if F is a permutation family, then for every $F_K(\cdot)$, we have a $F_K^{-1}(\cdot)$, such that for all $K \in \mathcal{K}$ and all $X \in \mathcal{D}$, $F_K^{-1}(F_K(X)) = X$. Note that, because of the definition of a block cipher in Section 2.2, a block-cipher is a permutation family.

We would be interested in probability distributions over a function family F, in particular we would often talk of sampling an instance at random from the family. By sampling an instance f uniformly at random from F we would mean $K \xleftarrow{\$} \mathcal{K}$ and $f = F_K()$, we will denote this by $f \xleftarrow{\$} F$.

Random Function. Let $\mathsf{Func}(\mathcal{D}, \mathcal{R})$ be the set of all functions mapping $\mathcal{D}$ to $\mathcal{R}$, if $\mathcal{D} = \{0,1\}^m$ and $\mathcal{R} = \{0,1\}^n$ then $\mathsf{Func}(m, n)$ is the set of all functions that map from m bits to n bits. Note, there are exactly 2^{n2^m} of these functions. i.e., $\#\mathsf{Func}(m, n) = 2^{n2^m}$. If $\mathcal{D}$ and $\mathcal{R}$ are specified, then by a random function with domain $\mathcal{D}$ and range $\mathcal{R}$ we mean a function sampled uniformly at random from $\mathsf{Func}(\mathcal{D}, \mathcal{R})$. Hence by a random function, we are not talking of the "randomness" of a specific function but we are talking of a function sampled from a probability distribution (specifically, the uniform distribution) over the set of all possible functions with a specified domain and range.

Random Permutations. Let $\mathsf{Perm}(\mathcal{D})$ be the set of all bijective maps from $\mathcal{D}$ to $\mathcal{D}$. If $\mathcal{D} = \{0,1\}^n$, then we denote $\mathsf{Perm}(\mathcal{D})$ by $\mathsf{Perm}(n)$, the set of all permutations from $\{0,1\}^n$ to $\{0,1\}^n$. Similar to a random function, we define a random permutation with domain $\mathcal{D}$ to be a function chosen uniformly at random from $\mathsf{Perm}(\mathcal{D})$.

2.3.1 Pseudorandom Functions

Informally a pseudorandom function (PRF) is a family of functions whose behavior is computationally indistinguishable from a random function. Consider the function family $F = \{F_K\}_{K \in \mathcal{K}}$ in the same way that we defined it in the previous section, and let $f \xleftarrow{\$} F$ and let $\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{D}, \mathcal{R})$. If F is a PRF family then there should be no efficient procedure to distinguish between f and ρ. To formalize this goal of distinguishing between a random instance of F and a random instance of $\mathsf{Func}(\mathcal{D}, \mathcal{R})$, we introduce an entity which we call as a PRF adversary. A PRF adversary is considered to be a probabilistic algorithm whose goal is to distinguish between f and ρ, and if it can successfully do so then we say that the adversary has broken the PRF property of F. The adversary is not provided with the description of the functions but it has an *oracle access* to a function g which is either f or ρ and it needs to decide whether $g = f$. By an oracle access we mean that for any $x \in \mathcal{D}$ of its choice, the adversary can obtain the value $g(x)$ by querying the oracle of g. The adversary has the ability to query its oracle g adaptively, i.e., it may be that first it wishes to query its oracle on x_1 and thus obtain $g(x_1)$, seeing $g(x_1)$ it decides its next query x_2 and so on. The adversary can query its oracle as long as it wants and finally it outputs a bit, say it outputs a 1 if it thinks that its oracle is f (a real instance from the family F) and a zero if it thinks its oracle is ρ (a random function). An adversary $\mathcal{A}$ interacting with an oracle $\mathcal{O}$ and outputting a 1 will be denoted by $A^{\mathcal{O}} \Rightarrow 1$.

The PRF advantage of an adversary $\mathcal{A}$ in distinguishing F from a random function is defined as

$$\mathbf{Adv}_F^{\text{prf}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{F(K,.)} \Rightarrow 1\right] - \Pr\left[\rho \xleftarrow{\$} \text{Func}(\mathcal{D}, \mathcal{R}) : \mathcal{A}^{\rho(.)} \Rightarrow 1\right]\right|. \quad (2.1)$$

Hence the PRF advantage of the adversary $\mathcal{A}$ is computed as a difference between two probabilities, the adversary $\mathcal{A}$ is required to distinguish between two situations, the first situation is where $\mathcal{A}$ is given a uniformly chosen member of the family F (i.e., $\mathcal{A}$ has oracle access to the procedure F_K, where $K \xleftarrow{\$} \mathcal{K}$) and in the other $\mathcal{A}$ is given oracle access to a uniformly chosen element of $\mathsf{Func}(\mathcal{D}, \mathcal{R})$. If the adversary cannot tell apart these two situations then we consider F to be a pseudorandom family. In other words F is considered to be pseudorandom if for all *efficient* adversaries $\mathcal{A}$, $\mathbf{Adv}_F^{\text{prf}}(\mathcal{A})$ is *small*.

In this definition we use *efficient* adversary with *small* advantage. We will never make this more precise, and this is standard with the paradigm of "concrete security" where a precise notion of efficiency and small advantage is never specified. What makes an adversary efficient and its advantage small is left to be interpreted with respect to a specific application where such an object would be used.

2.3.2 Pseudorandom Permutation

Let $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ be a family of functions such that for every $K \in \mathcal{K}$, $E_K : \mathcal{D} \to \mathcal{D}$ is a bijection. Analogous to the definition of PRF advantage, we define the PRP advantage of an adversary in distinguishing a random instance of the family E from a random permutation π as

$$\mathbf{Adv}_E^{\text{prp}}(\mathcal{A}) = \left|\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_K()} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \text{Perm}(\mathcal{D}) : A^{\pi()} \Rightarrow 1\right]\right|.$$

And, E is considered to be a pseudorandom permutation family if for all efficient adversaries $\mathcal{A}$, $\mathbf{Adv}_E^{\text{prp}}(\mathcal{A})$ is small.

As every member of a permutation family has an inverse, hence in case of permutations we can define a stronger notion of pseudorandomness. Here we assume that the adversary is given two oracles one of the permutation and other of its inverse and the adversary can adaptively query both oracles. As before, there are two possible scenarios, in the first scenario the adversary is provided with the oracles $E_K()$ and $E_K^{-1}()$ where $K \xleftarrow{\$} \mathcal{K}$ and in the other scenario the oracles $\pi(), \pi^{-1}()$ are provided where $\pi \xleftarrow{\$} \mathsf{Perm}(\mathcal{D})$. And the goal of the adversary is to distinguish between these two scenarios. We define the advantage of an adversary $\mathcal{A}$ in distinguishing a permutation family

E from a random permutation in the $\pm$prp sense as

$$\mathbf{Adv}_E^{\pm \mathrm{prp}}(\mathcal{A}) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{E_K(),E_K^{-1}()} \Rightarrow 1\right] - \Pr\left[\pi \xleftarrow{\$} \mathrm{Perm}(\mathcal{D}) : \mathcal{A}^{\pi(),\pi^{-1}()} \Rightarrow 1\right],$$

and if for all efficient adversaries $\mathcal{A}$, $\mathbf{Adv}_E^{\pm \mathrm{prp}}(\mathcal{A})$ is small then we say E is a strong pseudorandom permutation (SPRP) family.

Security of Block Ciphers. As defined in Section 2.2, a block cipher is a permutation family $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Of course any such permutation family cannot be considered as a block cipher, as a block cipher should have some security properties associated with it which any permutation family will not have. Defining security of a block cipher is tricky (as this is true for all cryptographic primitives), if we consider that a block cipher $E_K()$ is used to encrypt n bit strings then ideally given $E_K(X)$ one should not be able to obtain any information regarding K or X, this property can be achieved if $E_K(X)$ "looks random" to any computationally bounded adversary. In practice we consider a block cipher to be secure if it behaves like a strong pseudo-random permutation.

Unfortunately for the block ciphers that are in use we are not able to prove that they are really SPRPs. So we assume that a secure block-cipher is a SPRP, the assumption is based on our long term inability to find an efficient algorithm which can distinguish a block cipher from a random permutation. If such an algorithm is discovered then the block cipher would be broken. It is worth mentioning here that one can construct PRFs, PRPs and SPRPs based on other mathematical assumptions, in particular if we assume that one way functions exist then we can construct PRFs, PRPs and SPRPs using one way functions [32, 43], such constructions though theoretically are more appealing but are much inefficient compared to the block ciphers in use.

2.4 Symmetric Encryption

Historically, a primary goal of cryptography is to guarantee that two parties A and B can communicate in such manner, that an unauthorized party would be unable to understand the information transmitted during communication. Although there are two basic ways to achieve this goal, in this document we will focus on one of them, called *symmetric-key encryption*. In this setting, A and B share some secret information in advance called a *key*. Then to protect the message, a sender will use an *encryption* algorithm, which receives as input the key and the message or *plaintext*. This encryption algorithm will be used to "scramble" a *plaintext*, the output of this process, called *ciphertext*, will be sent to the receiver. To recover the plaintext, the receiver must *decrypt* the ciphertext, using the key, previously shared with the sender. It is important to notice that in this setting we use the same key to encrypt and decrypt and also that this key allows to distinguish between authorized parties from those who are not. Now we will formally describe a symmetric encryption scheme.

Definition 2.1. *A symmetric encryption scheme is a triple of algorithms* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, which works as follows*

- *The key generation algorithm* Gen *selects a key* K *uniformly at random from the key space* $\mathcal{K}$*, we will denote this as* $K \overset{\$}{\leftarrow} \mathcal{K}$*. This key* K *is used by both algorithms* Enc *and* Dec*, which means the sender and the receiver share a key.*

- *The encryption algorithm* Enc*, takes as input a plaintext* $M \in \mathcal{M}$ *and a key* K *generated by* Gen *and returns a ciphertext* $C \in \mathcal{C}$*. We usually denote this by* $C \leftarrow \mathsf{Enc}_K(M)$*.*

- *The decryption algorithm* Dec*, takes as input a ciphertext* C *and a key* K *and returns* M*. This operation is denoted by* $M \leftarrow \mathsf{Dec}_K(C)$*. A basic correctness requirement for all symmetric encryption schemes is that for all possible keys* K *and all possible messages* M
$$\mathsf{Dec}_K(\mathsf{Enc}_K(M)) = M.$$

The above definition gives the basic syntax of a symmetric encryption scheme. There can be different kinds of symmetric encryption schemes, which provide different kinds of security services.

The symmetric key encryption schemes are usually built using symmetric primitives like stream ciphers and block ciphers. Stream ciphers can be viewed as pseudorandom generators which produces a stream of random bits using a (small) key as the seed and the plaintext is suitably mixed with this random bit stream to produce the ciphertext. Examples of stream ciphers are RC4, A5, SEAL [68], etc. Examples of block ciphers are DES[55], AES[21], IDEA[47], etc.

Because block ciphers have proved to be more robust[1] than stream ciphers, it is most common to use them to construct symmetric encryption schemes. Hence we will focus on this primitive.

[1]The number of attacks reported on stream ciphers far exceeds the number of attacks reported on any block cipher.

2.4.1 Security of Symmetric Encryption Schemes

A fundamental question about symmetric encryption schemes is how to argue about their security. Intuitively we would like to have encryption schemes which provide the maximum security, i.e, we would like an ideal encryption, which would leak no information at all. In the case of *privacy* we already have a definition called *perfect security*, given by Shannon [73]. Perfect security asks that regardless of the computing power available to the adversary, the ciphertext provides no information about the plaintext beyond the apriori information the adversary had, prior to seeing the ciphertext. Unfortunately, to achieve this level of security, we require that the key length be as long as the length of the plaintext. This is not practical. Thus we must consider a different security notion, where the adversary has a limited computing power. In other words, we will consider a *computational security* approach. In this approach, to give a proper security definition, we need to consider the *power* of the adversary and a description of what means by *breaking* an encryption scheme. The power of the adversary is determined by the type of information he/she could access (for example the ciphertext) and the computational resources he/she has at his disposal. Here it is important to notice that we make no assumptions about the adversary's strategy. Since it would be impossible to consider all possible strategies. An example of *breaking* a scheme, could be that the adversary is able to learn partial information about the plaintext. An easier example of breaking a scheme is related with the notion of *indistinguishability*. This notion considers an adversary who chooses two messages of the same length. Then one of the messages is encrypted and the ciphertext is given to adversary. If the adversary can distinguish which message corresponds to the ciphertext, then the encryption scheme is broken.

Now we provide a well established and heavily used security definition of encryption schemes, which is called the IND-CPA security (read as indistinguishability against chosen plaintext adversaries). To define the IND-CPA security for a symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ we use a security game. A game describes the interactions between an adversary which tries to break the scheme in some sense and a challenger who claims that the scheme is secure. The Exp-IND-CPA game described in Figure 2.1 depicts an interaction between a challenger and an adversary $\mathcal{A}$.

In the game described in Figure 2.1, the challenger first randomly selects a key k from the key space $\mathcal{K}$ and instantiates the encryption algorithm $\mathcal{E}$. Later in step 2, it answers to queries of $\mathcal{A}$. The queries of $\mathcal{A}$ consist of valid messages from the message space, in response to a query x, $\mathcal{A}$ receives $\mathcal{E}_k(x)$. $\mathcal{A}$ can ask as many queries as he/she wants to his challenger. This step models a chosen plaintext adversary, who can see the ciphertexts corresponding to the messages of his choice. When $\mathcal{A}$ is satisfied with his/her

```
**Experiment** Exp-IND-CPA$^{\mathcal{A}}$
1. The challenger selects $k \overset{\$}{\leftarrow} \mathcal{K}$.
2. For each query $x$ of $\mathcal{A}$, the challenger returns $\mathcal{E}_k(x)$.
3. $\mathcal{A}$ selects two distinct messages $m_0, m_1$, such that $|m_0| = |m_1|$,
   and gives them to the challenger.
4. The challenger selects $b \overset{\$}{\leftarrow} \{0,1\}$ and gives $\mathcal{E}_k(m_b)$ to $\mathcal{A}$.
5. $\mathcal{A}$ returns a bit $b'$ to the challenger.
6. if $b = b'$ output 1
7. else output 0
```

Figure 2.1: The IND-CPA game depicting interaction between the challenger and the adversary $\mathcal{A}$

queries, he/she chooses two distinct messages m_0, m_1, such that $|m_0| = |m_1|$ and submits them to the challenger. Then the challenger randomly chooses one of them and gives to $\mathcal{A}$ the corresponding ciphertext. The task of $\mathcal{A}$ is to guess which message was encrypted by the challenger. If $\mathcal{A}$ can correctly guess then the game outputs a 1 else it outputs a zero. Note that in line 2 of the game, $\mathcal{A}$ can make several queries, this is called as the *query phase*. In line 3, $\mathcal{A}$ makes a single query which we call as the *challenge query*. When we say that an IND-CPA adversary $\mathcal{A}$ makes q queries, we mean that it makes $q-1$ queries in the query phase and one query in the challenge phase.

We define the IND-CPA advantage of $\mathcal{A}$ in breaking the scheme Π as

$$\mathbf{Adv}_{\Pi}^{\text{ind-cpa}}(\mathcal{A}) = \left|\Pr[\text{Exp-IND-CPA}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2}\right|. \tag{2.2}$$

As it is obvious, $\mathcal{A}$ can always guess the bit b with probability $\frac{1}{2}$. Thus, for Π to be secure $\mathbf{Adv}_{\Pi}^{\text{ind-cpa}}$ should be "small" for all computationally bounded adversaries $\mathcal{A}$.

The IND-CPA definition is a representative notion of security for a symmetric encryption scheme. Another popular notion of security is called *semantic security*, (informally) a scheme Π is said to be semantically secure if given the ciphertext, no computationally bounded adversary can compute *any* function involving the plaintext. It can be shown that the IND-CPA and semantic security are equivalent [43].

2.5 Proofs by Reduction

In modern cryptography, to design a new cryptographic protocol to solve a particular problem, we basically follow three principles [43], which we describe next.

1. The first step in solving any cryptographic problem is the formulation of a rigorous and precise definition of security.

2. When the security of a cryptographic construction relies on an unproven assumption, this assumption must be precisely stated. Furthermore, the assumption should be minimal as possible.

3. Cryptographic constructions should be accompanied by a rigorous proof of security with respect to a definition formulated according to principle 1.

In Section 2.4.1 we amply discussed the first principle, i.e, how to define the security of an encryption scheme. In this section, we will focus in the last two principles.

Once that we have a proper security definition, we need to *prove* that our cryptographic protocol Π satisfies this security definition. For this purpose, we will follow the techniques established by *provable security*. Provable security was proposed by Shafi Goldwasser and Silvio Micali in 1984 [34]. In this paradigm we construct a new protocol, assuming the existence of secure atomic primitives. These atomic primitives must satisfy certain security property and come from two sources: engineered constructs and mathematical problems. Block ciphers, which we discussed in Section 2.2, belong to the first category. A good example of the second category is the well known RSA function.

After we have a security definition and given a cryptographic protocol Π based on some specific atomic primitive, we are able to *prove* the security of our protocol. The security proof consists of arguments that show that a scheme really has the properties specified in the security definition. In computational security it is almost never possible to show that a scheme satisfies a security definition unconditionally. The security of Π is proved based on the security assumptions on the atomic primitive. The proof technique involves a reduction which shows that if the scheme is insecure then the primitive is also insecure. Thus a security theorem is not an absolute statement and needs to be interpreted carefully[2].

[2]In the recent years there have been some criticisms to the paradigm of provable security and there have been proposals that given the relative nature of a security theorem and its proof they should not be called as security proofs but as reductionist arguments [45, 46].

```
CTR.$\mathcal{E}_K(IV, M)$
1. Break $M$ into $n$ bit blocks $M_1, M_2, \ldots M_m$;
2. $W \leftarrow E_K(IV)$;
3. for $i \leftarrow 1$ to $m$
4.     $C_i \leftarrow E_K(W \oplus \mathsf{bin}_n(i)) \oplus M_i$;
5. end for
6. return $(IV, C_1||C_2|| \ldots ||C_m)$
```

```
CTR.$\mathcal{D}_K(IV, C)$
1. Break $C$ into $n$ bit blocks $C_1, C_2, \ldots C_m$;
2. $W \leftarrow E_K(IV)$
3. for $i \leftarrow 1$ to $m$
4.     $M_i \leftarrow E_K(W \oplus \mathsf{bin}_n(i)) \oplus C_i$;
5. end for
6. return $M_1||M_2|| \ldots ||M_m$
```

Figure 2.2: The counter (CTR) mode.

The reductionist argument used to prove security of a scheme involves computing probabilities in certain probability spaces which involve randomness of the scheme and also the randomness used by the adversary. Such computations become difficult if the arguments are not well structured. A popular way of constructing such reductions is the *game playing technique* or *sequence of games*. We have extensively used the game playing technique to prove security of our proposals. The reader is referred to [74] and [9] for detailed examples and discussions about the technique. In the next section we describe the security of the CTR mode of operation, to illustrate some techniques involved in proving security of a cryptographic scheme.

2.6 Security of CTR

Counter mode (commonly called as CTR) is a symmetric encryption scheme which uses a block cipher as the underlying atomic primitive. A popular version of the CTR mode is described in Figure 2.2. The key generation algorithm just selects an uniform random key K from the key space $\mathcal{K}$, and the encryption/decryption procedures uses a block cipher $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ as shown in Figure 2.2. The message space is assumed to contain all binary strings whose lengths are multiples of n (the block length of the block cipher). The counter mode can be described without this restriction on the message space also, but to maintain notational simplicity we describe the restricted version.

As can be seen from the description of Figure 2.2, the encryption algorithm takes in two quantities, an initialization vector IV, and a message M. The syntax of a symmetric encryption scheme in Definition 2.1, describes that the encryption algorithm takes as input only a plaintext and the key, but CTR.Encrypt, takes in an additional initialization vector (IV). There are also other encryption schemes where this additional input is necessary. For security, it is required that for each call the IV should be unique. When an initialization vector is used in such a way, it is called a nonce [67].

We want to prove that CTR mode is IND-CPA secure. Recall the IND-CPA security game in Figure 2.1. Firstly, to define security of CTR, we need to do some subtle changes in the game in Figure 2.1. For CTR, we will assume that the adversarial queries are of the form (IV, m), i.e., the adversary chooses an IV message pair and seeks the encryption of the chosen message using the chosen IV to his challenger. It is required that an IV is never repeated in the adversarial query. Finally, in the challenge phase, the adversary provides the pair $(IV^*, m_0), (IV^*, m_1)$ to his challenger. It is required that IV^* be different from any IV that the adversary has chosen before. The challenger provides the adversary with the encryption of either m_0 or m_1 using the IV^*, and the task of the adversary is to guess which message was encrypted. The above description is just to incorporate an IV in the IND-CPA game, this little syntactical change, does not change in any way the spirit or purpose of the original IND-CPA definition.

As CTR is built over the atomic primitive $E_K()$, to prove the security of CTR, we will assume that the block cipher $E_K()$ is secure. More specifically, we will assume that $E_K()$ is a pseudorandom function[3]. Thus, to show that CTR is secure, we will basically prove that if CTR is not IND-CPA secure then there exists an adversary who can break the PRF property of the underlying $E_K()$, thus violating our assumption on $E_K()$. We state this formally in the next Theorem.

Theorem 2.1. [Security of CTR Mode] *Let $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ be a function family and let $\mathsf{CTR} = (\mathcal{K}, \mathsf{CTR}.\mathcal{E}, \mathsf{CTR}.\mathcal{D})$ be the corresponding CTR symmetric encryption algorithm as described in Figure 2.2. Let $\mathcal{A}$ be an arbitrary adversary attacking CTR in the* IND-CPA *sense, who asks at most q queries, these totaling at most σ n bit blocks. Then there exists an adversary $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{CTR}}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{E}(\mathcal{B}) + \frac{\sigma^2}{2^n}.$$

And if $\mathcal{A}$ runs in time at most t then $\mathcal{B}$ runs in time at most $t' = t + O(q)$ and asks at most $q' = \sigma$ oracle queries.

Before we give the proof, it would be worthwhile to analyze a bit the statement of the Theorem. The Theorem states that the IND-CPA advantage of an arbitrary adversary in

[3]In Section 2.3.2 we stated that the well accepted security notion for a block cipher is the notion of a strong pseudorandom permutation. Note, that when a block cipher is used within the counter mode an inverse call to it is never required. Thus for correctness of the mode it is not necessary that $E_K()$ be invertible, thus it suffices to consider that $E_K()$ is a function. Hence we are able to prove that CTR is secure by a weaker assumption on $E_K()$, i.e., $E_K()$ is a PRF.

```
Adversary $\mathcal{B}^{\mathcal{O}}$
    Whenever $\mathcal{B}$ gets a query $(IV, M)$ from $\mathcal{A}$
    do the following until $\mathcal{A}$ stops querying
        $C \leftarrow \mathcal{E}_{\mathcal{O}}(IV, M)$
    return $(IV, C)$ to $\mathcal{A}$
    After $\mathcal{A}$ submits $(IV^*, M_0), (IV^*, M_1)$
    do the following
        $b \xleftarrow{\$} \{0, 1\}$
        $C_b \leftarrow \mathcal{E}_{\mathcal{O}}(IV^*, M_b)$
    return $(IV^*, C_b)$ to $\mathcal{A}$
    $\mathcal{A}$ returns a bit $b'$
    If $b = b'$ then return $1$ else $0$
```

Figure 2.3: Adversary $\mathcal{B}$ for the proof of Theorem 2.1.

breaking CTR is upper bounded by the sum of $\sigma^2/2^n$ and the PRF advantage of some adversary $\mathcal{B}$ in breaking E_K. If we assume that E is a PRF-family then for any adversary $\mathcal{B}$, $\mathbf{Adv}_E^{\text{prf}}(\mathcal{B})$, is bound to be small. Thus, if σ^2 is negligibly small compared to 2^n, then the Theorem guarantees that any adversary will have a small IND-CPA advantage in breaking CTR. Thus, the Theorem is a relative statement regarding the security of CTR. It guarantees CTR to be IND-CPA secure if the following holds:

- E is a pseudorandom function family.
- σ^2 is negligibly small compared to 2^n. This condition basically says that CTR is not secure, if an unlimited number of queries is asked by the adversary. Also, this bound may be useful in deciding how many plaintexts are to be encrypted under the same key.

Now we proceed to prove Theorem 2.1.

Proof. For convenience, we denote $\mathsf{CTR}.\mathcal{E}$ by $\mathcal{E}$. Let $\mathcal{A}$ be an adversary attacking $\mathcal{E}$ in the IND-CPA sense. Now we will construct a PRF adversary $\mathcal{B}$ which will act as the challenger for $\mathcal{A}$ in the IND-CPA game. We describe $\mathcal{B}$ in Figure 2.3. $\mathcal{B}$ being a PRF adversary has access to an oracle $\mathcal{O}$ which can be either the block cipher $E_K()$, for $K \xleftarrow{\$} \mathcal{K}$, or a function $\rho \xleftarrow{\$} \mathsf{Func}(n, n)$.

In the description of $\mathcal{B}$, $\mathcal{E}_{\mathcal{O}}(\cdot)$ denotes the encryption function of CTR shown in Figure 2.2, where $E_K()$ is replaced by $\mathcal{O}$. Note, $\mathcal{O}$ is the oracle of $\mathcal{B}$. The goal of $\mathcal{B}$ is to guess if $\mathcal{O}$ is $E_K()$ or a random function ρ. $\mathcal{B}$ tries to reach this goal, with the help of $\mathcal{A}$.

Observe that if the oracle $\mathcal{O}$ is the block cipher E_K, then $\mathcal{B}$ provides the exact environment of the IND-CPA game to $\mathcal{A}$, and it is clear that

$$\Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{E_K(\cdot)} \Rightarrow 1\right] = \Pr[\text{Exp-IND-CPA}^{\mathcal{A}} \Rightarrow 1]. \tag{2.3}$$

Now, we would like to analyze the situation where $\mathcal{O}$ is a random function. In particular we want to find a bound on

$$\Pr\left[\rho \xleftarrow{\$} \mathsf{Func}(n,n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1\right].$$

To do this, we use the technique of sequence of games. Consider the game **G0** shown in Figure 2.4. The game **G0** includes a function **Choose**-$\rho()$, which acts as a random function. It returns uniform random strings in $\{0,1\}^n$ when it is invoked, but it returns the same string if invoked twice on the same input. It does this by maintaining a table ρ of outputs that it has already returned. Additionally in the set Dom, it maintains the points on which it has been queried. The function sets the bad flag to true if it is queried twice on the same input. The game **G0** assumes that the i^{th} query of the adversary is a message containing t_i n-bit blocks, and the challenge query contains messages of t blocks.

The game **G0** does the same as $\mathcal{B}$ does where the oracle of $\mathcal{B}$ is replaced by the function **Choose**-$\rho()$. As **Choose**-ρ acts like a random function, hence it is immediate that

$$\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] = \Pr[\mathbf{G0} \Rightarrow 1] \tag{2.4}$$

Now, we do a small change in game **G0**, i.e., we remove the boxed entry in the function **Choose**-ρ, we call this changed game as **G1**. Notice that games **G1** and **G0** are identical until the flag bad is set to true, hence by the fundamental lemma of game playing [9] we have

$$\Pr[\mathbf{G0} \Rightarrow 1] - \Pr[\mathbf{G1} \Rightarrow 1] \leq \Pr[\mathbf{G1} \text{ sets } \mathsf{bad}] \tag{2.5}$$

Also in game **G1**, the function **Choose**-ρ, returns random strings for any input it gets, thus $\mathcal{A}$ when interacts with **G1** gets random strings of size of the queried plaintexts as response to its queries. Note that the goal of $\mathcal{A}$ is to guess correctly which of the messages M_0 or M_1 was encrypted in the challenge phase, and **G1** outputs a 1, if $\mathcal{A}$ makes a correct guess. As the responses that $\mathcal{A}$ gets in game **G1** are independent of the

```
Game $\boxed{\mathbf{G0}}$, $\mathbf{G1}$

function Choose-$\rho(X)$
    $Y \xleftarrow{\$} \{0,1\}^n$;
    if $X \in$ Dom then
        bad $\leftarrow$ true ; $\boxed{Y \leftarrow \rho[X]}$;
    else
        $\rho[X] \leftarrow Y$; Dom $\leftarrow$ Dom $\cup \{X\}$;
    end if
    return $Y$;
Initialization
    bad $\leftarrow$ false;
    Dom $= \emptyset$;
Query Phase
For a query $(iv^{(i)}, m^{(i)})$ of $\mathcal{A}$ do the following
    Parse $m^{(i)}$ as $m_1^{(i)}||m_2^{(i)}||\dots m_{t_i}^{(i)}$;
    (we assume that $m^{(i)}$ contains $t_i$ blocks)
    $w^{(i)} \leftarrow$ Choose-$\rho(iv^{(i)})$;
    for $s \leftarrow 1$ to $t_i$,
        $\mu \leftarrow$ Choose-$\rho(w^{(i)} \oplus \mathsf{bin}_n(s))$;
        $z_s^{(i)} \leftarrow \mu \oplus m_s^{(i)}$;
    endfor
    return $(iv^{(i)}, z_1^{(i)}||z_2^{(i)}||\dots||z_s^{(i)})$ to $\mathcal{A}$;
Challenge Phase
    When $\mathcal{A}$ submits $(IV^*, M_0), (IV^*, M_1)$
    do the following:
        $b \xleftarrow{\$} \{0,1\}$;
        parse $M_b$ as $M_{b1}||M_{b2}||\cdots||M_{bt}$;
        $w^* \leftarrow$ Choose-$\rho(IV^*)$;
        for $s \leftarrow 1$ to $t$,
            $z_s^* \leftarrow M_{bs} \oplus$ Choose-$\rho(w^* \oplus \mathsf{bin}_n(s))$;
        end for
        return $(IV^*, z_1^*||z_2^*||\dots||z_t^*)$ to $\mathcal{A}$
Finalization phase
After $\mathcal{A}$ returns $b'$ do the following:
    if $b = b'$, return 1;
    else return 0;
```

```
Game $\mathbf{G2}$

Initialization
    bad $\leftarrow$ false;
    Dom $= \emptyset$;
    (We assume that Dom is a multiset )

Query Phase
For a query $(iv^{(i)}, m^{(i)})$ of $\mathcal{A}$, do the following
    $z^{(i)} \xleftarrow{\$} \{0,1\}^{|m^{(i)}|}$;
    return $(iv^{(i)}, z^{(i)})$ to $\mathcal{A}$;

Challenge Phase
    When $\mathcal{A}$ submits $(IV^*, M_0), (IV^*, M_1)$
    do the following:
        $b \xleftarrow{\$} \{0,1\}$; $w^* \xleftarrow{\$} \{0,1\}^n$;
        $z^* \xleftarrow{\$} \{0,1\}^{|M_0|}$;
    return $(IV^*, z^*)$ to $\mathcal{A}$;
Adjustment Phase
    for $i \rightarrow 1$ to $q-1$
        Break $m^{(i)}$ into $t_i$ blocks $m_1^{(i)}||m_2^{(i)}||\dots m_t^{(i)}$ ;
        $w^{(i)} \xleftarrow{\$} \{0,1\}^n$;
        Dom $\leftarrow$ Dom $\cup \{iv^{(i)}\}$
        for $s \leftarrow 1$ to $t_i$
            Dom $\leftarrow$ Dom $\cup \{w^{(i)} \oplus s\}$
        endfor
    endfor
    for $s \leftarrow 1$ to $t$
        Dom $\leftarrow$ Dom $\cup \{w^* \oplus s\}$
    endfor
    if there is a collision in Dom then
        bad $\leftarrow$ true

Finalization phase
After $\mathcal{A}$ returns $b'$ do the following:
    if $b = b'$, return 1
    else return 0
```

Figure 2.4: Games **G0, G1, G2** used for the proof of Theorem 2.1.

queries it makes. Thus the only strategy that $\mathcal{A}$ can follow is to make a random guess. Which results in

$$\Pr[\mathbf{G1} \Rightarrow \mathbf{1}] = 1/2. \tag{2.6}$$

Now, we do some small syntactic changes in the game $\mathbf{G1}$ to obtain the game $\mathbf{G2}$, also shown in Figure 2.4. In game $\mathbf{G2}$ the function **Choose-**ρ is no more used. Here random strings are returned immediately as a response to a query of $\mathcal{A}$. Thus, the responses that $\mathcal{A}$ gets in the games $\mathbf{G1}$ and $\mathbf{G2}$ are identically distributed. The game assumes that the adversary makes a total of $q-1$ queries in the query phase, and after the query and challenge phases are over, the game executes an adjustment phase. In the adjustment phase appropriate values are inserted in the multiset Dom, note as Dom is a multiset hence there can be several instances of the same element present here.

From the discussion above it is clear that $\mathbf{G2}$ is just a different way to write $\mathbf{G1}$, thus

$$\Pr[\mathbf{G1} \Rightarrow 1] = \Pr[\mathbf{G2} \Rightarrow 1], \tag{2.7}$$

also

$$\Pr[\mathbf{G1} \text{ sets bad}] = \Pr[\mathbf{G2} \text{ sets bad}]. \tag{2.8}$$

Thus, using equations (2.4), (2.5), (2.6), (2.7) and (2.8) we get

$$\begin{aligned}\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] &= \Pr[\mathbf{G0} \Rightarrow 1] \\ &\leq \Pr[\mathbf{G1} \Rightarrow 1] + \Pr[\mathbf{G1} \text{ sets bad}] \\ &\leq \Pr[\mathbf{G2} \Rightarrow 1] + \Pr[\mathbf{G2} \text{ sets bad}] \\ &\leq \frac{1}{2} + \Pr[\mathbf{G2} \text{ sets bad}] \end{aligned} \tag{2.9}$$

Now we need to bound $\Pr[\mathbf{G2} \text{ sets bad}]$. Let COLLD be the event that there is a collision in the multiset Dom in game $\mathbf{G2}$, then from the description of game $\mathbf{G2}$, we have

$$\Pr[\mathbf{G2} \text{ sets bad}] = \Pr[\mathsf{COLLD}] \tag{2.10}$$

Now we concentrate on finding an upper bound for $\Pr[\mathsf{COLLD}]$. The elements present in Dom are given by

$$\mathsf{Dom} = S_1 \bigcup S_2 \bigcup S_3,$$

where,

$$\begin{aligned} S_1 &= \{iv^{(1)}, iv^{(2)}, \ldots, iv^{(q-1)}, IV^*\} \\ S_2 &= \bigcup_{i=1}^{q-1} \{w^{(i)} \oplus \mathsf{bin}_n(1), w^{(i)} \oplus \mathsf{bin}_n(2), \ldots, w^{(i)} \oplus \mathsf{bin}_n(t_i)\} \\ S_3 &= \{w^* \oplus 1, w^* \oplus 1, \ldots, w^* \oplus t\}. \end{aligned}$$

Note the sets S_1, S_2 and S_3 are also multisets. Now it would be important to note the following points:

1. $\#S_1 = q$, $\#S_2 = \sum_{i=1}^{q-1} t_i$, $\#S_3 = t$. And as the total n bit blocks of queries that the adversary makes is σ, hence $\sigma = \#S_2 + \#S_3$.
2. For $x, y \in S_1$, $\Pr[x = y] = 0$, as the elements in S_1 are the IVs, which are never repeated.
3. For $x, y \in S_2 \cup S_3$, $\Pr[x = y] \leq 1/2^n$. To see this, notice that each $w^{(i)}$ and also w^* is a uniform random element in $\{0,1\}^n$.
4. For $x \in S_1$ and $y \in S_2 \cup S_3$, $\Pr[x = y] \leq 1/2^n$, because of the same reason as the point before.

Using the above information and the union bound, we have the probability of collision in the multiset Dom as

$$\begin{aligned} \Pr[\mathsf{COLLD}] &= \frac{1}{2^n}\binom{\sigma}{2} + \frac{q\sigma}{2^n} \\ &= \frac{\sigma(\sigma-1)}{2^{n+1}} + \frac{q\sigma}{2^n} = \frac{\sigma^2 - \sigma + 2q\sigma}{2^{n+1}} \\ &\leq \frac{2\sigma^2}{2^{n+1}} \leq \frac{\sigma^2}{2^n}. \end{aligned} \tag{2.11}$$

Now, using equations (2.9), (2.10) and (2.11), we have

$$\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] \leq \frac{1}{2} + \frac{\sigma^2}{2^n}. \tag{2.12}$$

Finally, using equations (2.3), (2.12) and the definitions of PRF advantage of $\mathcal{A}$ and IND-CPA advantage of $\mathcal{B}$, we have the theorem. □

2.7 Summary

Up to now we briefly discussed the basic concepts required to understand the rest of this document. Cryptographic objects such as block ciphers, pseudorandom permutations, and pseudorandom functions will appear often from now on. In the next Chapters we will analyze several encryption schemes, which offer a solution to the problems of profiling adversaries and tokenization. As we will see, the main tools to propose these schemes and to analyze their security fall in the provable security paradigm. Here the proofs by reduction, that we discussed previously, play a very important role.

Chapter 3

Profiling Adversaries

In this chapter we examine the problem of protecting communications against profiling adversaries. To understand the importance of this problem, we start by describing some scenarios where classifiers become a menace to privacy and discuss briefly about previous work on the matter. As we will see, applying strong cryptography to solve this problem is not the best solution. Instead we propose a weaker security notion, which considers the specific properties of a profiling adversary. The rest of this chapter is organized as follows. In Section 3.2 we describe basic concepts related to indistinguishability, CAPTCHA and secret sharing. In Section 3.3 we present a formal definition of a profiling adversary and security against such adversaries. In Sections 3.4 and 3.5 we describe our protocols and argue regarding their security in terms of the security notion given in Section 3.3. In Section 3.6, we revisit the scheme in [35], and give some preliminary arguments regarding its security in accordance with our definitions and security notions. Finally in Section 3.7 we discuss about the limitations of our approach and some future directions.

3.1 Motivation

Informally a spam email is an email which is not of interest to the receiver. Everyday almost every one of us finds hundreds of such spam emails waiting in our in-boxes. A spammer (who sends spam emails) generally has a business motive and most spam emails try to advertise a product, a web-page or a service. If the spam emails can be sent in a directed manner, i.e., if a spammer can send a specific advertisement to a user who would be interested in it, then the motive of the spammer would be successful to a large extent. Thus, one of the important objectives of a spammer would be to know the preferences or interests of the users to whom it is sending the un-solicited messages.

In today's connected world we do a lot of communication through emails and it is not un-realistic to assume that a collection of email messages which originate from a specific user U carries information about the preferences and interests of U. Based on this assumption a spammer can collect email information originating from different users and based on these emails try to make a profile of each user (based on their preferences or interests), and later use this profile for directed spamming.

Here we assume that given a message space an adversary aims to map each message in the message space into certain classes of its interest. Using this classification of messages the adversary can try to conclude which user is associated with which class and this is expected to reveal information regarding the profile of a given user. Thus, in the scenario of our interest we consider an adversary that classifies messages into pre-defined classes. Such an adversary would be further called a *profiler*.

There may be other motives for user profiling in addition to directed spamming. Currently there has been a paradigm shift in the way products are advertised in the internet. In one of the popular new paradigm of *online behavioral advertising* (OBA) [77], internet advertising companies display advertisements specific to user preferences. This requires profiling the users. To support this big business of internet advertising, innovative techniques for user profiling have also developed. It is known that some internet service providers perform a procedure called *deep packet inspection* on all traffic to detect malware etc., but this technique has been used to generate user profiles from the information contents of the packets received or sent by an user, and this information is later sold to advertising companies [77]. This currently has led to many policy related debates, and it has been asked whether such practices should be legally allowed [56].

In the context of emails, a solution to the problem of profiling attacks would be encrypting the communications so that the contents of the emails are not available to the profiler. Or to make the communications anonymous so that given a message it would not be possible for a profiler to trace the origin of the message. Here we try to solve the following question: What would be the exact security requirements for an encryption scheme which can protect the communication from profilers? Intuitively a cipher obtained from a secure encryption algorithm should not reveal any information regarding the plaintext which was used to produce the cipher. Hence, a secure encryption algorithm should surely resist attacks by profilers. But, as the goal of a profiler is only to classify the messages, it is possible that an encryption algorithm which provides security in a weaker sense would be enough to resist profilers. We explore in this direction and try to fix the appropriate security definition of an encryption scheme which would provide security against profilers.

Using any encryption scheme involves the complicated machinery of key exchange (for symmetric encryption) or a public key infrastructure (for asymmetric encryption). When the goal is just to protect information against profilers the heavy machinery of key exchange or public key infrastructure may be unnecessary. Keeping in mind security against profilers we propose a new protocol which does not require explicit key exchange. To do this we use the notion of CAPTCHAs, which are programs that can distinguish between humans and machines by automated Turing tests which are easy for humans to pass but difficult for any machine. The use of CAPTCHAs makes our protocol secure from non-human profilers, but the protocol is still vulnerable to human adversaries. In the context that we see the activity of profiling, it would be only profitable if a large number of users can be profiled and this goal seems to be infeasible if human profilers are employed for the task.

To our knowledge the only prior work on the issue of securing email communication from profilers have been reported by Golle and Farahat in [35]. In [35] it was pointed out that an encryption scheme secure against profilers can be much weaker than normal encryption algorithms, and thus using a normal encryption algorithm can be an overkill. The solution in [35] hides the semantic of the plaintext by converting an English text into another English text with the help of a key. In their protocol also they do not need explicit key exchange or a public key infrastructure. The key is derived from the email header by using a hash function with a specific property. The hash function they use is a "slow one-way hash function", which was first proposed in [30]. Such hash functions are difficult to compute, i.e., may take a few seconds to get computed and are hard to invert. This high computational cost for the hash function prevents a profiler to derive the key for a large number of messages. Our method is fundamentally different from [35] in its use of CAPTCHAs. Slow hash functions which were proposed long ago have not seen much use, and its suitability is not well tested. But CAPTCHAs are ubiquitous in today's world and had been used successfully in diverse applications. Also, our work presents a theoretical analysis of the problem, and provides the security definitions which to our knowledge is new to the literature.

Golle and Farahat [35] did not give a security analysis of their proposed scheme. We provide a formal security analysis of their scheme and we point out the exact assumptions required for the hash function used in their protocol. This analysis uses the security definitions proposed in this work.

3.2 Preliminaries

3.2.1 Indistinguishability in the Presence of an Eavesdropper

As we discussed in Section 2.4.1 security of encryption schemes is best defined in terms of indistinguishability. Here we consider indistinguishability in presence of an eavesdropping adversary, which we call as IND-EAV security. As in the case of IND-CPA notion, this security notion, is also defined with the help of an interaction between two entities called an adversary and a challenger. It considers that an adversary chooses a pair of plaintext messages and then ask for the encryption of those messages to the challenger. The challenger provides the adversary with the encryption of one of the messages chosen by the adversary. The adversary is considered to be successful if it can correctly guess which message of its choice was encrypted. More formally, to define the security of an encryption algorithm $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$, we consider the interaction of an adversary $\mathcal{A}$ with a challenger in the experiment below:

Experiment Exp-IND-EAV$^{\mathcal{A}}$

1. The challenger selects K uniformly at random from $\mathcal{K}$.
2. The adversary $\mathcal{A}$ selects two messages $m_0, m_1 \in \mathcal{M}$, such that $|m_0| = |m_1|$.
3. The challenger selects a bit b uniformly at random from $\{0, 1\}$, and returns $c \leftarrow E_K(m_b)$ to $\mathcal{A}$.
4. The adversary $\mathcal{A}$ outputs a bit b'.
5. If $b = b'$ output 1 else output 0.

Definition 3.1. *Let $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$ be an encryption scheme. The* IND-EAV *advantage of an adversary $\mathcal{A}$ in breaking E is defined as*

$$\mathbf{Adv}_E^{\text{ind-eav}}(\mathcal{A}) = \left| \Pr[\text{Exp-IND-EAV}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \right|.$$

Moreover, E is (ϵ, t)- IND-EAV *secure if $\mathbf{Adv}_E^{\text{ind-eav}}(\mathcal{A}) \leq \epsilon$, for all adversaries $\mathcal{A}$ running for time at most t.*

The IND-EAV security as defined above is used only for one time encryption and it is different from the most used security notion for symmetric encryption which is indistinguishability under chosen plaintext attack (IND-CPA). Recall that in an IND-CPA attack the adversary has the option of asking for the encryption of multiple pairs of messages before he chooses the pair of messages for the challenge phase. IND-EAV notion is strictly weaker than the IND-CPA notion of security. All IND-CPA secure encryption schemes are also IND-EAV secure.

A related notion of security is that of semantic security. Informally a symmetric encryption scheme is called semantically secure if an adversary is unable to compute any function on the plaintext given a ciphertext.

Definition 3.2. *Let $E : \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$ be an encryption scheme. E is called (ϵ, t)- SEM-EAV secure, if for all functions f and for all adversaries running for time at most t*

$$\left|\Pr[\mathcal{A}(E_K(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)]\right| \leq \epsilon \tag{3.1}$$

where the running time of $\mathcal{A}'$ is polynomially related to t, and x is chosen uniformly at random from $\mathcal{M}$.

Note, in the above definition, by $\mathcal{A}'(.)$ we mean that the adversary is given no input, i.e., $\mathcal{A}'$ is trying to predict $f(x)$ without seeing $E_K(x)$. And in the second term of Equation (3.1) the maximum is taken over all adversaries $\mathcal{A}'$ which runs for time at most $poly(t)$, for some polynomial $poly()$. Thus, if E is SEM-EAV secure then no adversary can do better in predicting $f(x)$ from $E_K(x)$ than an adversary who does so without seeing $E_K(x)$. It is well known that IND-EAV security implies SEM-EAV security (for example see Claim 3.11 in [43]).

3.2.2 CAPTCHA

A CAPTCHA is a computer program designed to differentiate a human being from a computer. The fundamental ideas for such a program were first proposed in an unpublished paper [54] and then these ideas were formalized in [79], where the name CAPTCHA was first proposed. CAPTCHA stands for *Completely Automated Public Turing test to tell Computers and Humans Apart*. In fact, a CAPTCHA is a test which is easy to pass by a human user but hard to pass by a machine. One of the most common CAPTCHAs are distorted images of short strings. For a human it is generally very easy to recover the original string from the distorted image, but it is difficult for state of the art character recognition algorithms to recover the original string from the distorted image. Other types of CAPTCHAs which depend on problems of speech recognition, object detection, classification, etc. have also been developed.

Recently, CAPTCHAs have been used in many different scenarios for identification of humans, like in chat rooms, online polls, etc. Also they can be used to prevent dictionary attacks on the password based systems [61], and more recently for key establishment [31].

A CAPTCHA is a randomized algorithm G, which given an input string from a set of strings STR produces the CAPTCHA $G(x)$. A CAPTCHA G is called (α, β) secure, if for any human or legitimate solver S

$$\Pr[x \xleftarrow{\$} \mathsf{STR} : S(G(x)) \Rightarrow x] \geq \alpha,$$

and for any efficient machine C

$$\Pr[x \xleftarrow{\$} \mathsf{STR} : C(G(x)) \Rightarrow x] \leq \beta.$$

For a CAPTCHA to be secure it is required that there is a large gap between α and β. In Section 3.4, we will propose an alternative security definition for CAPTCHAs.

3.2.3 Secret Sharing Schemes

A secret sharing scheme is a method designed to share a secret between a group of participants. These schemes were first proposed by Shamir in 1979 [72]. Although there have been improvements to these kind of schemes, here we will use the basic construction due to Shamir. In a (u, w) threshold secret sharing scheme a secret K is divided into w pieces called *shares*. These w shares are given to w participants. To recover the secret, at least u of w shares, where $u \leq w$, are required. And it is not possible to recover the secret with less than u shares.

We describe the specific construction proposed by Shamir. To construct a (u, w) secret sharing scheme we need a prime $p \geq w+1$ and the operations take place in the field $\mathbb{Z}_p$. The procedure for splitting a secret K into w parts is depicted in the algorithm below:

```
$\mathsf{SHARE}^p_{u,w}(K)$
1. Choose $w$ distinct, non-zero elements of $\mathbb{Z}_p$, denote them as $x_i$, $1 \leq i \leq w$.
2. Choose $u-1$ elements of $\mathbb{Z}_p$ independently at random. Denote them as
   $a_1, \ldots, a_{u-1}$.
3. Let, $a(x) = K + \sum_{j=1}^{u-1} a_j x^j \bmod p$, and $y_i = a(x_i)$, $1 \leq i \leq w$.
4. Output $\mathcal{S} = \{(x_1, y_1), \ldots, (x_w, y_w)\}$ as the set of $w$ shares.
```

The secret K can be easily recovered using any $B \subset \mathcal{S}$ such that $|B| \geq u$, but if $|B| < u$ then K cannot be recovered. To see this, observe that the polynomial used in step 3 to compute the y_i's is a $u-1$ degree polynomial. Thus using u pairs of the type (x_i, y_i) one can generate u linear equations, each of the type $y_i = K + a_1 x_i + \cdots a_{u-1} x_i^{u-1}$. Using these equations the value of K can be found. It can be shown that this set of u equations would always have a unique solution.

3.3 Profiling Adversaries

Let $\mathcal{M}$ be a message space and let $\mathcal{P} = \{1, 2, \ldots, k\}$ be a set of labels for different possible profiles. We assume that each message x in $\mathcal{M}$ can be labeled by a unique $j \in \mathcal{P}$. Thus, there exists a function $f : \mathcal{M} \to \mathcal{P}$, which assigns a label to each message in the message space. In other words, we can assume that the message space can be partitioned into disjoint subsets as $\mathcal{M} = M_1 \cup M_2 \cup \cdots \cup M_k$ and for every $x \in \mathcal{M}$, $f(x) = i$ if and only if $x \in M_i$.

We call f as the profiling function or a classifier. Thus, in this setting we are assuming that each message in the message space $\mathcal{M}$ represents some profile, and messages in M_i, where $1 \leq i \leq k$, correspond to the profile i. The function f is a classifier which given a message can classify it into one of the profiles. We also assume that the function f is efficiently computable for every $x \in \mathcal{M}$, in particular, we assume that for any $x \in \mathcal{M}$, $f(x)$ can be computed in time at most μ, where μ is a constant.

The function f is public, thus given $x \in \mathcal{M}$ any adversary can efficiently compute $f(x)$. We want to define security for an encryption scheme which is secure against profiling adversaries, i.e., we want that when a message from $\mathcal{M}$ is encrypted using the encryption algorithm no efficient adversary would be able to profile it.

3.3.1 PROF-EAV Security

Here we propose a definition for encryption schemes secure against profiling adversaries.

Definition 3.3. [PROF-EAV security] *Let $\mathcal{M}$ be a message space and let $f : \mathcal{M} \to \mathcal{P}$ be a profiling function. Let $E : \mathcal{M} \times \mathcal{K} \to \mathcal{C}$ be an encryption algorithm. We define the advantage of an adversary $\mathcal{A}$ in the* PROF-EAV *(read profiling under eavesdropping) sense in breaking E as*

$$\mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{A}) = \Pr[\mathcal{A}(E_K(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)],$$

where $K \xleftarrow{\$} \mathcal{K}$, $x \xleftarrow{\$} \mathcal{M}$ and $\mathcal{A}'$ is an adversary whose running time is a polynomial of the running time of $\mathcal{A}$. An encryption algorithm $E : \mathcal{M} \times \mathcal{K} \rightarrow \mathcal{C}$ is called (ϵ, t) PROF-EAV *secure for a given profiling function f, if for all adversaries $\mathcal{A}$ running in time at most t, $\mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{A}) \leq \epsilon$.*

In the definition above, we want to capture the notion that for a PROF-EAV secure encryption scheme, an adversary $\mathcal{A}$ trying to find the profile of a message seeing its cipher cannot do much better than the best adversary $\mathcal{A}'$, who tries to guess the profile without seeing the ciphertext.

This definition is in accordance with the definition of semantic security as discussed in Section 3.2.1. Recall that an encryption scheme is called semantically secure if no adversary can efficiently compute *any* function of the plaintext given its ciphertext. But in the PROF-EAV definition we are interested only on a specific function f. Thus, PROF-EAV security is strictly weaker than semantic security. Semantic security trivially implies PROF-EAV security but PROF-EAV security does not imply IND-EAV security, we give a concrete example to illustrate this.

Example 3.1. *Let $\mathcal{M} = \{0,1\}^n = M_1 \cup M_2$ be a message space, where*

$$M_1 = \{x \in \mathcal{M} : \textit{first bit of } x \textit{ is 0}\},$$

and $M_2 = \mathcal{M} \setminus M_1$, and f be the profiling function such that $f(x) = i$ iff $x \in M_i$. Let E^{one} be an encryption scheme which uses a one bit key k (chosen uniformly from $\{0,1\}$) and given a message $x \in \mathcal{M}$ it xors k with the first bit of x. It is easy to see that an adversary trying to guess the profile of a message x given $E_k^{\text{one}}(x)$ cannot do better than with probability half, and this success probability can be achieved even without seeing the ciphertext, as here $|M_1| = |M_2|$. Hence E^{one} is PROF-EAV secure, but trivially not secure in the IND-EAV sense.

3.4 Encryption Protocol Secure Against Profiling Adversaries

In this section we describe a complete protocol which would be secure against profiling adversaries. Our motivation is to prevent communications getting profiled in large scale mechanically. The protocol is not secure from human adversaries, and we do not care much about that as we hope that it would be economically infeasible to employ a human for large scale profiling.

```
Protocol $\mathbb{P}(x)$
1. $k \overset{\$}{\leftarrow}$ STR;
2. $k' \leftarrow G(k)$;
3. $K \leftarrow H(k)$;
4. $c \leftarrow E_K(x)$;
5. return $(c, k')$
```

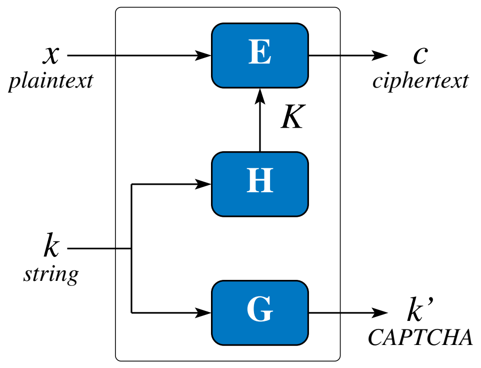


Figure 3.1: The protocol $\mathbb{P}$.

The protocol $\mathbb{P}$ consists of the following entities:

- The message space $\mathcal{M}$, the cipher space $\mathcal{C}$.
- The set of profiles $\mathcal{P}$ and the profiling function f associated with $\mathcal{M}$.
- A set STR which consists of short strings over a specified alphabet.
- An encryption scheme $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$.
- A hash function H : STR $\to \mathcal{K}$.
- A CAPTCHA generator G which takes inputs from STR.

Given a message $x \in \mathcal{M}$, $\mathbb{P}$ produces a ciphertext as shown in Figure 3.1. In the protocol as described in Figure 3.1, k, an element of STR is hashed to form the key K and k is also converted into a CAPTCHA and transmitted along with the ciphertext. The only input to $\mathbb{P}$ is the message and the key generation is embedded in the protocol. It resembles the scenario of hybrid encryption [2], which consists of two mechanisms called key encapsulation and data encapsulation where an encrypted version of the key is also transmitted along with the cipher. For a human decryption is easy, as given a ciphertext (c, k') a human user can recover k from k' by solving the CAPTCHA and thus compute $E^{-1}_{H(k)}(c)$ to decipher.

3.4.1 Security of $\mathbb{P}$

The security of a protocol $\mathbb{P}$ against profilers is defined in the same way as in Definition 3.3.

Definition 3.4. [PROF security] *The advantage of an adversary attacking protocol $\mathbb{P}$ is defined as*

$$\mathbf{Adv}^{\text{prof}}_{\mathbb{P},f}(\mathcal{A}) = \Pr[\mathcal{A}(\mathbb{P}(x)) \Rightarrow f(x)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)],$$

where $x \xleftarrow{\$} \mathcal{M}$ and $\mathcal{A}'$ is an adversary whose running time is a polynomial of the running time of $\mathcal{A}$. Additionally $\mathbb{P}$ is called (ϵ, t) secure in the PROF sense if for all adversaries running in time at most t, $\mathbf{Adv}^{\text{prof}}_{\mathbb{P},f}(\mathcal{A}) < \epsilon$.

The above definition is different from Definition 3.3 by the fact that it does not mention the key explicitly, as key generation is embedded in the protocol itself. To prove that $\mathbb{P}$ is secure in the PROF sense we need an assumption regarding the CAPTCHA G and the hash function H. We state this next.

Definition 3.5. [The Hash-Captcha Assumption] *Let G be a CAPTCHA generator, let r be a positive integer, let $H : \mathsf{STR} \to \{0,1\}^r$ be a hash function, and let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ in violating the Hash-Captcha assumption as*

$$\begin{aligned}\mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{A}) \;\; = \;\; & \Pr[x \xleftarrow{\$} \mathsf{STR} : \mathcal{A}(G(x), H(x)) \Rightarrow 1] \\ & - \Pr[x \xleftarrow{\$} \mathsf{STR}, z \xleftarrow{\$} \{0,1\}^r : \mathcal{A}(G(x), z) \Rightarrow 1].\end{aligned}$$

Moreover, (G, H) is called (ϵ, t)-HC secure if for all adversaries $\mathcal{A}$ running in time at most t, $\mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{A}) \leq \epsilon$.

This definition says that the pair formed by a CAPTCHA generator G and a hash function H is secure, if an adversary $\mathcal{A}$ is unable to distinguish between $(G(x), H(x))$, where x is some string, and $(G(x), z)$, where z is a random string. This security notion of a CAPTCHA inspired by the notion of *indistinguishability* is quite different from the (α, β) security notion as described in Section 3.2.2. Here the adversary has some more information regarding x through the value $H(x)$. If the adversary can efficiently solve the CAPTCHA G then it can break (G, H) in the HC sense irrespective of the hash function. Given the CAPTCHA is secure, i.e., no efficient adversary can find x from $G(x)$ still an adversary may be able to distinguish $H(x)$ from a string randomly selected from the range of H.

If we consider a keyed family of hash functions $\mathcal{H} = \{H_\ell\}_{\ell \in \mathcal{L}}$, such that for every $\ell \in \mathcal{L}$, $H_\ell : \mathcal{D} \to \mathcal{R}$ for some sets $\mathcal{D}$ and $\mathcal{R}$. Then $\mathcal{H}$ is called an entropy smoothing family if for any efficient adversary it is difficult to distinguish between $(\ell, H_\ell(x))$ and (ℓ, z), where ℓ, x, z are selected uniformly at random from $\mathcal{L}$, $\mathcal{D}$ and $\mathcal{R}$ respectively. An entropy smoothing hash along with a secure CAPTCHA can resist HC attacks. Entropy smoothing hashes can be constructed from universal hash functions using the left over hash lemma [38], but the parameter sizes which would be required for such provable guarantees can be prohibitive. We believe that using ad-hoc cryptographic hashes like the ones from the SHA family can provide the same security. In our definition we do not use a keyed family of hash functions, but such a family can be easily used in the protocol $\mathbb{P}$, and in that case the hash key will also be a part of the ciphertext.

With these discussions we are now ready to state the theorem about security of $\mathbb{P}$.

Theorem 3.1. *Let $\mathbb{P}$ be a protocol as in Figure 3.1 and let $\mathcal{A}$ be an adversary attacking $\mathbb{P}$ in the* PROF *sense. Then there exist adversaries $\mathcal{B}$ and $\mathcal{B}'$ such that*

$$\mathbf{Adv}^{\text{prof}}_{\mathbb{P},f}(\mathcal{A}) \leq \mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{B}) + \mathbf{Adv}^{\text{prof-eav}}_{E,f}(\mathcal{B}').$$

And, if $\mathcal{A}$ runs for time t, both $\mathcal{B}$ and $\mathcal{B}'$ run for time $O(t)$.

Proof. Let $\mathcal{A}$ be an adversary attacking the protocol $\mathbb{P}$ in Figure 3.1. We construct an adversary $\mathcal{B}$ attacking the hash-captcha (G, H), using $\mathcal{A}$ as follows.

```
Adversary $\mathcal{B}(G(k), z)$
1. $x \xleftarrow{\$} \mathcal{M}$;
2. Send $(E_z(x), G(k))$ to $\mathcal{A}$;
3. $\mathcal{A}$ returns $j$;
4. if $f(x) = j$;
5.     return 1;
6. else return 0;
```

As $\mathcal{B}$ is an adversary attacking the hash-captcha assumption, hence there are two possibilities regarding the input $(G(k), z)$ of $\mathcal{B}$, z can either be $H(k)$ or a uniform random element in $\mathcal{K}$, and the goal of $\mathcal{B}$ is to distinguish between these two possibilities.

Considering the first possibility that z is $H(k)$, the way the adversary $\mathcal{B}$ is defined, $\mathcal{A}$ gets a valid encryption of the message x (which is a random element in the message space) according to the protocol $\mathbb{P}$. Hence we have

$$\begin{aligned} & \Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), H(k)) \Rightarrow 1] \\ = \; & \Pr[k \xleftarrow{\$} \mathcal{K}, x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_{H(k)}(x), G(k)) \Rightarrow f(x)] \\ = \; & \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(\mathbb{P}(x)) \Rightarrow f(x)]. \end{aligned} \tag{3.2}$$

Similarly, for the second possibility, i.e., when the input z to $\mathcal{B}$ is an element chosen uniformly at random from $\mathcal{K}$, we have

$$\begin{aligned} & \Pr[k, K \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), K) \Rightarrow 1] \\ = \; & \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_K(x), G(k)) \Rightarrow f(x)]. \end{aligned} \tag{3.3}$$

In Equation (3.3), k and K are chosen independently uniformly at random from $\mathcal{K}$. Thus, the adversary $\mathcal{A}$ has as input $E_K(x)$ and $G(k)$, where k is independent of K, thus $G(k)$ carries no information about K. Hence $\mathcal{A}$ cannot do better than some PROF-EAV adversary $\mathcal{B}'$ who has only $E_K(x)$ as its input, and runs for same time as that of $\mathcal{A}$. Thus

$$\begin{aligned} & \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_K(x), G(k)) \Rightarrow f(x)] \\ & \quad \leq \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_K(x)) \Rightarrow f(x)] \end{aligned} \tag{3.4}$$

From definition of PROF-EAV advantage of $\mathcal{B}'$ we have

$$\begin{aligned} & \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_K(x)) \Rightarrow f(x)] \\ = \; & \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)] \end{aligned} \tag{3.5}$$

Thus, using Equations (3.3), (3.4) and (3.5) we have

$$\begin{aligned} & \Pr[k, K \xleftarrow{\$} \mathcal{K} : \mathcal{B}(G(k), K) \Rightarrow 1] \\ \leq \; & \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)] \end{aligned} \tag{3.6}$$

Finally, from Equations (3.2) and (3.6) and Definitions 3.5 and 3.4 we have

$$\mathbf{Adv}_{\mathbb{P},f}^{\text{prof}}(\mathcal{A}) \leq \mathbf{Adv}_{G,H}^{\text{hc}}(\mathcal{B}) + \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}'),$$

as desired. Also if $\mathcal{A}$ runs for time t, then $\mathcal{B}'$ runs for time t and $\mathcal{B}$ runs for time $t + c$ for some small constant c. □

Some remarks about security of $\mathbb{P}$: We defined the security of the protocol $\mathbb{P}$ for only a fixed profiling function f, but note that we can modify the definition for any arbitrary function f which would give us a security definition equivalent to SEM-EAV (discussed in Section 3.2.1). If the encryption algorithm E used within the protocol is SEM-EAV secure then using the same proof we can obtain SEM-EAV security for $\mathbb{P}$.

```
Protocol $\mathbb{P}'(x)$
1. $k \xleftarrow{\$} \mathsf{STR}$;
2. $k' \leftarrow \mathsf{ENCD}(k, 0)$;
3. $\{(x_1, k'_1), \ldots, (x_w, k'_w)\} \leftarrow \mathsf{SHARE}^p_{u,w}(k')$;
4. for $i = 1$ to $w$;
5.     $(k_i, \lambda_i) \leftarrow \mathsf{ENCD}^{-1}(k'_i)$;
6.     $c_i \leftarrow G(k_i)$;
7. end for
8. $K \leftarrow H(k)$;
9. $C \leftarrow E_K(x)$;
10. return $[C, \{(x_1, c_1, \lambda_1), \ldots, (x_w, c_w, \lambda_w)\}]$
```

Figure 3.2: The protocol $\mathbb{P}'$ which uses a secret-sharing scheme.

3.5 A Practical Instantiation

A very common problem using CAPTCHAs is that sometimes even humans may fail to solve them. As in the protocol $\mathbb{P}$ if a human user fails to solve the CAPTCHA then he will not be able to decipher and there is no way to repeat the test (as it is done in normal CAPTCHA usage), hence this stands as a serious weakness of the proposed protocol $\mathbb{P}$. A solution to this problem can be attempted by providing some redundancy in the CAPTCHAs so that a valid user can have more chance in solving the CAPTCHA. As a solution we propose that the initial string k chosen by the protocol is broken into w shares such that with u or more of the shares would be enough to generate k. These w shares are converted into CAPTCHAs and sent along with the ciphertext. To incorporate this idea we changed the initial protocol $\mathbb{P}$ to $\mathbb{P}'$. The protocol $\mathbb{P}'$ is a specific instantiation, thus before we describe the protocol we fix some details of its components, in particular for $\mathbb{P}'$ we would require an encoding mechanism ENCD which we discuss first.

Let $\mathsf{AL} = \{A, B, \ldots, Z\} \cup \{a, b, \ldots, z\} \cup \{0, 1, \ldots, 9\} \cup \{+, /\}$, thus making $|\mathsf{AL}| = 64$. We define an arbitrary (but fixed) bijection $\rho : \mathsf{AL} \to \{0, 1, \ldots, 63\}$, and for any $\sigma \in \mathsf{AL}$ and $n \geq 6$, $\mathsf{bin}_n(\sigma)$ will denote the n bit binary representation of $\rho(\sigma)$. Note that for all $\sigma \in \mathsf{AL}$, at most 6 bits are required to represent $\rho(\sigma)$. If ψ is a binary string, then let $\mathsf{toInt}(\psi)$ be the positive integer corresponding to ψ, similarly for a positive integer $v < 2^n$, $\mathsf{toBin}_n(v)$ denotes the n bit binary representation of v. We fix a positive integer m and let STR be the set of all m character strings over the alphabet AL. Let p be the smallest prime greater than 2^{6m} and let $d = p - 2^{6m}$. Let $\mathsf{ENCD} : \mathsf{STR} \times \{0, 1, ..., d\} \to \mathbb{Z}_p$ be defined as follows

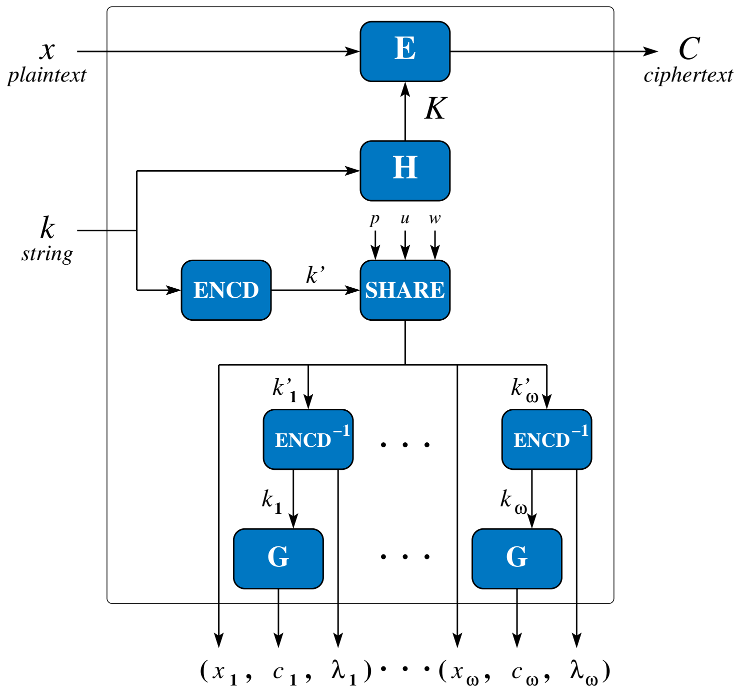


Figure 3.3: Diagram of Protocol $\mathbb{P}'$.

```
ENCD$(s, \lambda)$
1. Parse $s$ as $\sigma_0||\sigma_1||\ldots||\sigma_m$, where each $\sigma_i \in$ AL;
2. $\psi \leftarrow \mathsf{bin}_6(\sigma_0)||\ldots||\mathsf{bin}_6(\sigma_m)$;
3. $v \leftarrow \mathsf{toInt}(\psi)$;
4. return $v + \lambda$;
```

And let $\mathsf{ENCD}^{-1} : \mathbb{Z}_p \to \mathsf{STR} \times \{0, 1, \ldots, d\}$ be defined as

```
ENCD$^{-1}(y)$
1. if $y \geq 2^{6m}$,
2.     $\lambda \leftarrow y - 2^{6m} + 1$;
3.     $y \leftarrow 2^{6m} - 1$;
4. else $\lambda \leftarrow 0$;
5. $z \leftarrow \mathsf{toBin}_{6m}(y)$;
6. Parse $z$ as $z_0||z_1||\ldots||z_m$, where $|z_i| = 6$;
7. $s \leftarrow \rho^{-1}(\mathsf{toInt}(z_0))||\ldots||\rho^{-1}(\mathsf{toInt}(z_m))$;
8. return $(s, \lambda)$;
```

The modified protocol $\mathbb{P}'$ is shown in Figure 3.2. It uses the encoding function ENCD and the secret sharing scheme as depicted in Section 3.2.3. For $\mathbb{P}'$ we assume that STR contains all m character strings over the alphabet AL, and p is the smallest prime greater than 2^{6m}, these can be considered the fixed and public parameters for $\mathbb{P}'$. The encoding mechanism is specifically designed to convert a string in STR to an element in $\mathbb{Z}_p$ so that Shamir's secret sharing can be suitably used.

To decrypt a cipher produced by $\mathbb{P}'$ a human user must solve at least some u of w CAPTCHAs. Using these u solutions together with x_i, k can be recovered. A specific recommendation for SHARE can be Shamir (2,5)-threshold scheme. Thus the user would have much flexibility on solving the CAPTCHAs.

3.5.1 Security of $\mathbb{P}'$

The security of $\mathbb{P}'$ can be easily proved in the sense of Definition 3.4 in a similar way as we prove Theorem 3.1 if we make a new assumption regarding the CAPTCHA as follows:

Definition 3.6. [The Hash-MultiCaptcha (HMC) Assumption] *Let G be a CAPTCHA generator, let r be a number, let $H : \mathsf{STR} \to \{0,1\}^r$ be a hash function, and let $\mathcal{A}$ be an adversary. Also, let $x = g(x_1, \ldots x_w)$ be such that if at least u out of w of $x_1, \ldots, x_w$ are known then x can be recovered. We define the advantage of $\mathcal{A}$ in violating the Hash-MultiCaptcha assumption as*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{hmc}}_{G,H}(\mathcal{A}) &= \Pr[\mathcal{A}(G(x_1), \ldots G(x_w), H(x)) \Rightarrow 1] \\ &\quad - \Pr[z \overset{\$}{\leftarrow} \{0,1\}^r : \mathcal{A}(G(x_1), \ldots G(x_w), z) \Rightarrow 1].\end{aligned}$$

where $x \overset{\$}{\leftarrow} \mathsf{STR}$. Moreover, (G, H) is called (ϵ, t)-HMC secure if for all adversaries $\mathcal{A}$ running in time at most t, $\mathbf{Adv}^{\mathrm{hmc}}_{G,H}(\mathcal{A}) \leq \epsilon$.

As in the definition of Hash-Captcha assumption, in this definition if the adversary can efficiently solve at least u of w CAPTCHAs, then it can break (G, H) in the HMC sense irrespective of the hash function. If this assumption is true, then we can show the security of protocol $\mathbb{P}'$ just as we did for protocol $\mathbb{P}$.

A CAPTCHA is an example of a weakly-verifiable puzzle [16], since a legitimate solver S may not be able to verify the correctness of its answer. For this kind of puzzles, it has been proved [37] that if it is difficult for an attacker to solve a weakly-verifiable puzzle P, then trying to solve multiple instances of a puzzle in parallel is harder. Most recently, Jutla found a better bound to show how hard it is for an attacker to solve multiple instances of weakly-verifiable puzzles [41]. The next theorem is based on the main theorem proposed by Jutla, but it has been adapted to CAPTCHAs, which are of our interest in this work.

Theorem 3.2. *Let G be a CAPTCHA generator which is (α, β) secure. Let $k \in \mathbb{N}$, $\delta = 1 - \beta$ and γ $(0 < \gamma < 1)$ be arbitrary. Let $\mathcal{A}$ be an arbitrary polynomial time adversary, which is given as input k CAPTCHAs $(G(x_1), \ldots, G(x_k))$ and outputs a set X of solutions of the k CAPTCHAs. If $\mathsf{InCorr}(X)$ denotes the number of incorrect solutions in X, then*

$$\Pr[\mathsf{InCorr}(X) < (1-\gamma)\delta k] < e^{-(1-\gamma)\gamma^2 \delta k/2}$$

This theorem establishes that for any adversary if the probability of failure in solving a CAPTCHA is at least δ, then the probability of failing on less than $(1-\gamma)\delta k$ out of k puzzles, is at most $e^{-(1-\gamma)\gamma^2\delta k/2}$.

Based on this fact, it may be possible to show that for any arbitrary adversary $\mathcal{A}$ attacking the HMC assumption, there exists a HC adversary $\mathcal{B}$ such that $\mathbf{Adv}^{\text{hmc}}_{G,H}(\mathcal{A}) < \mathbf{Adv}^{\text{hc}}_{G,H}(\mathcal{B})$. This would imply that the HC assumption implies the HMC assumption. But, for now we are not sure whether such a result holds.

3.5.2 Discussions

- **About the encryption scheme:** Till now we have not given details about the encryption scheme to be used in the protocol. We only mentioned that we require our encryption scheme to be PROF-EAV secure and any IND-EAV secure encryption scheme can provide such security. Thus most symmetric encryption schemes which are usually in use like CBC mode, counter mode etc. (which provide security in the IND-CPA sense) can be used for the encryption function E in $\mathbb{P}'$. A more efficient scheme which provides security only in the PROF-EAV sense would be much interesting, we would like to explore in this direction.

- **Key sizes:** Another important thing to consider is that the effective size of a key for the protocol is dictated by the parameter m, i.e., the size of each string in STR. This value cannot be made arbitrarily large as solving big CAPTCHAs for human beings may be tiresome, a usual CAPTCHA length is five to eight characters. If we use eight character strings from the alphabet AL then the effective size of the key space would be 2^{48}. Increasing the alphabet size is also not feasible as we need un-ambiguous printable characters to make CAPTCHAs. Thus, the key space is not sufficiently large for a modern cryptographic application, but for the application which we have in mind this may be sufficient, as we do not expect that a profiler would be ready to use so much computational resource for profiling a single message.

- **Usability:** Given that our main goal is securing email communications, our protocol $\mathbb{P}$ has a main drawback: a user must solve two CAPTCHAs for each message in the mailbox to read them. This could be tedious even for a small number of emails. We will discuss some alternatives in Chapter 7. These alternatives still use CAPTCHAs, but now the user is required to solve more than one CAPTCHA, only if he is unable to solve the CAPTCHA in his first attempt.

3.6 Analyzing the Golle and Farahat's scheme

As we mentioned in the Introduction, a previous work on the issue of securing email communication from profilers was proposed by Golle and Farahat [35]. The basic idea behind their proposal is to hash the header of an email message to generate the key, and further use the key to encrypt the payload of the message. They require that the encryption algorithm should be resistant to profiling attacks. In addition, they assume some properties of the hash function for assuring security of the scheme. In particular they mention that the hash function used for deriving the key should be a *slow one-way hash* function. They argued that the security of the scheme relies on the high computational cost for the hash function and on (some weak) privacy of the encryption scheme. Though it is not clear, why the property of one-wayness of the hash is important in this context. They claimed that though their scheme do not satisfy the semantic security notion, but it would provide adequate security for the application. No proper security analysis of the scheme is done in the paper. In this section we apply the security notions that we previously defined to establish the security of the Golle and Farahat's scheme.

In Figure 3.4 we show a slightly different version of Golle and Farahat's scheme. Here instead of using the header to derive the key we are considering a random string (line 1). We assume that $H : \mathsf{STR} \to \mathcal{K}$, and E an encryption scheme with key space $\mathcal{K}$ and message space $\mathcal{M}$. Now we proceed to analyze it.

3.6.1 Slow Hash Functions

Slow hash functions have been previously used to protect passwords. The main purpose of this kind of hash functions is preventing *off-line* attacks, where a list of password hashes is stolen and the attacker tries to guess the password by testing all possible

```
Protocol $\mathbb{S}(x)$
1. $s \xleftarrow{\$} \mathsf{STR}$;
2. $K \leftarrow H(s)$;
3. $c \leftarrow E_K(x)$;
4. return $(c, s)$
```

Figure 3.4: Protocol proposed by Golle and Farahat to protect email communications

candidates and verifying if the result matches. If a hash function is slow enough, then guessing a password would be harder, since an attacker must try many candidates and the cost of doing this will be high. Usually a slow one-way hash function can be constructed by iterating cryptographic hashes (such as SHA1) multiple times. There have been some interesting proposals like bcrypt [63] and scrypt [60] which are designed to achieve the slow-ness goal. Moreover, these functions can be tuned to achieve various degrees of slowness as desired. More recently another interesting option to construct slow hash functions appeared which spends computing cycles to solve other computational problems [29]. All these proposals can be suitably used in the context of the protocol $\mathbb{S}$.

3.6.2 Security of Protocol $\mathbb{S}$

It is to be noted that the hash function H used in $\mathbb{S}$ is public, hence any adversary can derive the key from the ciphertext and hence decrypt the message. The type of security that is expected from the protocol $\mathbb{S}$ is a bit different from that expected from usual encryption schemes. A profiling adversary would be successful if it can profile "many" messages, in particular we can think that for the activity of profiling to be economically profitable an adversary has to profile at least N different messages in a day (where N can be in the order of millions).

The main argument in [35] was to choose such a hash function which is "slow". Suppose the evaluation of one hash requires 5 seconds, then an adversary would not be able to profile more than 17280 messages in a day. Whereas for a normal user, this "slow"-ness of the hash function does not have a significant effect, as (s)he would need to decrypt far less than say 100 messages in a day.

It is obvious that the security of the scheme $\mathbb{S}$ critically depends on the time an adversary spends to break the scheme. Restricting the running time of the adversary is always essential in any cryptographic scheme which is not information theoretically secure, for example in all the previous security definitions we talked of (ϵ, t)-secure schemes. In the definitions that we would use to argue about the security of $\mathbb{S}$, the running time restriction would be of central importance, hence for convenience we define a time restricted adversary as follows.

Definition 3.7. [T-restricted adversary] *A T-restricted adversary is an adversary which runs for time at most T.*

Next we formalize the notion of a "slow hash function".

Definition 3.8. [T-slow hash function] *Let $H : \mathsf{STR} \to \{0,1\}^\ell$ be a hash function. If the time required to compute $H(x)$ for every $x \in \mathsf{STR}$ is at least T then we say that H is a T-slow hash function.*

For $\mathbb{S}$ to be secure, we need the hash function to be "well behaved" in some sense in addition to being slow. The specific property that we would require is that given x, there should be no other feasible way to predict $H(x)$ other than computing the value of $H(x)$. We formalize this notion in the following definition.

Definition 3.9. [(ϵ, T)-indistinguishable hash function] *Let $H : \mathsf{STR} \to \{0,1\}^\ell$ be a T-slow hash function, let $\mathcal{A}$ be a T-restricted adversary, and $y \xleftarrow{\$} \{0,1\}^\ell$. We define the advantage of $\mathcal{A}$ in distinguishing (x, y) from $(x, H(x))$ as*

$$\begin{aligned}\mathbf{Adv}_H^{\text{ind}}(\mathcal{A}) \;&=\; \Pr[x \xleftarrow{\$} \mathsf{STR} : \mathcal{A}(x, H(x)) \Rightarrow 1)] \\ &\quad - \Pr[x \xleftarrow{\$} \mathsf{STR}, y \xleftarrow{\$} \{0,1\}^\ell : \mathcal{A}(x,y) \Rightarrow 1].\end{aligned}$$

H is (ϵ, T)-indistinguishable if for all T-restricted adversaries $\mathcal{A}$, $\mathbf{Adv}_H^{\text{ind}}(\mathcal{A}) \leq \epsilon$.

With this we are ready to establish the security of protocol $\mathbb{S}$ in the PROF sense (see Definition 3.4) as we did with protocols $\mathbb{P}$ and $\mathbb{P}'$.

Theorem 3.3. *Let $\mathbb{S}$ be a protocol as in Figure 3.4. Let us assume that for all $x \in \mathcal{M}$ to compute $E_k(x)$ one requires a constant time t_c, and recall that computing $f(x)$ for every $x \in \mathcal{M}$ requires μ time. If H is $(\epsilon_1, T + t_c + \mu)$-indistinguishable, and E is (ϵ_2, t)-*PROF-EAV *secure, then $\mathbb{S}$ is $(\epsilon_1 + \epsilon_2, T)$-*PROF *secure, where $T \leq t$.*

Proof. To prove the Theorem we will construct a $(T + t_c + \mu)$-restricted adversary $\mathcal{B}$ which breaks the hash function H in the indistinguishability sense by using a T-restricted adversary $\mathcal{A}$ which attacks $\mathbb{S}$. Subsequently we show that

$$\mathbf{Adv}_{\mathbb{S},f}^{\text{prof}}(\mathcal{A}) \leq \mathbf{Adv}_H^{\text{ind}}(\mathcal{B}) + \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}'),$$

where $\mathcal{B}'$ is a t restricted adversary. This asserts the Theorem.

Let $\mathcal{A}$ be an adversary attacking the protocol $\mathbb{S}$ in Figure 3.4. We construct an adversary $\mathcal{B}$ attacking the hash H, using $\mathcal{A}$ as follows.

```
Adversary $\mathcal{B}(s,k)$
1. $x \overset{\$}{\leftarrow} \mathcal{M}$;
2. Send $(E_k(x), s)$ to $\mathcal{A}$;
3. $\mathcal{A}$ returns $j$;
4. if $f(x) = j$;
5.     return 1;
6. else return 0;
```

As $\mathcal{B}$ is an adversary attacking the indistinguishability of the hash function, it receives as input a pair (s,k), hence there are two possibilities regarding the input (s,k) of $\mathcal{B}$, k can either be $H(s)$ or a uniform random element in $\{0,1\}^\ell$, and the goal of $\mathcal{B}$ is to distinguish between these two possibilities.

Considering the first possibility that k is $H(s)$, the way the adversary $\mathcal{B}$ is defined, $\mathcal{A}$ gets a valid encryption of the message x (which is a random element in the message space) according to the protocol $\mathbb{S}$. Hence we have

$$\begin{aligned}\Pr[s \overset{\$}{\leftarrow} \mathsf{STR}\text{: } \mathcal{B}(s, H(s)) \Rightarrow 1] &= \Pr[s \overset{\$}{\leftarrow} \mathsf{STR},\ x \overset{\$}{\leftarrow} \mathcal{M} : \mathcal{A}(E_{H(s)}(x), s) \Rightarrow f(x)] \\ &= \Pr[x \overset{\$}{\leftarrow} \mathcal{M} : \mathcal{A}(\mathbb{S}(x)) \Rightarrow f(x)]. \qquad (3.7)\end{aligned}$$

Similarly, for the second possibility, i.e., when the input k to $\mathcal{B}$ is an element chosen uniformly at random from $\{0,1\}^\ell$, we have

$$\Pr[s \overset{\$}{\leftarrow} \mathsf{STR}\ , k \overset{\$}{\leftarrow} \{0,1\}^\ell : \mathcal{B}(s,k) \Rightarrow 1] = \Pr[x \overset{\$}{\leftarrow} \mathcal{M} : \mathcal{A}(E_k(x), s) \Rightarrow f(x)]. \qquad (3.8)$$

In Equation (3.8), s and k are not related at all, since s is chosen independently at random from STR and k is chosen independently uniformly at random from $\{0,1\}^\ell$. Thus, the adversary $\mathcal{A}$ has as input $E_K(x)$ and s, where s is independent of k, thus k cannot be derived from s. Hence $\mathcal{A}$ cannot do better than some PROF-EAV adversary $\mathcal{B}'$ who has only $E_k(x)$ as its input, and runs for same time as that of $\mathcal{A}$, which must be less than T. Thus

$$\Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{A}(E_k(x), s) \Rightarrow f(x)] \quad \leq \quad \Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_k(x)) \Rightarrow f(x)] \qquad (3.9)$$

From definition of PROF-EAV advantage of $\mathcal{B}'$ we have

$$\begin{aligned} &\Pr[x \xleftarrow{\$} \mathcal{M} : \mathcal{B}'(E_K(x)) \Rightarrow f(x)] \\ &\quad = \quad \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)], \end{aligned} \qquad (3.10)$$

for any t restricted adversary for E. Thus, using Equations (3.8), (3.9) and (3.10) we have

$$\begin{aligned} &\Pr[s \xleftarrow{\$} \mathsf{STR}\ , k \xleftarrow{\$} \{0,1\}^{\ell} : \mathcal{B}(s,k) \Rightarrow 1] \\ &\quad \leq \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}') + \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)] \end{aligned} \qquad (3.11)$$

Finally, from Equations (3.7) and (3.11) and Definitions 3.4 and 3.9 we have

$$\mathbf{Adv}_{\mathbb{S},f}^{\text{prof}}(\mathcal{A}) \leq \mathbf{Adv}_{H}^{\text{ind}}(\mathcal{B}) + \mathbf{Adv}_{E,f}^{\text{prof-eav}}(\mathcal{B}'),$$

as desired. □

3.6.3 More Considerations

In a real scenario, a profiler will see a large amount of messages and will try to profile them. The restriction that we want to put on an adversary is that (s)he should not be able to profile more than N^* messages in time less than Γ, where the values of N^* and Γ would be decided based on the application.

If $\mathcal{A}$ be a Γ restricted sequential adversary, and $\mathsf{NProf}_{\Gamma}^{\mathcal{A}}$ be a random variable denoting the number of messages profiled correctly by $\mathcal{A}$ within time Γ. It is desirable that for all sequential adversaries $\mathcal{A}$, $\Pr[\mathsf{NProf}_{\Gamma}^{\mathcal{A}} \geq N^*] \leq \delta$, for some small constant δ.

It is easy to see that the following result holds.

Proposition 3.1. *Let $\mathbb{S}$ be a (ϵ, T) PROF secure scheme, and let Γ be a number. Then, for any Γ restricted sequential adversary $\mathcal{A}$*

$$\Pr\left[\mathsf{NProf}_{\Gamma}^{\mathcal{A}} \geq \frac{\Gamma}{T}\right] \leq \epsilon.$$

Thus, assuming a sequential adversary and adjusting the slowness of the hash function H, one can obtain the desired security objective out of $\mathbb{S}$.

We assume a sequential adversary, which may not be realistic. As an adversary may try to profile several messages in the same time. But, an adversary using more parallelism uses more computation power in terms of amount of hardware used, etc. Thus, instead of restricting the adversary on computation time, one can put a restriction on the total *computation cost*, and based on this cost metric it is possible to develop a security model where the restriction on sequential adversaries can be removed.

3.7 Concluding Remarks

We have done a theoretical analysis of profiling adversaries and ultimately described a protocol which is secure against profiling adversaries. Our protocol does not require any key exchange or public key infrastructure and uses CAPTCHAs and secret sharing schemes in a novel way. We also applied our definitions and methodologies to analyze security of an existing scheme.

Encryption may not be the only way to protect a user from profilers. As profilers can use many different techniques which cannot be stopped using encryption. For example it is possible to track the web usage of a specific user and profile him on that basis. Here (probably) encryption has no role to play, or at least cannot be used in the way we propose in our protocol. Anonymity is probably the correct direction to explore in solving such problems. Also, as user profiling is a big business, and some think that the free content in the web is only possible due to online advertisements, so putting a total end to user profiling may not be desirable. So there have been current attempts to develop systems which would allow targeted advertisements without compromising user security [77]. These issues are not covered in our current work.

As we said previously, here we did not mention anything about the encryption scheme which is secure in the PROF-EAV sense, in the next chapter we propose a cryptographic construction that can be useful against profilers.

Chapter 4

A PROF-EAV Secure Encryption Scheme

In the previous Chapter we mentioned that the main purpose of a profiling adversary is to classify messages into some pre-defined profiles. We also proposed two complete protocols which achieve PROF security. These protocols require a PROF-EAV secure encryption scheme. In this chapter we aim to construct a PROF-EAV secure encryption scheme, which can successfully fool classifiers. The scheme that we propose here is very different from existing encryption schemes. We can see this if we consider two facts: first our protocol will only encrypt text messages and second we are not really "encrypting" in the usual way. We first give an intuitive overview of the idea behind our construction.

A *spam filter* is a tool which can classify an email into two classes, namely, spam and non-spam. Most modern email services are equipped with such a tool, and with the current technology spam filters perform reasonably good. However, these tools are not perfect and one of the most important objectives of spammers is to design techniques such that they can fool a spam filter, i.e., the sent message would not be classified as a spam. To illustrate this situation let us consider how a popular method to combat spam works and how a spammer can sneak it. A popular method to detect spam is Bayesian filtering, which ranks words according to the probability of appearing in a spam message. Those words that appear in a spam message, will have a high score and those words that occur in a legitimate email will have a low score. When the system receives an email, it breaks it into words (which already had a score), using this the system computes a score for the whole email message. If the email has more spam words than legitimate words, then it will be classified as spam.

A traditional technique, which has been heavily used by spammers, consists of adding extra words to the original message. These extra words usually do not appear in spam messages. For example, it is common to see emails which have two distinct parts, where the first part advertises Viagra but it ends with a passage from Shakespeare. A human receiver of the message generally has no problem to understand the intent of the sender, but an automated tool like a spam filter may get easily deceived by such a message. Informally, the technique is used to conceal the original message within garbage, so that a specific objective is met. The objective is to fool a machine (a computer program) which tries to make some sense of the message but to convey the message to a human.

Rivest in [65] proposed a scheme called *Chaffing and Winnowing* (CW) for secure communication. CW also uses the idea of hiding a message within garbage. We build on the basic idea of CW in several ways, and we propose two concrete encryption schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$. Moreover, we prove them to be PROF-EAV secure under some assumptions. Both $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are restricted in the sense that their message space consists of text in some natural language. To build $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ we extensively use some ideas of document classification, we know of no encryption scheme which has been developed using ideas from document classification.

In what follows, in Section 4.1 we discuss the basic syntax of message authentication codes, the basic idea of chaffing and winnowing as proposed in [65] and some ideas of document classification. In Section 4.2 we discuss two schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$, which are built on the original scheme proposed by Rivest. In Section 4.3 we present some preliminary theoretical analysis involving the schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$. Finally in Section 4.4 we present some experiments.

4.1 Preliminaries

Before introducing our PROF-EAV scheme, we will describe the Chaffing and Winnowing scheme proposed by Rivest. This scheme uses a cryptographic mechanism known as a *message authentication code* (MAC), which provides authentication to the messages, in the private key setting. We also describe some important concepts of document classification, that we employ to develop our scheme.

4.1.1 Message Authentication Code

A *message authentication code* (MAC) is a cryptographic object, whose main purpose is to detect whether a message has been modified by an adversary. For this purpose, authorized parties must share a secret key K in advance, this key must remain unknown to the adversary. If two users wish to communicate in an authenticated way, they must perform the following steps. To send a message M to the receiver, the sender must compute a *tag* based on the message and the shared key K, this is done using a *tag-generation algorithm*. The pair composed of the message and the tag, (M, tag) is sent to the receiver. Then the receiver must *verify* that the tag is valid on the message M. To do this, a *verification algorithm* is used, which takes as input the key K, the message M and the tag, the output indicates whether the given tag is valid or not.

Formally, a message authentication code (or MAC) is a tuple of algorithms (Gen, Mac, Vrfy), such that

- The key-generation algorithm Gen takes as input the security parameter 1^n and outputs a key K from a pre-defined key space $\mathcal{K}$.
- The tag-generation algorithm Mac takes as input a key $K \in \mathcal{K}$ and a message $M \in \{0,1\}^*$ and outputs a tag.
- The verification algorithm Vrfy takes as input a key K, a message $M \in \{0,1\}^*$ and a tag. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning $\mathsf{invalid}$.

  Generally the verification algorithm also uses the tag generation algorithm, i.e., on receiving (M, tag), it computes $\mathsf{tag}' = \mathsf{Mac}_K(M)$ and if $\mathsf{tag} = \mathsf{tag}'$ then it returns a 1 else returns a 0.

The most popular MAC is HMAC [5], but exist others like CBC-MAC [40], and UMAC [12].

4.1.2 Chaffing and Winnowing

Chaffing and Winnowing (CW) is an interesting scheme which was proposed by Rivest in [65] to secure communications over a public channel. The basic idea is the following: Suppose we have a message $M = w_1||w_2||\dots||w_m$, where each $|w_i| = n$ bits , from M we create a new message M', by adding random n-bit blocks in random positions of the original message M, i.e., we obtain $M' = w'_1||w'_2||\dots||w'_{m'}$, where $m' > m$, and for each $i \in \{1, 2, \dots, m'\}$, either $w'_i = w_j$ for some $j \in \{1, 2, \dots, m\}$, or it is a random n-bit string. For convenience, we distinguish the two possibilities, we would say that $\mathsf{ty}(w'_i) = 0$ if w'_i is a random n-bit string, and say $\mathsf{ty}(w'_i) = 1$ otherwise. In addition to M', the scheme creates $\mathsf{tag} = t_1||t_2||\dots||t_{m'}$, where, $t_i = \mathsf{Mac}_K(w_i)$, if $\mathsf{ty}(w'_i) = 1$ and otherwise t_i is a random string. The pair (M', tag) is sent to the receiver. If the receiver shares the key K with the sender, then it checks if $\mathsf{Mac}_K(w_i) = t_i$, if this is the case then w_i is retained and if $\mathsf{Mac}_K(w_i) \neq t_i$, then w_i is discarded. Thus the receiver with the knowledge of the secret key can recover the original message M from (M', tag) with high probability.

The procedure for creating M' from M is called as *Chaffing*, and the reverse procedure of recovering the message by removing the chaff is called *Winnowing*. These terms are derived from the ancient procedure of "*winnowing the chaff from the grain*". Rivest in [65] discusses several interesting properties of this scheme. He argues that CW falls in the borderline of encryption and steganography, as in the procedure no encryption of the original data is performed, what is done is to hide the message within the chaff. And a valid receiver can extract the message from the chaff. He also argued that this procedure may find use in secret communication in scenarios where encryption is not allowed by law. There have been a few further studies on CW, which includes [4, 18].

Bellare and Boldyreva in [4], analyze the security of this scheme, and they proved that when $n = 1$, i.e., if we divide the message in blocks of one bit, the scheme is IND-CPA secure, but it is inefficient. Also, they proposed a modification to this scheme to make it efficient and IND-CPA secure. For this reason, we do not consider to use it against profiling adversaries, which as we have discussed require a more adequate notion of security.

4.1.3 Document Classification.

The problem of *document classification* or *document categorization* is as follows: given a set of classes and a document, determine in which class the given document belongs. Document classification is an active research area and has many applications in different domains such as email spam filtering, sentiment detection, information retrieval, among others.

Let $\mathscr{X}$ be the set of all possible documents of interest, called as the *document space*. And let $\mathscr{C} = \{c_1, c_2, \ldots c_j\}$ be a fixed set of j *classes*, also called *categories* or *labels*. Let $\mathcal{L}$ be the set of labeled documents, i.e., documents from $\mathscr{X}$ whose classes are known. We call $\mathcal{L}$ as the *training set*. The problem of designing a document classifier is to find a function $\gamma : \mathscr{X} \to \mathscr{C}$, such that γ can predict with high accuracy the class of all documents present in $\mathscr{X}$. Note, that the training set $\mathcal{L}$ is a very small subset of $\mathscr{X}$, and the information in $\mathcal{L}$ is used to design γ. The procedure of designing γ from $\mathcal{L}$ is commonly known as a *learning algorithm*.

There is a variety of learning algorithms which can be applied for document classification, for example, Naive-Bayes classifier, Support Vector Machines, k-nearest neighbor algorithm, decision tree, etc. [28, 50]. For this work we use two of them: Naive-Bayes classifier and k-nearest neighbor. Now we will briefly describe the main characteristics of these two algorithms.

Naive-Bayes Classifier. The Naive-Bayes algorithm is a probabilistic method based on the Bayes Theorem. In this algorithm we make several assumptions. First of all, we assume a very simple representation of a document, which is considered as a sequence of n words or terms. That is to say, the model considers that a document is generated by repeatedly drawing one word out of a *bag of words*. It is assumed that each word w has a different probability of occurrence for a document in each class, thus we can talk of the class conditional probability of occurrence of a word w in a certain document in class $c \in \mathscr{C}$, we denote this probability by $p(w|c)$. Moreover, it is assumed that each class $c \in \mathscr{C}$ has a certain probability of occurrence called the apriori probability of class c, this is denoted by $p(c)$. If we assume that for all possible words w and all classes $c \in \mathscr{C}$, the probabilities $p(c)$ and $p(w|c)$ are known, then we can come up with a decision rule to decide in which class a document belongs. Let a document d contain n distinct words $w_1, w_2, \ldots, w_n$. For every $c \in \mathscr{C}$, we define

$$p(d|c) = \prod_{i=1}^{n} p(w_i|c). \tag{4.1}$$

This definition of the class conditional probability of a document d uses the assumption that each word w is conditionally independent, given the class c. This is called the *naive Bayes* assumption. Then by using the Bayes Theorem we have

$$p(c|d) = \frac{p(c)p(d|c)}{\sum\limits_{c \in \mathscr{C}} p(d|c)p(c)}. \tag{4.2}$$

Finally given the document d the class c^* of d is predicted using the *maximum a posteriori* (MAP) estimate, i.e.,

$$c^* = \arg\max_{c \in \mathscr{C}} p(c|d). \tag{4.3}$$

The probabilities $p(c)$ and $p(w|c)$ are generally not known, and hence they are estimated from the training set $\mathcal{L}$. The estimate of $p(c)$, denoted by $\hat{p}(c)$ is obtained by

$$\hat{p}(c) = \frac{N_c}{N},$$

where N is the total number of documents in the training set $\mathcal{L}$, and N_c is the number of documents in $\mathcal{L}$ which belongs to the class c. The conditional probabilities $p(w|c)$ is estimated as

$$\hat{p}(w|c) = \frac{T(c,w)}{\sum\limits_{w' \in \mathsf{Voc}} T(c,w')},$$

where Voc is the vocabulary, i.e., the set of all possible words and $T(c, w)$ is the number of occurrences of the word w in training documents from class c. Note the denominator is the number of occurrences of all the words in the vocabulary for a class c. These estimates can be plugged in Eq. (4.3) to predict the class of a document.

k-Nearest Neighbor (kNN) Classifier. In this model, we use a different approach to represent a document. Let there be N words $w_1, w_2, \ldots, w_N$ in the vocabulary V, we represent a document d as a vector $\vec{V}(d)$ of N dimensions, where the i^{th} component is the number of times the word w_i occurs in the document. Thus we can see a set of documents as a set of vectors in a vector space. Hence if we take the vector representation of those documents, which belong to the same class, we will find that they are neighbors in a region of the vector space. Thus we can divide the vector space in regions, where each region corresponds to a class.

To measure the *similarity* between two documents d_1 and d_2, we compute the *cosine similarity* of their respective vector representations $\vec{V}(d_1)$ and $\vec{V}(d_2)$. The cosine similarity is defined as $\text{sim}(d_1, d_2) = \frac{\vec{V}(d_1) \cdot \vec{V}(d_2)}{||\vec{V}(d_1)|| ||\vec{V}(d_2)||}$ where $\vec{V}(d_1) \cdot \vec{V}(d_2)$ denotes the dot product of $\vec{V}(d_1)$ and $\vec{V}(d_2)$, and $||\vec{V}(d)||$ is the Euclidean norm. We must remember that the dot product between two vectors $\vec{x}$ and $\vec{y}$ is defined as $\vec{x} \cdot \vec{y} = \sum_{i=1}^{m} x_i y_i$ and the Euclidean norm of a vector $\vec{x}$ is defined as $\sqrt{\sum_{i=1}^{m} x_i^2}$.

To classify a new document d, using kNN algorithm, we proceed as follows. Once that we have the vector representation of d, denoted as $\vec{V}(d)$, we compute the similarity of $\vec{V}(d)$ with vector representations of all documents in $\mathcal{L}$, using the cosine similarity. Then we select the k nearest neighbors of $\vec{V}(d)$ in $\mathcal{L}$ and assign document d the majority class of its k closest neighbors. Generally k is a user chosen parameter.

4.2 Chaffing and Winnowing Text in a Natural Language

Let Voc be the vocabulary of any language, i.e. a set of words in any language. We consider a message M to be a finite sequence of words in Voc. We will denote a message by $M = \langle w_1, w_2, \ldots, w_m \rangle$, where $w_1, \cdots, w_m \in$ Voc and are not necessarily distinct. By length of a message we will mean the number of words in it and denote it by $\ell(M)$. By $M[i]$ we will denote the i^{th} word in M, where $1 \leq i \leq \ell(M)$. A sub-sequence of M is the message Z with zero or more words left out from M. Formally, if $M = \langle w_1, w_2, \ldots, w_m \rangle$, be a message then $Z = \langle z_1, z_2, \ldots, z_k \rangle$ is a subsequence of M if there exists a strictly increasing sequence $i_1, i_2, \ldots, i_k$ of indices of words in M such that for all $1 \leq j \leq k$, $w_{i_j} = z_j$. We will write $Z \trianglelefteq M$ to denote that Z is a subsequence of M, and if $Z \trianglelefteq$

```
$\mathbf{\Psi}_1.\mathsf{Chaff}_K(M)$
1. $M' \leftarrow \mathsf{AddChaff}(M, n)$ ;
2. $S \leftarrow \mathsf{Sub}(M', M)$ ;
3. for $i = 1$ to $\ell(M')$;
4.     if $i \in S$,
5.         $\mathsf{idx}[i] \leftarrow \mathsf{MAC}_K(\mathsf{bin}(M'[i]))$;
6.     else $\mathsf{idx}[i] \xleftarrow{\$} \{0,1\}^{\tau}$ ;
7.     end if
8. end for
9. return $(M', \mathsf{idx})$;
```

```
$\mathbf{\Psi}_1.\mathsf{Winnow}_K(M', \mathsf{idx})$
1. $j \leftarrow 1$ ;
2. for $i = 1$ to $\ell(M')$,
3.     if $\mathsf{MAC}_K(\mathsf{bin}(M'[i])) = \mathsf{idx}[i]$
4.         $M[j] \leftarrow M'[i]$;
5.         $j \leftarrow j + 1$;
6.     end if
7. end for
8. return $M$
```

Figure 4.1: $\mathbf{\Psi}_1$: Chaffing procedure proposed by Rivest.

M, then $\mathsf{Sub}(M, Z)$ will denote the set of indices of the words in M which forms the sequence Z. We give an example to illustrate these concepts.

Example 4.1. *Consider the following messages*

$$\begin{aligned} M'' &= \langle \texttt{books}, \texttt{dogs}, \texttt{rice}, \texttt{are}, \texttt{usually}, \texttt{useful}, \texttt{bark} \rangle \\ M' &= \langle \texttt{books}, \texttt{are}, \texttt{useful} \rangle \\ M &= \langle \texttt{dogs}, \texttt{usually}, \texttt{bark} \rangle \end{aligned}$$

If we fix an order of the words in M'', then $\ell(M'') = 7$ and $\ell(M') = \ell(M) = 3$. Both M and M' are subsequences of M'', i.e., $M' \trianglelefteq M''$ and $M \trianglelefteq M''$. Moreover, $\mathsf{Sub}(M'', M') = \{1, 4, 6\}$ *and* $\mathsf{Sub}(M'', M) = \{2, 5, 7\}$.

Formally a CW procedure is a tuple of algorithms $\mathbf{\Psi} = (\mathsf{Gen}, \mathsf{Chaff}, \mathsf{Winnow})$. Gen is the key generation algorithm, which generates a key from the finite key space $\mathcal{K}$. We will always assume that Gen outputs a uniform random element in $\mathcal{K}$. The procedure Chaff takes in a key generated by Gen and a message $M \in \mathsf{Voc}^*$ and outputs a pair $(M', \mathsf{idx}) \in \mathsf{Voc}^* \times \{0,1\}^*$, such that M is a subsequence of M'. We call M' the chaffed message and idx as the index. The index idx helps the winnowing procedure to recover the original message from the chaffed one. The winnowing procedure Winnow undo the chaffing. It takes in a key $K \in \mathcal{K}$ and the pair $(M', \mathsf{idx}) \in \mathsf{Voc}^* \times \{0,1\}^*$, and outputs $M \in \mathsf{Voc}^*$.

For a given CW procedure $\mathbf{\Psi} = (\mathsf{Gen}, \mathsf{Chaff}, \mathsf{Winnow})$ we define the error of $\mathbf{\Psi}$ as

$$\mathsf{err}_{\mathbf{\Psi}} = 1 - \Pr[K \xleftarrow{\$} \mathcal{K} : \mathsf{Winnow}_K(\mathsf{Chaff}_K(M)) = M]. \tag{4.4}$$

Thus for a CW procedure $\mathbf{\Psi}$ to be useful it is required that $\mathsf{err}_{\mathbf{\Psi}}$ is small.

We describe two CW procedures $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ in Figures 4.1 and 4.2. The description in the figures does not specify the function AddChaff. In both cases, for $n > \ell(M)$, $\mathsf{AddChaff}(M, n)$ expands M by adding random words to it in random positions and creates M', such that $\ell(M') = n$ and $M \trianglelefteq M'$. We will discuss specific considerations in the design of AddChaff later in Section 4.2.1.

$\mathbf{\Psi}_1$ is basically the method proposed by Rivest [65], when applied to the present situation. Besides adding random words to the message, the chaffing procedure in $\mathbf{\Psi}_1$ creates the index idx. The index contains $n = \ell(M')$ entries, and each entry $\mathsf{idx}[i]$ is a τ bit string, where τ is the output size of the message authentication code MAC. If $i \in \mathsf{Sub}(M', M)$ then $\mathsf{idx}[i]$ contains the message authentication code corresponding to the word $M'[i]$, otherwise $\mathsf{idx}[i]$ is a uniform random τ bit string. The corresponding winnowing procedure computes the tags for each word in M' and matches them with the tags in idx. If the tag for a specific word in M' matches, then that word is retained, otherwise it is discarded.

In Figure 4.2 we propose a new CW procedure. $\mathbf{\Psi}_2$ is different from $\mathbf{\Psi}_1$ by the fact that it uses an encryption scheme $\mathbf{E}_K$ instead of the message authentication code MAC. $\mathbf{\Psi}_2$ creates a string L which acts as an index for the returned message M'. We denote the i^{th} bit of L by L_i. $L_i = 1$ signifies that the word $M'[i]$ is a real word of the message and $L_i = 0$ means that $M'[i]$ is a fake word. The encryption of L along with M' is returned by $\mathbf{\Psi}_2.\mathsf{Chaff}_\mathsf{K}$. The corresponding winnowing procedure can use the key to obtain L, and using the information in L can remove the chaff from M' to obtain M.

There are some interesting differences in terms of both functionality and efficiency between the two CW procedures $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$, we discuss some relevant issues next. For the discussion we assume that the original message M contains m words, and α additional words are added to obtain M', i.e, $\ell(M') = n = m + \alpha$.

1. **Efficiency:** Irrespective of the AddChaff procedure, in $\mathbf{\Psi}_1$, n message authentication codes are required to be computed, whereas in $\mathbf{\Psi}_2$ a message of length n is to be encrypted. The exact efficiency comparison between these two options would depend on the specific choice of the message authentication code and the encryption algorithm, but for any practical choice, $\mathbf{\Psi}_2$ would be more efficient than $\mathbf{\Psi}_1$.

2. **Length Expansion:** The length expansion in $\mathbf{\Psi}_1$ would depend on the size of the tag produced by the message authentication code. If the MAC algorithm produces tags of τ bits then the size of idx would be $n\tau$ bits, where as in case of $\mathbf{\Psi}_2$ the length of idx would be just n bits.

```
$\mathbf{\Psi}_2$.Chaff$_K(M)$
1. $M' \leftarrow$ AddChaff$(M)$ ;
2. $S \leftarrow$ Sub$(M', M)$ ;
3. **for** $i = 1$ to $\ell(M')$;
4.     **if** $i \in S$,
5.         $L_i \leftarrow 1$;
6.     **else** $L_i \leftarrow 0$ ;
7.     **end if**
8. **end for**
9. idx $\leftarrow \mathbf{E}_K(L)$
10**return** $(M', \text{idx})$;
```

```
$\mathbf{\Psi}_2$.Winnow$_K(M', \text{idx})$
1. $L \leftarrow \mathbf{E}_K^{-1}(\text{idx})$ ;
2. $j \leftarrow 1$;
3. **for** $i = 1$ to $\ell(M')$,
4.     **if** $L_i = 1$
5.         $M[j] \leftarrow M'[i]$;
6.         $j \leftarrow j + 1$;
7.     **end if**
8. **end for**
9. **return** $M$
```

Figure 4.2: $\mathbf{\Psi}_2$: A new chaffing procedure using encryption.

3. **Error in Winnowing:** Assuming that the MAC is a random function, for $\mathbf{\Psi}_1$ we will have $\mathsf{err}_{\mathbf{\Psi}_1} \leq n/2^{\tau}$. But, in case of $\mathbf{\Psi}_2$, $\mathsf{err}_{\mathbf{\Psi}_2} = 0$.

4.2.1 Realizing AddChaff

Our main goal is to design the chaffing procedure in such a manner that it can fool a classifier. While designing AddChaff we use some tools and techniques for classifier design.

We assume that Voc is a finite set of words and the message space $\mathcal{M} \subseteq \text{Voc}^*$. Let $\mathcal{P} = \{1, 2, \ldots, k\}$ be the set of class labels or profiles, and let $f : \mathcal{M} \rightarrow \mathcal{P}$, be the profiling function or the classifier which induces a k-partition on the message space $\mathcal{M}$. In other words we can say $\mathcal{M} = \cup_{i=1}^{k} \mathcal{M}_i$, where $\mathcal{M}_i \cap \mathcal{M}_j = \emptyset$ for all $i \neq j$, and $x \in \mathcal{M}_i$, iff $f(x) = i$.

In all practical classification tasks, the real partition of the message space (i.e., the function f) is generally not known. What is known is the class labels of a finite number of messages in $\mathcal{M}$. Let COR $\subset \mathcal{M}$, be the set of messages in $\mathcal{M}$ whose class labels are known, and we define $\mathcal{L} = \{(x, f(x)) : x \in \text{COR}\}$. The set COR is called the corpus and $\mathcal{L}$ the training set.

A classifier is generally designed using $\mathcal{L}$, and we call such a classifier as $f_{\mathcal{L}}^*$. Note, that $f_{\mathcal{L}}^*$ is not the actual function f, but a "good" approximation of f. We will assume that a user knows $\mathcal{L}$, and has access to a "good" classifier $f^*_{\mathcal{L}}$ designed by some technique which is not of interest to us.

Now, we introduce some additional notations, which will help in the exposition that follows. For $i \in \mathcal{P}$, let $\mathscr{C}_i = \{x \in \mathsf{COR} : f(x) = i\}$, i.e., $\mathscr{C}_i$ contains the messages in COR which belongs to class i. For $w \in \mathsf{Voc}$ and $x \in \mathsf{COR}$ define $\mathtt{cnt}_x(w)$ as the number of times the word w occurs in the message x. For each $w \in \mathsf{Voc}$ and $i \in \mathcal{P}$, define

$$p(w,i) = \frac{\sum\limits_{x \in \mathscr{C}_i} \mathsf{cnt}_x(w)}{\sum\limits_{x \in \mathscr{C}_i} \ell(x)} \tag{4.5}$$

$$p'(w,i) = \frac{\sum\limits_{x \in \mathsf{COR} \setminus \mathscr{C}_i} \mathsf{cnt}_x(w)}{\sum\limits_{x \in \mathsf{COR} \setminus \mathscr{C}_i} \ell(x)} \tag{4.6}$$

$$\mathsf{OR}(w,i) = p(w,i) \log \frac{p(w,i)}{p'(w,i)}. \tag{4.7}$$

The quantity $\mathsf{OR}(w,i)$ is called the *odds ratio* of the word w in class i. This has been used as a measure of how important the word w is for class i, thus if $\mathsf{OR}(w,i)$ is big it means that a message which has high occurrence of the word w has a high "likelihood" of belonging to class i. In the literature [52] there are several variants of the definition of odds ratio but they are all small variants of the definition that we describe in Eq. (4.7).

Based on the corpus COR, and the odds ratio, we want to find the most significant words in each class $i \in \mathcal{P}$. Let $\mathbb{W}_i$ be the set containing the distinct words in the messages in $\mathscr{C}_i$, and $\mathbb{W} = \cup_{i=1}^k \mathbb{W}_i$. For $i \in \mathcal{P}$ and $p \leq |\mathbb{W}_i|$, let Sig_i^p be the p words in $\mathbb{W}_i$ with the highest $\mathsf{OR}(\cdot, i)$ values. With these definitions and notations we are now ready to specify some procedures for AddChaff.

The heart of each method contains the procedure $\mathsf{Basic}(M, S, n, \mathtt{prob})$, which is described in Figure 4.3. The procedure Basic takes as input a message M, a finite set $S \subset \mathbb{W}$, a positive integer $n > \ell(M)$ and a discrete probability distribution prob on the set S.

Basic returns a chaffed message M of size n. The procedure involves in expanding M by inserting $n - \ell(M)$ words in random positions, and each of these new words is drawn from the given set S following the given probability distribution prob. Various procedures of adding chaff can be derived using the procedure Basic by varying the set S and the probability distribution prob. We describe some interesting variants in Figure 4.4.

Our first chaffing procedure AddChaff1, is perhaps the most naive way to corrupt a

```
Basic($M, n, S,$ prob)
1.  $n' \leftarrow n - \ell(M)$ ;
2.  $M'$ be a sequence of length $n$;
3.  for $i = 1$ to $\ell(M')$,
4.      $M'[i] \leftarrow$ NULL
5.  end for
6.  $I$ be a set of $n'$ elements sampled uniformly
    without replacement from $\{1, \ldots, n\}$;
7.  for each $i \in I$
8.      Sample a word $w$ from $S$
        according to the distribution prob;
9.      $M'[i] \leftarrow w$;
10. end for
11. $j \leftarrow 1$;
12. for $i \leftarrow 1$ to $\ell(M)$,
13.     while $M'[j] \neq$ NULL,
14.         $j \leftarrow j + 1$;
15.         $M'[j] \leftarrow M[i]$;
16. end for
17. return $M'$
```

Figure 4.3: The basic procedure for adding chaff

message M. Here we take S as $\mathbb{W}$, the set of all the words in the k different classes. For the rest of the procedures we assume that we have a classifier $f^*_{\mathcal{L}}$ and we use it to classify a message M. Then we choose a word uniformly at random from set S.

In procedure AddChaff2, first we classify the message M, using $f^*_{\mathcal{L}}$, let us call c the class of M. Then we construct the set S as the union of the significant words in other classes c' different from c. Finally we set prob as the uniform distribution on S.

In procedure AddChaff3, just as before we compute c by using $f^*_{\mathcal{L}}$. In the second step we choose a class $j \neq c$ uniformly at random from $\mathcal{P}$. In set S we put the most significant words in class j. Here we also set prob as the uniform distribution on S.

Procedure AddChaff4, is the same as AddChaff3, the only difference is the way we set the probability distribution prob. For $s \in S$ set $\texttt{prob}(s)$ as $\frac{\mathsf{OR}(s,j)}{\sum_{w \in S} \mathsf{OR}(w,j)}$. Note that $\texttt{prob}(s)$ is a discrete probability distribution on S as $\sum\limits_{s \in S} \texttt{prob}(s) = 1$ and for all $s \in S$, $0 \leq \texttt{prob}(s) \leq$

```
AddChaff1$(M, n)$
1. $S \leftarrow \mathbb{W}$;
2. for each $s \in S$,
3.     prob$(s) \leftarrow 1/|S|$;
4. end for
5. $M' \leftarrow$ Basic$(M, n, S,$ prob$)$;
6. return $M'$;
```

```
AddChaff2$(M, n)$
1. $c \leftarrow f^*_{\mathcal{L}}(M)$;
2. $S \leftarrow \bigcup_{i \in \mathcal{P} \setminus \{c\}} \mathsf{Sig}^p_i$;
3. for each $s \in S$,
4.     prob$(s) \leftarrow 1/|S|$;
5. end for
6. $M' \leftarrow$ Basic$(M, n, S,$ prob$)$;
7. return $M'$;
```

```
AddChaff3$(M, n)$
1. $c \leftarrow f^*_{\mathcal{L}}(M)$;
2. $j \xleftarrow{\$} \mathcal{P} \setminus \{c\}$;
3. $S \leftarrow \mathsf{Sig}^p_j$;
4. for each $s \in S$,
5.     prob$(s) \leftarrow 1/|S|$;
6. end for
7. $M' \leftarrow$ Basic$(M, n, S,$ prob$)$;
8. return $M'$;
```

```
AddChaff4$(M, n)$
1. $c \leftarrow f^*_{\mathcal{L}}(M)$;
2. $j \xleftarrow{\$} \mathcal{P} \setminus \{c\}$;
3. $S \leftarrow \mathsf{Sig}^p_j$;
4. for each $s \in S$,
5.     prob$(s) \leftarrow \frac{\mathsf{OR}(s,j)}{\sum_{w \in S} \mathsf{OR}(w,j)}$;
6. end for
7. $M' \leftarrow$ Basic$(M, n, S,$ prob$)$;
8. return $M'$;
```

Figure 4.4: Different chaffing procedures.

1. Moreover a word $s \in S$ will have more probability assigned to it if its odds ratio in belonging to class c is high. Thus this distribution when used to add chaff would guarantee that more significant words in class j are added more frequently as chaff.

Both AddChaff3 and AddChaff4 try to add chaff in a directed manner, i.e, they try to add words specific to a class different from the class where the message really belongs. These two procedures can also be modified by allowing the user to choose a class instead of randomly selecting a class. This may allow the user to try to add chaff in such a way that the message gets classified to a class of his choice. We present some experiments to evaluate the performance of these procedures on a real corpus in Section 4.4. In the next section we provide a preliminary analysis of $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$.

4.3 Security Analysis

Here we provide a preliminary analysis of the schemes which we already described.

Theorem 4.1. *$\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are not IND-CPA secure.*

Proof. Let $\mathsf{LongSubSeq}(A, B)$ be the longest common subsequence of two messages A and B. Note that we consider both A and B as a sequence of words. If $n = \max\{\ell(A), \ell(B)\}$, then $\mathsf{LongSubSeq}(A, B)$ can be found in $O(n^3)$ time by using dynamic programming [19].

Consider an adversary who chooses two messages M_0 and M_1, such that $\ell(M_0) = \ell(M_1)$ and M_0, M_1 contains distinct words. The adversary submits these two messages to the challenger. The challenger returns $C = \mathbf{\Psi}_1.\mathsf{Chaff}_K(M_b)$ where $b \overset{\$}{\leftarrow} \{0, 1\}$ and $K \overset{\$}{\leftarrow} \mathcal{K}$, and b and K are unknown to the adversary. The goal of the adversary is to predict the bit b, i.e., to guess whether the ciphertext C corresponds to the message M_0 or M_1.

The adversary parses C as (idx, M'), and if $\mathsf{LongSubSeq}(M_0, M') = M_0$, the adversary outputs 0 otherwise it outputs a 1.

If C corresponds to the message M_0 then the adversary is successful with probability 1. On the other hand, if C corresponds to M_1 then the adversary outputs a zero only if M_0 is the longest common subsequence of M' and M_0, which can only happen if M_0 is a subsequence of the chaffed sequence of words. The exact probability of this happening would depend on the details of the AddChaff procedure, but in any case this probability would be sufficiently low.

The same argument holds for the procedure $\mathbf{\Psi}_2$. □

We want to prove that the schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are PROF-EAV secure. To do this we need an additional assumption on the AddChaff procedure.

Definition 4.1. *A scheme for adding chaff* AddChaff *is called $(1 - \epsilon)$-fooling, if for all classifiers f^**

$$\Pr[M \overset{\$}{\leftarrow} \mathcal{M} : f^*(\mathsf{AddChaff}(M)) = f(M)] \leq \epsilon$$

The probability is taken over the random choice of M and the randomness in AddChaff.

Assuming that the procedure AddChaff is $(1 - \epsilon)$-fooling we can prove some interesting properties of $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ as follows.

Theorem 4.2. *In $\mathbf{\Psi}_1$, if* AddChaff *is $(1-\epsilon)$-fooling and* $\mathsf{Mac}_K()$ *is a pseudorandom function, then $\mathbf{\Psi}_1$ is* PROF-EAV *secure. In other words, let $\mathcal{A}$ be a* PROF-EAV *adversary attacking*

```
Adversary $\mathcal{B}^{\mathcal{O}}$
1. $M \xleftarrow{\$} \mathcal{M}$;
2. $M' \leftarrow \mathsf{AddChaff}(M, n)$ ;
3. $S \leftarrow \mathsf{Sub}(M', M)$ ;
4. for $i = 1$ to $\ell(M')$;
5.     if $i \in S$,
6.         $\mathsf{idx}[i] \leftarrow \mathcal{O}(M'[i])$;
7.     else $\mathsf{idx}[i] \xleftarrow{\$} \{0,1\}^{\tau}$ ;
8.     end if
9. end for
10. Send $(M', \mathsf{idx})$ to $\mathcal{A}$
11.$\mathcal{A}$ returns $j$
12.if $j = f(M)$ return 1;
13.else return 0;
```

Figure 4.5: Adversary $\mathcal{B}$ used for the proof of Theorem 4.2

$\mathbf{\Psi}_1$*, then there exists a* PRF *adversary* $\mathcal{B}$*, such that*

$$\mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_1,f}(\mathcal{A}) \leq \mathbf{Adv}^{\text{prf}}_{\mathsf{MAC}}(\mathcal{B}) + \epsilon.$$

Proof. Let $\mathcal{A}$ be an arbitrary adversary attacking the protocol $\mathbf{\Psi}_1$. We construct a PRF adversary $\mathcal{B}$ attacking the $\mathsf{Mac}_K()$ as shown in Figure 4.5.

Note that $\mathcal{B}$ being a PRF adversary has as its oracle $\mathcal{O}$ a random function or $\mathsf{Mac}_K()$, where $K \xleftarrow{\$} \mathcal{K}$, and $\mathcal{B}$ is required to distinguish between these two situations. Moreover, it is clear from the description of $\mathcal{B}$, that if its oracle is $\mathsf{Mac}_K()$ then it behaves like the procedure $\mathbf{\Psi}_1$ as described in Figure 4.1, thus

$$\begin{aligned}
p_1 &= \Pr[K \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{MAC}_K()} \Rightarrow 1] \\
&= \Pr[M \xleftarrow{\$} \mathcal{M}, K \xleftarrow{\$} \mathcal{K} : \mathcal{A}(\mathbf{\Psi}_1.\mathsf{Chaff}_K(M)) \Rightarrow f(M)] \\
&\geq \Pr[M \xleftarrow{\$} \mathcal{M}, K \xleftarrow{\$} \mathcal{K} : \mathcal{A}(\mathbf{\Psi}_1.\mathsf{Chaff}_K(M)) \Rightarrow f(M)] - \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)] \\
&\geq \mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_1,f}(\mathcal{A}). \qquad (4.8)
\end{aligned}$$

On the other hand, if the oracle $\mathcal{O}$ is a random function in $\mathsf{Func}(\mathcal{M}, \{0,1\}^\tau)$ then

$$\begin{aligned} p_2 &= \Pr[\rho \xleftarrow{\$} \mathsf{Func}(\mathcal{M}, \{0,1\}^\tau) : \mathcal{B}^{\rho()} \Rightarrow 1] && (4.9) \\ &= \Pr[M \xleftarrow{\$} \mathcal{M} : \mathcal{A}(\mathsf{AddChaff}(M)) \Rightarrow f(M)] && (4.10) \\ &\le \epsilon. && (4.11) \end{aligned}$$

The transition from Eq. (4.9) to Eq. (4.10) is due to the fact that, when the oracle of $\mathcal{B}$ is a random function then the whole idx is a random string which has no relationship with the message. Hence the idx obtained by $\mathcal{A}$ is of no use, hence for $\mathcal{A}$ it is same as interacting with just $\mathsf{AddChaff}$. The transition from Eq. (4.10) to Eq. (4.11) is due to the the assumption that $\mathsf{AddChaff}$ is $(1-\epsilon)$-fooling.

Now, using the definition of the PRF advantage of $\mathcal{B}$ and the equations (4.8) and (4.11), we have

$$\begin{aligned} \mathbf{Adv}^{\text{prf}}_{\mathsf{MAC}}(\mathcal{B}) &= p_1 - p_2 \\ &\ge \mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_1, f}(\mathcal{A}) - \epsilon. \end{aligned}$$

Hence the Theorem follows. □

Theorem 4.3. *In $\mathbf{\Psi}_2$, if* $\mathsf{AddChaff}$ *is $(1-\epsilon)$-fooling and $\mathbf{E}_K()$ is* IND-EAV *secure, then $\mathbf{\Psi}_2$ is* PROF-EAV *secure. In other words if $\mathcal{A}$ is an arbitrary* PROF-EAV *adversary attacking $\mathbf{\Psi}_2$, then there exists an* IND-EAV *adversary $\mathcal{B}$, such that*

$$\mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_2, f}(\mathcal{A}) \le 2\mathbf{Adv}^{\text{ind-eav}}_{\mathbf{E}}(\mathcal{B}) + \epsilon.$$

Proof. The proof is similar to the proof of Theorem 4.2 above. Let $\mathcal{A}$ be an arbitrary adversary attacking the protocol $\mathbf{\Psi}_2$. We construct an adversary $\mathcal{B}$ attacking the $\mathbf{E}_K$ in the IND-EAV sense, as shown in Figure 4.6.

The adversary $\mathcal{B}$ selects a message M randomly from $\mathcal{M}$ and adds chaff to it. It then creates two indices L_0 and L_1, where L_0 is the real index for the message M as in $\mathbf{\Psi}_2$ and L_1 is a random string of same length as L_0. $\mathcal{B}$ submits L_0 and L_1 to its challenger and receives idx as a response. Note that according to the IND-EAV game, the challenger selects a bit b uniformly at random, and $\mathsf{idx} = \mathbf{E}_K(L_b)$. After receiving idx from the challenger, $\mathcal{B}$ sends (M', idx) to $\mathcal{A}$. $\mathcal{A}$ guesses the profile of the message M as j. If j corresponds to the real profile of M then $\mathcal{B}$ outputs a 1 otherwise $\mathcal{B}$ outputs a 0.

```
Adversary $\mathcal{B}$
1. $M \xleftarrow{\$} \mathcal{M}$;
2. $M' \leftarrow \mathsf{AddChaff}(M)$;
3. $S \leftarrow \mathsf{Sub}(M', M)$ ;
4. for $i = 1$ to $\ell(M')$;
5.     if $i \in S$,
6.         $L_0[i] \leftarrow 1$;
7.     else $L_0[i] \leftarrow 0$ ;
8.     end if
9. end for
10. $L_1 \xleftarrow{\$} \{0,1\}^{\ell(M')}$;
11. Send $(L_0, L_1)$ to the challenger.
12. The challenger returns idx.
13. Send $(M', \mathsf{idx})$ to $\mathcal{A}$;
14. $\mathcal{A}$ returns $j$;
15. if $j = f(M)$ return 1;
16. else return 0;
```

Figure 4.6: The adversary $\mathcal{B}$ used in the proof of Theorem 4.3

Hence, we have,

$$\begin{aligned}\Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1] &= \Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1|b=0]\Pr[b=0] + \\ &\quad \Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1|b \neq 0]\Pr[b \neq 0] \quad (4.12)\\ &= \frac{1}{2}(p_0 + p_1), \quad (4.13)\end{aligned}$$

where $p_0 = \Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1|b=0]$ and $p_1 = \Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1|b=1]$. If the encrypted message was L_0 then $\mathcal{A}$ gets the pair (M', idx), which is a valid output of $\mathbf{\Psi}_2.\mathsf{Chaff}_K$, thus we have

$$p_0 = \Pr[M \xleftarrow{\$} \mathcal{M}, K \xleftarrow{\$} \mathcal{K} : \mathcal{A}(\mathbf{\Psi}_2.\mathsf{Chaff}_K(M)) \Rightarrow f(M)] \quad (4.14)$$

Otherwise, i.e., if $b = 1$, then idx has no relation with the chaffed message M', and it is of no use to $\mathcal{A}$. Hence,

$$\begin{aligned} p_1 &= \Pr[M \xleftarrow{\$} \mathcal{M} : \mathcal{A}(\mathsf{AddChaff}(M)) \Rightarrow f(M)] \\ &\leq \epsilon. \end{aligned} \tag{4.15}$$

The last inequality is due to the fact that AddChaff is $(1-\epsilon)$-fooling.

Now let,

$$p_2 = \max_{\mathcal{A}'} \Pr[\mathcal{A}'(.) \Rightarrow f(x)]. \tag{4.16}$$

Thus using the definition of PROF-EAV security of of $\mathbf{\Psi}_2$, and Eq. (4.14), we have

$$\mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_2,f}(\mathcal{A}) = p_0 - p_2. \tag{4.17}$$

Finally, using the definition of IND-EAV advantage of $\mathcal{B}$ and equations (4.13), (4.14), (4.15) and (4.17) we have

$$\begin{aligned} \mathbf{Adv}^{\text{ind-eav}}_{\mathbf{E}}(\mathcal{B}) &= \frac{1}{2} - \Pr[\text{Exp-IND-EAV}^{\mathcal{B}} \Rightarrow 1] \\ &= \frac{1}{2} - \frac{1}{2}(p_0 + p_1) \\ &\geq \frac{p_2}{2} - \frac{1}{2}(p_0 + p_1) \\ &= \frac{1}{2}(p_2 - p_0) - \frac{p_1}{2} \\ &\geq \frac{1}{2}\mathbf{Adv}^{\text{prof-eav}}_{\mathbf{\Psi}_2,f}(\mathcal{A}) - \frac{\epsilon}{2} \end{aligned}$$

as desired. □

Remark. The validity of Theorems 4.3 and 4.2 highly depends on AddChaff being $(1-\epsilon)$-fooling. This is a strong and not a well studied assumption. Some preliminary experiments presented in Section 4.4, suggest that the AddChaff procedures described in Section 4.2 do fool some classifiers, but still they are not optimal, as we will see, still there are a small percentage of messages which are correctly classified. Although Theorems 4.3, 4.2 are mathematically correct, we must notice that we are considering an encryption scheme $\mathbf{E}_K$ which is IND-EAV secure. We believe that this level of security is still very strong if we are considering a profiling adversary.

4.4 Some Experiments

In this section we present some experiments to test the effectiveness of our proposed methods for adding chaff. For the experiments we use a tool called *Rainbow* which is provided within a library called *Bow (or libbow)*. This library was created by Andrew McCallum [51], it was intended for writing statistical text-processing programs. In particular we use libbow 20020213-10 in amd64, which is a distribution for Ubuntu. Rainbow is a program that does document classification. We use two classification methods, provided within this library: Naive-Bayes and K-nearest neighbor.

Also we use the *20_newsgroup* data set, which is a collection of 20,000 newsgroup documents, partitioned in 20 different classes.

We proceeded as follows. We used 80% of the documents in the corpus as the training set and used the remaining 20% for testing. Using the training set, we estimated the required probabilities for the Naive-Bayes classifier, and thus we designed a classifier for the 20_newsgroup data, further we will call this classifier as f_{nb}^{ng}. We also designed a kNN classifier with the same training set and we denote this classifier as f_{knn}^{ng}. From the test set we chose those documents which were correctly classified by our designed classifier f_{nb}^{ng}. There were 3269 documents in the test set which got correctly classified. We used only these documents for our experiments, for ease of reference, we will further denote this set of documents by $\mathbb{D}$.

The goal of the experiments was to test the performance of the AddChaff procedures depicted in Figure 4.4. The following protocol was followed for the experiments.

1. We added chaff to the documents in $\mathbb{D}$ using each of the procedures in Figure 4.4 and we tried to classify these transformed documents using f_{nb}^{ng} (an also f_{knn}^{ng}, in some cases), and measured the classification accuracy, which we computed as the percentage of the number of documents correctly classified. As each of the proposed AddChaff procedures are randomized, hence we repeated each experiment 100 times, and we report the mean, the minimum, maximum and the standard deviation of the classification accuracy over these 100 repetitions. Note that a low classification accuracy signifies high performance for the AddChaff procedure.

2. The AddChaff procedures takes in a parameter n, which represents the length of the output document. If n is bigger then relatively more chaff has been added, thus the performance of an AddChaff procedure would depend on n. We tried different values of n in our experiments. For a message M, let $n = (\lambda + 1)\ell(M)$, from this expression we can see that λ indicates how much increases the length of the original message. For most experiments, we report results by varying λ from 1 to 4. For example, a value of $\lambda = 1$ implies that we will add as many words as there are in the original text, i.e., the length of the ciphertext will be twice the length of the plaintext.

3. The procedures AddChaff2, AddChaff3 and AddChaff4 uses the set Sig_j^p, which de-

notes the p most significant words in class j. Thus, p is a user defined parameter. We report results for $p = 100, 500$.

4. The procedures AddChaff2, AddChaff3 and AddChaff4 also uses a classifier $f^*_{\mathcal{L}}$ in their description, we used f^{ng}_{nb} as the classifier in all cases.

We report the results in Figure 4.7, we discuss and analyze these results below.

1. Table 4.7(a) reports the classification accuracy when the documents in $\mathbb{D}$ were corrupted using AddChaff1, and classified using f^{ng}_{nb}. The results show a high classification accuracy. As expected, if we increase λ, then the accuracy goes down, but even with $\lambda = 4$ the classification accuracy is above 60%, which is not in general acceptable. This shows that AddChaff1 is probably not a good method to add chaff.

2. Tables 4.7(b) and 4.7(c) show the results when AddChaff2 was used to add chaff and the classifier used was f^{ng}_{nb}. As AddChaff2 uses the set Sig^p_j, thus we used two values of p for our experiments. Tables 4.7(b) shows the results for $p = 100$ and 4.7(c) shows the same when $p = 500$. The results are much better than in case of AddChaff1, for $\lambda = 4$, a classification accuracy of around 16% is achieved which seems to be reasonable. The results are significantly better when $p = 100$ compared to when $p = 500$. An intuitive justification of this can be that higher the value of p, there are more words in to the set S, from which the words for chaffing are chosen, thus for a higher value of p the set S contains more words which are not "so important" for representing the specific class.

3. The results of using AddChaff3 are shown in the Tables 4.7(d), 4.7(e), 4.7(f) and 4.7(g).

   - Table 4.7(d) shows the results by using f^{ng}_{nb} and $p = 100$.
   - Table 4.7(e) shows the results by using f^{ng}_{nb} and $p = 500$.
   - Table 4.7(f) shows the results by using f^{ng}_{knn} and $p = 100$.
   - Finally, Table 4.7(g) shows the results by using f^{ng}_{nb} and $p = 500$.

   These results are quite encouraging and it shows that AddChaff3 does a good job in fooling a classifier. Guided by these encouraging results we also tried with $\lambda = 0.5$, i.e., when the quantity of added chaff is just half the size of the original document. With $\lambda = 0.5$ the results are not that good. It is also to be noted that when we use kNN as the classifier (Tables 4.7(f) and 4.7(g)) the performance degrades, as the chaff is added based on a Naive-Bayes classifier. But still the results with kNN classifier are within acceptable limits.

λ	**Mean**	**Min**	**Max**	**STD**
1	96.55	96.24	96.88,	0.12
2	93.41	92.99	93.94	0.18
3	84.29	83.60	85.01	0.27
4	67.76	66.96	68.8	0.33

(a)

λ	**Mean**	**Min**	**Max**	**STD**
1	84.01	83.27	84.92	0.34
2	52.92	51.61	54.24	0.51
3	29.85	28.79	30.87	0.42
4	16.54	15.57	17.43	0.40

(b)

λ	**Mean**	**Min**	**Max**	**STD**
1	89.53	88.89	90.30	0.29
2	66.69	65.52	67.85	0.50
3	44.40	43.22	45.46	0.47
4	28.39	27.13	29.37	0.47

(c)

λ	**Mean**	**Min**	**Max**	**STD**
1	0.36	0.21	0.55	0.07
2	0.12	0.12	0.21	0.01
3	0.12	0.12	0.15	0.0
4	0.12	0.12	0.12	0.0
0.5	9.06	8.14	9.79	0.37

(d)

λ	**Mean**	**Min**	**Max**	**STD**
1	0.97	0.67	1.28	0.12
2	0.13	0.12	0.21	0.02
3	0.12	0.12	0.12	0.0
4	0.12	0.12	0.12	0.0
0.5	24.29	23.19	25.70	0.53

(e)

λ	**Mean**	**Min**	**Max**	**STD**
1	9.48	8.66	10.31	0.36
2	5.17	4.28	5.97	0.33
3	3.47	2.88	4.22	0.25
4	2.65	2.05	3.09	0.23
0.5	16.69	13.91	21.35	2.33

(f)

λ	**Mean**	**Min**	**Max**	**STD**
1	18.29	17.16	19.36	0.41
2	12.65	11.56	13.95	0.41
3	9.48	8.53	10.43	0.40
4	4.40	0.92	8.41	3.16
0.5	20.73	19.76	21.78	0.38

(g)

λ	**Mean**	**Min**	**Max**	**STD**
1	0.36	0.21	0.52	0.07
2	0.13	0.12	0.15	0.01
3	0.12	0.12	0.12	0.00
4	0.12	0.12	0.12	0.00
0.5	10.00	9.15	11.20	0.35

(h)

λ	**Mean**	**Min**	**Max**	**STD**
1	7.86	7.13	8.84	0.29
2	3.40	2.60	3.88	0.25
3	1.96	1.56	2.45	0.21
4	1.24	0.92	1.65	0.15
0.5	15.55	12.97	21.35	2.98

(i)

Figure 4.7: a) Exp1:AddChaff1, Naive-Bayes. b) Exp2: AddChaff1, Naive-Bayes, $p = 100$. c) Exp2: AddChaff1, Naive-Bayes, $p = 500$. d) Exp3: AddChaff3, Naive-Bayes, $p = 100$. e) Exp3: AddChaff3, Naive-Bayes, $p = 500$. f) Exp3: AddChaff3, kNN, $p = 100$. g) Exp3: AddChaff3, kNN, $p = 500$. h) Exp4: AddChaff4, Naive-Bayes, $p = 500$. i) Exp4: AddChaff4, kNN, $p = 500$. A value of $\lambda = 0.5$ means that the length of the ciphertext is 1.5 the length of the plaintext.

4. Tables 4.7(h) and 4.7(i) shows the results of AddChaff4 with the Naive-Bayes and kNN classifiers respectively. In both cases $p = 500$ is used, with $p = 100$ marginally better results are obtained. These results show almost the same characteristics as the previous one, but they are marginally better in all cases.

Based on the reported results, we can summarize some of the characteristics and recommendations for the proposed methods to add chaff:

1. **Comparative performance:** AddChaff1 performs the worst and AddChaff4 performs the best. The performance of all schemes except AddChaff1 for reasonable values of λ seems to be acceptable.

2. **The role of λ:** The experiments support our intuition that a higher value of λ, gives better performance. But, with a higher value of λ there is more expansion in the message which would imply more bandwidth requirements for communication and more computational costs. Hence a proper tradeoff for the value of λ is needed. The experiments demonstrate that $\lambda > 2$ can be a suitable choice for AddChaff2 and AddChaff3.

3. **Role of p:** The smaller the value of p, better performance is obtained. But a very small value of p would affect the variability of the cipher, as only a few extra words would be repeatedly added as chaff, which may allow other techniques to remove the chaff. For reasonably sized messages, $p = 500$ seems to be a good choice in terms of both classification accuracy and ciphertext variability.

4.5 Final Remarks

We studied the CW scheme and based on this we proposed two constructions which are good candidates for an encryption scheme secure in the PROF-EAV sense. We would like to note certain specific characteristics of the constructions here.

1. The schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ are very different from existing encryption schemes, as here the message goes in clear hidden within the chaff. Thus, it provides a very weak form of security but their security seems to be enough against profilers.

2. We live in an era of large scale surveillance which is performed by corporations for their business goals, and even by government agencies for different purposes. Such surveillance compromises privacy of every individual. Government agencies

may be sometimes interested to know more about individuals who communicate through encrypted messages. If usual strong encryption is used for day to day communication (say through emails), it is easy to detect whether the messages are encrypted. In the proposed schemes it may be difficult to detect encryption, as the message consists of valid words in a natural language. This may find use in certain scenarios.

3. The experimental results show that the schemes $\mathbf{\Psi}_1$ and $\mathbf{\Psi}_2$ cannot fool a classifier $\%100$ of time. This should not be seen as a weakness, as if most of the messages sent by a user cannot be correctly classified then a profiling adversary would fail to build a meaningful profile. To see this, consider a user U, who always sends messages of a specific class c. If for example, only one of nine messages sent by U are classified by the adversary as belonging to c, and other then the adversary would not have enough information to profile U into a specific class.

4. One drawback of our scheme is that, it results in expansion of the messages. The experiments suggest that the best option is to use the procedure AddChaff4 for adding chaff. With this procedure, if the amount of chaff added is same as the size of the message, then a reasonable level of security can be achieved. Assuming an average email to be of size 75Kb, this would approximately make the chaffed message 150 Kb long. In addition to the chaffed message the ciphertext would include an index. The index size in case of $\mathbf{\Psi}_2$ is minimal. Recall, that if $\mathbf{\Psi}_2$ is used then the index only occupies k bits, where k is the number of words in the message. The number of words in a message of size 75Kb would be much less than 75Kb, thus the size of the index would be negligible compared to the message size. In addition, if the protocol $\mathbb{P}'$ is used, then according to our recommendation, five CAPTCHAs are required to be sent. Assuming the average size of each CAPTCHA to be 15Kb, then the size of the whole ciphertext would be 225 KB $(150 + 5 \times 15)$. Thus the length expansion is tolerable.

5. The theoretical analysis, that we provide in Section 4.3, shows that the security of $\mathbf{\Psi}_2$ depends on the IND-EAV security of the encryption algorithm $\mathbf{E}$. It would be more desirable to show the PROF-EAV security of $\mathbf{\Psi}_2$ using a less stringent security requirement on $\mathbf{E}$. We wish to take up this problem in the near future.

Chapter 5

A Formal Treatment of Tokenization

There are some scenarios where the messages that we want to encrypt have a small length. This is the case of credit card numbers. 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. If we wish to avoid this length expansion, then we must consider a cipher $E : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ where $\mathcal{M} = \{0,1\}^n$ where $n = 53$, this means that the size of our domain is 2^{53}, this size is so small that it is a difficult problem to find such an encryption scheme which also has a good security bound. This problem is known as the *small domain encryption problem*.

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 [8]. 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 [8, 10, 14, 36, 53, 75].

More recently, an alternative method to protect credit card numbers has been proposed. This method is called *tokenization*. As the name suggests it is implemented using *tokens*.

Tokens are numbers which replace credit card numbers but are unrelated to them. To our knowledge a formal cryptographic analysis of the problem has not been done till now. In this Chapter we study *tokenization* from a cryptographic viewpoint. We propose security notions according to different threat models and analyze the adequacy of these notions in practical scenarios. We also propose some constructions of tokenization systems and prove their security.

5.1 A Brief History

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) [58] 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 [71]:

> *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 [69, 70, 71, 78]. PCI SSC has also formulated its guidelines regarding *tokenization* [59]. Voltage Security [78] has provided a solution where a credit card number encrypted by a FPE scheme can act as a token. To the best of our knowledge, this is the only solution to the tokenization problem with known cryptographic guarantees. But to our knowledge, there does not exist a formal security model for tokenization. It has even 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 [71]. But it is not clear what basic cryptographic objects should be used and in what way, to achieve the goals of tokenization.

5.2 Tokenization Systems: Requirements and PCI DDS Guidelines

The basic architecture of a tokenization system is described in Fig 5.1. 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.

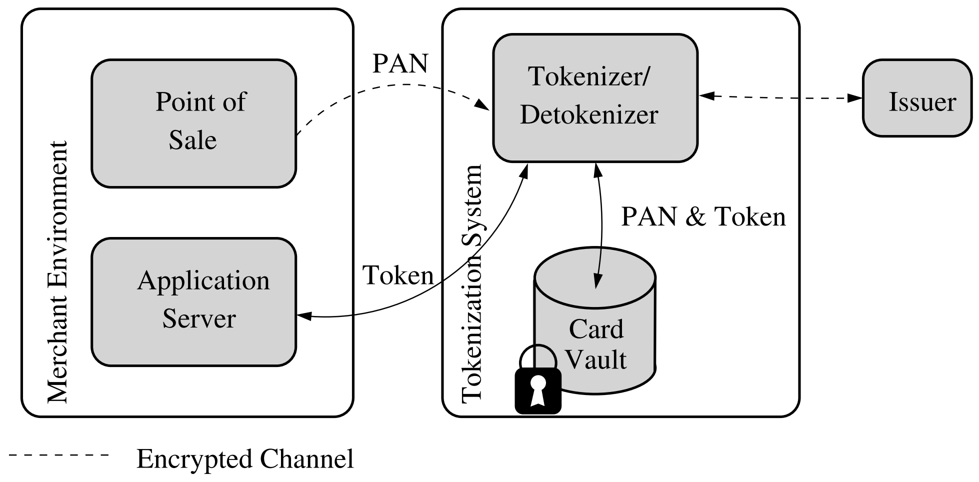


Figure 5.1: Architecture of the tokenization system

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 [59], 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 [59] 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 to 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 [70]), 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 [39] 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.

5.3 Additional Cryptographic Objects

In this Section we introduce two additional cryptographic objects, which we did not describe previously. These cryptographic objects, namely *tweakable enciphering schemes (TES)* and *deterministic encryption* that we describe here, are important components in our constructions.

TWEAKABLE ENCIPHERING SCHEME (TES). A tweakable enciphering scheme is a block cipher *mode of operation*. As we mentioned in Chapter 2, a block cipher let us encrypt messages which have a bit length smaller than block length of the block cipher. To solve this inconvenience, we use a mode of operation, which is a procedure that allows us to encrypt messages of arbitrary length. A TES is generally constructed using a block cipher as the basic primitive. It also use a special value, called a *tweak*. The purpose of this tweak is to break the determinism inherent in a block cipher. This mechanism is useful in different scenarios such as disk encryption. Now we give a formal definition of a TES.

Let $\mathcal{E}$ be 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 $\mathcal{E}(K, T, \cdot) = \mathcal{E}_K^T(\cdot)$ is a length preserving permutation. We define the $\widetilde{\text{prp}}$ advantage of an adversary $\mathcal{A}$ as

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

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. An encryption scheme which produces the same ciphertext whenever we give it as input the same plaintext and key, is called a *deterministic encryption scheme*.

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

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

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

5.4 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].

[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.

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 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 $\mathsf{TKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{X} \times \mathcal{D} \to \mathcal{T} \times \mathbb{C}$. For convenience, we shall implicitly assume the interaction of TKR with CV, and we will use $\mathsf{TKR}^{(1)}_K(x, d)$ and $\mathsf{TKR}^{(2)}_K(x, d)$ 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 $\mathsf{DTKR}[\mathsf{CV}] : \mathcal{K} \times \mathcal{T} \times \mathcal{D} \to \mathcal{X} \cup \{\perp\}$. 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 $K \in \mathcal{K}$, $\mathsf{DTKR}_K(\mathsf{TKR}^{(1)}_K(x, d), d) = x$.

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

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.

```
Experiment Exp-IND-TKR$^{\mathcal{A}}$
 1. The challenger selects $K \xleftarrow{\$} \mathcal{K}$
 2. $Q \leftarrow \emptyset$.
 3. for each query $(x, d) \in \mathcal{X} \times \mathcal{D}$ of $\mathcal{A}$,
 4.     the challenger computes
        $t \leftarrow \mathsf{TKR}_K^{(1)}(x, d)$,
        and returns $t$ to $\mathcal{A}$.
 5.     $Q \leftarrow Q \cup \{(x, d)\}$
 6. until $\mathcal{A}$ stops querying
 7. $\mathcal{A}$ selects $(x_0, d_0), (x_1, d_1) \in (\mathcal{X} \times \mathcal{D}) \setminus Q$
    and sends them to the challenger
 8. The challenger selects a bit $b \xleftarrow{\$} \{0, 1\}$
    and returns $t \leftarrow \mathsf{TKR}_K^{(1)}(x_b, d_b)$ to $\mathcal{A}$.
 9. The adversary $\mathcal{A}$ outputs a bit $b'$.
10.If $b = b'$ output $1$ else output $0$.
```

```
Experiment Exp-IND-TKR-CV$^{\mathcal{A}}$
 1. The challenger selects $K \xleftarrow{\$} \mathcal{K}$
 2. $Q \leftarrow \emptyset$.
 3. for each query $(x, d) \in \mathcal{X} \times \mathcal{D}$ of $\mathcal{A}$,
 4.     the challenger computes
        $(t, c) \leftarrow \mathsf{TKR}_K(x, d)$,
        and returns $(t, c)$ to $\mathcal{A}$.
 5.     $Q \leftarrow Q \cup \{(x, d)\}$
 6. until $\mathcal{A}$ stops querying
 7. $\mathcal{A}$ selects $(x_0, d_0), (x_1, d_1) \in (\mathcal{X} \times \mathcal{D}) \setminus Q$
    and sends them to the challenger
 8. The challenger selects a bit $b \xleftarrow{\$} \{0, 1\}$
    and returns $(t, c) \leftarrow \mathsf{TKR}_K(x_b, d_b)$ to $\mathcal{A}$.
 9. The adversary $\mathcal{A}$ outputs a bit $b'$.
10.If $b = b'$ output $1$ else output $0$.
```

```
Experiment Exp-IND-TKR-KEY$^{\mathcal{A}}$
 1. The challenger selects $K \xleftarrow{\$} \mathcal{K}$
 2. $Q \leftarrow \emptyset$.
 3. for each query $(x, d) \in \mathcal{X} \times \mathcal{D}$ of $\mathcal{A}$,
 4.     the challenger computes
        $t \leftarrow \mathsf{TKR}_K^{(1)}(x, d)$, and returns $t$ to $\mathcal{A}$.
 5.     $Q \leftarrow Q \cup \{(x, d)\}$
 6. until $\mathcal{A}$ stops querying
 7. $\mathcal{A}$ selects $(x_0, d_0), (x_1, d_1) \in (\mathcal{X} \times \mathcal{D}) \setminus Q$ and sends them to the challenger
 8. The challenger selects a bit $b \xleftarrow{\$} \{0, 1\}$ and returns $t \leftarrow \mathsf{TKR}_K^{(1)}(x_b, d_b)$ and $K$ to $\mathcal{A}$.
 9. The adversary $\mathcal{A}$ outputs a bit $b'$.
10.If $b = b'$ output $1$ else output $0$.
```

Figure 5.2: Experiments used in the security definitions: IND-TKR, IND-TKR-CV and IND-TKR-KEY

5.5 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 inaccessible.

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 it is usually done for encryption schemes. Three experiments corresponding to the three attack scenarios discussed above are described in Figure 5.2. 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 5.1. *Let* $\mathsf{TKR[CV]} : \mathcal{K} \times \mathcal{X} \times \mathcal{D} \to \mathcal{T} \times \mathbb{C}$ *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*

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

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-vault 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 uses this service.

3. IND-TKR-KEY is a stronger form of the IND-TKR notion. Some public documents like [71] have 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.

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

Figure 5.3: The TKR1 tokenization scheme using a format preserving encryption scheme FP.

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 5.7 are very different. For the later constructions our main aim is to by-pass the use of FPE schemes and use standard cryptographic schemes along with some encoding mechanism to achieve both security and the format requirements for arbitrarily formatted PANs/tokens.

5.6 Construction TKR1: Tokenization Using FPE

The construction TKR1 is described in Figure 5.3. 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 5.1. *1. Let $\Psi = \mathsf{TKR1}$ be defined as in Figure 5.3, and $\mathcal{A}$ be an adversary attacking Ψ in the IND-TKR sense. Then there exists a $\widetilde{prp}$ adversary $\mathcal{B}$ such that*

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

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

2. Let $\Psi = \mathsf{TKR1}$ be defined as in Figure 5.3, and $\mathcal{A}$ be an adversary attacking Ψ in the IND-TKR-CV sense. Then there exists a $\widetilde{prp}$ adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that

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

Proof. We only prove the first claim in the theorem, the second claim directly follows from the first, as in the construction TKR1, there is no card-vault, thus an IND-TKR-CV

```
**Adversary** $\mathcal{B}^{\mathcal{O}(.,.)}$
    Whenever $\mathcal{B}$ gets a query $(x, d)$ from A
    **do** the following until $\mathcal{A}$ stops querying
        $t_i \leftarrow \mathcal{O}(x, d)$;
        **return** ($t$,NULL) to $\mathcal{A}$
    After $\mathcal{A}$ submits $(m_0, d_0), (m_1, d_1)$
    **do** the following
        $b \xleftarrow{\$} \{0, 1\}$;
        $t^* \leftarrow \mathcal{O}(m_b, d_b)$
        **return** $(t^*, \mathrm{NULL})$ to $\mathcal{A}$;
    $\mathcal{A}$ returns a bit $b'$;
    **if** $b = b'$,
        **return** 1;
    **else return** 0;
```

Figure 5.4: Adversary $\mathcal{B}$ for the proof of Theorem 5.1

adversary for TKR1 do not have any additional information compared to an IND-TKR adversary.

To prove the first claim, we construct a $\widetilde{\text{prp}}$ adversary $\mathcal{B}$ which runs an arbitrary adversary $\mathcal{A}$ who attacks TKR1. $\mathcal{B}$ being a $\widetilde{\text{prp}}$ adversary has access to an oracle $\mathcal{O}(.,.)$, which is either the real tweakable permutation $\mathsf{FP}_k(.,.)$ for a randomly chosen key k, or a random permutation chosen uniformly at random from the set of all tweak index permutations from $\mathcal{T}$ to $\mathcal{T}$. $\mathcal{B}$ with its oracle provides the environment to $\mathcal{A}$ and simulates the experiment EXP-IND-TKR$^{\mathcal{A}}_{\mathsf{TKR1}}$ as shown in Figure 5.4.

We assume without loss of generality that $\mathcal{A}$ does not repeat queries, as $\mathcal{A}$ knows that TKR1 is a deterministic scheme, hence it does not gain anything by repeating a query.

It is easy to see that if the oracle $\mathcal{O}(.,.)$ of $\mathcal{B}$ is $\mathsf{FP}_k(.,.)$, then $\mathcal{B}^{\mathcal{O}}$ provides the perfect environment for $\mathcal{A}$ as in EXP-IND-TKR$^{\mathcal{A}}_{\mathsf{TKR1}}$. Hence,

$$\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{FP}_k(.,.)} \Rightarrow 1] = \Pr[\text{EXP-IND-TKR}^{\mathcal{A}}_{\mathsf{TKR1}} \Rightarrow 1]. \tag{5.1}$$

Also,

$$\Pr[\pi \xleftarrow{\$} \mathbf{Perm}^{\mathcal{D}}(\mathcal{T}) : \mathcal{B}^{\pi(.,.)} \Rightarrow 1] \leq \frac{1}{2}, \tag{5.2}$$

as, when $\mathcal{O}(.,.)$ is a uniform random tweakable permutation on $\mathcal{T}$, for each of its queries

```
TKR2_k(x, d)
 1. $S_1 \leftarrow$ SrchCV$(2, x)$;
 2. $S_2 \leftarrow$ SrchCV$(3, d)$;
 3. $S \leftarrow S_1 \cap S_2$;
 4. if $S = \emptyset$
 5.     $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k]()$;
 6.     $c \leftarrow (t, x, d)$;
 7.     InsertCV$(c)$;
 8. else let tup $\in S$
 9.     $t \leftarrow \mathsf{tup}^{(1)}$
 10.    $c \leftarrow (t, x, d)$
 11. end if
 12.return $(t, c)$

DTKR2_k(t, d)
 1. $S_1 \leftarrow$ SrchCV$(1, t)$;
 2. $S_2 \leftarrow$ SrchCV$(3, d)$;
 3. $S \leftarrow S_1 \cap S_2$;
 4. if $S = \emptyset$
 5.     return $\perp$ ;
 6. else let tup $\in S$
 7.     $x \leftarrow \mathsf{tup}^{(2)}$;
 8. end if
 9. return $x$
```

Figure 5.5: The TKR2 tokenization scheme using a random number generator $\mathsf{RN}^{\mathcal{T}}()$.

$\mathcal{A}$ gets uniform random elements in $\mathcal{T}$, thus b' which $\mathcal{A}$ outputs is independent of b which is selected by $\mathcal{B}$.

Hence from equations (5.1) and (5.2), we have

$$\mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathsf{FP}}(\mathcal{B}) \geq \Pr[\text{EXP-IND-TKR}^{\mathcal{A}}_{\mathsf{TKR1}} \Rightarrow 1] - \frac{1}{2},$$

and hence

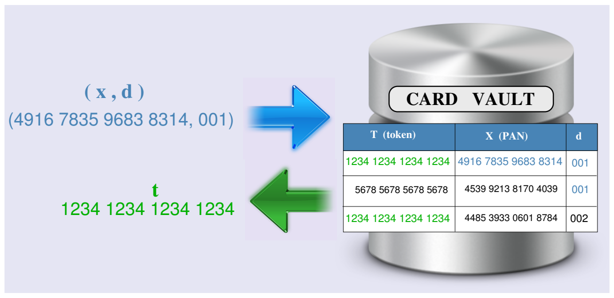
$$\mathbf{Adv}^{\text{ind-tkr}}_{\mathsf{TKR1}}(\mathcal{A}) \leq \mathbf{Adv}^{\widetilde{\mathrm{prp}}}_{\mathsf{FP}}(\mathcal{B}),$$

as desired. □

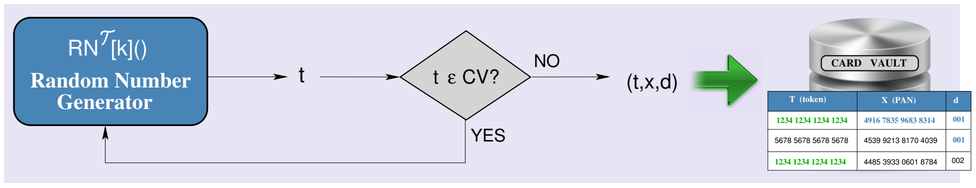
This scheme can be instantiated using any format preserving encryption scheme as described in [8, 10, 14, 36, 53, 75].

5.7 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



(a)



(b)

Figure 5.6: a) Step 1: b) Step2

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

$$\mathbf{Adv}^{\mathrm{rnd}}_{\mathsf{RN}}(\mathcal{A}) \;=\; \left|\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{RN}^{\mathcal{T}}[k]()} \Rightarrow 1] - \Pr[\mathcal{A}^{\$^{\mathcal{T}}()} \Rightarrow 1]\right|. \tag{5.3}$$

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 5.5, 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 it is evident from the description in Figure 5.5, 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 $(x, d) \neq (x', d')$.

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

Figure 5.7: Modified TKR2 to ensure uniqueness

As discussed before, for practical purposes a weak form of uniqueness is required, 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 his databases. Thus if $d \neq d'$, it can be tolerated that $\mathsf{TKR2}(x, d) = \mathsf{TKR2}(x', d')$, for any $x, x' \in \mathcal{X}$.

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 5.7. 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 a unique token is obtained would be small.

[2]According to [17] 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.

```
$\mathsf{TKR2a}_{k_1,k_2}(x,d)$
1.  $z \leftarrow \mathbf{E}_{k_1}(d, \mathsf{a}||x)$;
2.  $S \leftarrow \mathsf{SrchCV}(2, z)$;
3.  if $S = \emptyset$,
4.      do
5.          $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;
6.          $t' \leftarrow \mathbf{E}_{k_1}(d, \mathsf{b}||t)$;
7.      while $\mathsf{SrchCV}(1, t') \neq \emptyset$;
8.      $c \leftarrow (t', z)$;
9.      $\mathsf{InsertCV}(c)$;
10. else let $\mathsf{tup} \in S$
11.     $t'' \leftarrow \mathsf{tup}^{(1)}$
12.     $t' \leftarrow \mathbf{E}_{k_1}^{-1}(d, t')$
13.     $c \leftarrow \mathsf{tup}$;
14.     parse $t'$ as $\mathsf{b}||t$;
15. end if
16. return $(t, c)$
```

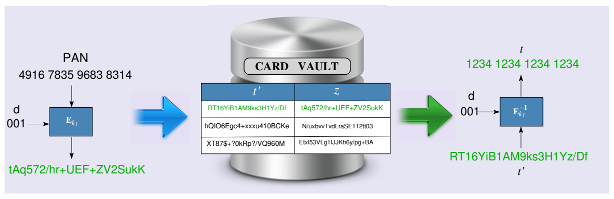
```
$\mathsf{DTKR2a}_{k_2}(t,d)$
1.  $t' \leftarrow \mathbf{E}_{k_1}(d, \mathsf{b}||t)$;
2.  $S \leftarrow \mathsf{SrchCV}(1, t')$;
3.  if $S = \emptyset$
4.      return $\perp$;
5.  else let $\mathsf{tup} \in S$
6.      $z \leftarrow \mathsf{tup}^{(2)}$;
7.      $x' \leftarrow \mathbf{E}_{k_1}^{-1}(d, z)$;
8.      parse $x'$ as $\mathsf{a}||x$;
9.  end if
10. return $x$
```

Figure 5.8: The TKR2a tokenization scheme.

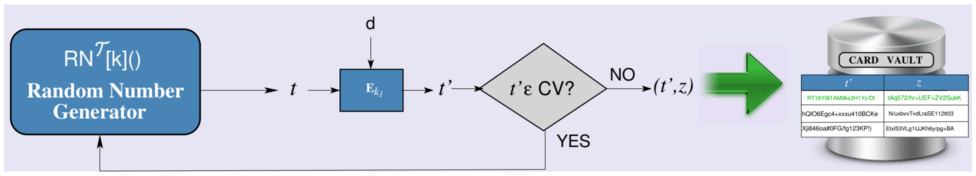
The detokenization corresponding to the modified tokenization scheme described in Figure 5.7 remains the same as described in Figure 5.5.

We formally specify the security of TKR2 later in Section 5.8, 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 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 is a bit different than for TKR2. In this case, each tuple contains two components. The first being the encryption of the token and the second being the encryption of the PAN. We additionally use a deterministic CPA secure encryption (supporting associated data) scheme $\mathbf{E} : \mathbb{K} \times \mathcal{D} \times \mathcal{M} \rightarrow \mathbb{C}$, with key space $\mathcal{K}$, tweak (associated data) space $\mathcal{D}$ and message space $\mathcal{M}$. We assume that $\mathcal{T} = \mathcal{X} = \mathsf{AL}^*$, where AL is an arbitrary alphabet,



(a)



(b)

Figure 5.9: a) Step 1: b) Step2

such that $\#\mathsf{AL} \geq 2$. We fix $\mathsf{a}, \mathsf{b} \in \mathsf{AL}$ and define the message space $\mathcal{M}$ of $\mathbf{E}$ to be

$$\mathcal{M} = \{\mathsf{a}||x : x \in \mathcal{X}\} \bigcup \{\mathsf{b}||t : t \in \mathcal{T}\}.$$

Note that a and b are public quantities. 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. We assume that $\mathcal{D}, \mathbb{C} \subseteq \{0, 1\}^*$.

The tokenization scheme TKR2a described in Figure 5.8 uses the objects described above. The main difference with TKR2 is that pairs of token and PAN are stored in the card-vault in the encrypted form. An important characteristic of the way the encryption is applied is that the inputs are differently encoded in case of a token and a PAN. This ensures that even if a PAN and a token are the same, they produce different ciphertexts.

5.8 Security of TKR2 and TKR2a

The following three theorems specify the security of TKR2 and TKR2a.

Theorem 5.2. *Let* $\Psi \in \{\mathsf{TKR2}, \mathsf{TKR2a}\}$ *and let* $\mathcal{A}$ *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*

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

Proof. Note that the token generation algorithm for both TKR2 and TKR2a are the same, the only difference between the two procedures is the structure and content of the card-vault. Hence the proof of security in IND-TKR sense for both TKR2 and TKR2a are same, as in case of IND-TKR security the adversary does not have access to the contents of the card-vault.

The structure of the proof is same as the proof of Theorem 5.1. We assume an arbitrary adversary $\mathcal{A}$ which attacks TKR2 in IND-TKR sense, and we construct a RND adversary $\mathcal{B}$ which attacks $\mathsf{RN}^{\mathcal{T}}[k]$ using $\mathcal{A}$.

$\mathcal{B}$ has an oracle $\mathcal{O}$, which is either $\mathsf{RN}^{\mathcal{T}}[k]$ for a random key, or $\$^{\mathcal{T}}()$, which on each invocation returns a random element in $\mathcal{T}$.

$\mathcal{B}$ responds to queries of $\mathcal{A}$ as follows. First $\mathcal{B}$ initiates with an empty card-vault and then $\mathcal{B}$ performs the query phase, which in fact is the procedure $\mathsf{TKR2}_k$ in Figure 5.7. Only when a call to $\mathsf{RN}^{\mathcal{T}}[k]()$ is required, it is replaced by a call to its oracle $\mathcal{O}$. After $\mathcal{A}$ stops querying and outputs the challenge pair $(m_0, d_0), (m_1, d_1)$, $\mathcal{B}$ selects a bit b uniformly at random from $\{0, 1\}$ and provides $\mathcal{A}$ with t computed by following $\mathsf{TKR2}_k()$ (the call to $\mathsf{RN}^{\mathcal{T}}[k]()$ replaced by a call to $\mathcal{O}$). Finally $\mathcal{A}$ outputs a bit b', and if $b = b'$, then $\mathcal{B}$ outputs 1 else outputs a 0. Note that the challenge pair $(m_0, d_0), (m_1, d_1)$, is different from any previous query of $\mathcal{A}$.

From the above description it is clear that if the oracle $\mathcal{O}(.,.)$ of $\mathcal{B}$ is $\mathsf{RN}^{\mathcal{T}}[k]()$, then $\mathcal{B}$ is performing experiment EXP-IND-TKR$^{\mathcal{A}}_{\mathsf{TKR2}}$. Hence

$$\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{B}^{\mathsf{RN}^{\mathcal{T}}[k]()} \Rightarrow 1] = \Pr[\text{EXP-IND-TKR}^{\mathcal{A}}_{\mathsf{TKR2}} \Rightarrow 1]. \tag{5.4}$$

Otherwise, i.e. if the oracle $\mathcal{O}(.,.)$ of $\mathcal{B}$ is $\$^{\mathcal{T}}()$ then,

$$\Pr[\mathcal{B}^{\$^{\mathcal{T}}()} \Rightarrow 1] \leq \frac{1}{2}. \tag{5.5}$$

As in this case the output that $\mathcal{B}$ provides to $\mathcal{A}$ is independent of $(m_0, d_0), (m_1, d_1)$.

From equations (5.4), (5.5) we have,

$$\mathbf{Adv}^{\mathrm{rnd}}_{\mathsf{RN}}(\mathcal{B}) \geq \Pr[\text{EXP-IND-TKR}^{\mathcal{A}}_{\mathsf{TKR2}} \Rightarrow 1] - \frac{1}{2},$$

and from the definition of IND-TKR advantage of $\mathcal{A}$ it follows

$$\mathbf{Adv}^{\text{ind-tkr}}_{\mathsf{TKR2}}(\mathcal{A}) \leq \mathbf{Adv}^{\mathrm{rnd}}_{\mathsf{RN}}(\mathcal{B}).$$

□

Theorem 5.3. *Let* $\Psi = \mathsf{TKR2a}$ *and let* $\mathcal{A}$ *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

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

where s is the size of the shortest element in the cipher space of $\mathbf{E}$.

Proof. For this proof we use the sequence of games. The three games $\mathbf{EXP}_0^{\mathcal{A}}$, $\mathbf{EXP}_1^{\mathcal{A}}$ and $\mathbf{EXP}_2^{\mathcal{A}}$ are described in Figure 5.10. Each game depicts the interaction of an IND-TKR-CV adversary with a tokenization procedure. In all the three games we assume that the adversary $\mathcal{A}$ does not repeat a query in the query phase, and the queries presented in the challenge phase are also distinct from the queries made in the query phase. Also, to keep things simple in terms of notations, without loss of generality we assume that the ciphertext space $\mathbb{C}$ of the encryption algorithm $\mathbf{E}$ contains strings of length s. The proof can be made to work without this restriction. We describe the three different games briefly next:

1. In game $\mathbf{EXP}_0^{\mathcal{A}}$, $\mathcal{A}$ interacts with $\mathsf{TKR2a}$, instantiated by $\mathsf{RN}^{T}[k_2]()$ and $\mathbf{E}_{k_1}(\cdot,\cdot)$, where k_1 and k_2 are chosen uniformly at random from the respective key spaces $\mathcal{K}_1$ and $\mathcal{K}_2$. The game is designed with the assumption that, $\mathcal{A}$ does not repeat a query.

2. Game $\mathbf{EXP}_1^{\mathcal{A}}$ is almost same as the game $\mathbf{EXP}_0^{\mathcal{A}}$. The differences are as follows:

   - Here the encryption scheme $\mathbf{E}_{k_1}(\cdot,\cdot)$, is no more used. Instead, each call to $\mathbf{E}_{k_1}(\cdot,\cdot)$ is responded by a random string from $\mathbb{C}$. To maintain the same behaviour of $\mathbf{E}_{k_1}$, a set Ran_1 is maintained to keep track of the values already returned as output, and it is ensured that the same value is not returned for two different inputs.
   - In the game $\mathbf{EXP}_0^{\mathcal{A}}$, in lines 11 to 14 and 53 to 56 it is ensured that a distinct token t is returned for each distinct (x, d). This is done by a search in the card-vault (see lines 14 and 56), as the card-vault contains encryption of the token t with associated data d. As in the game $\mathbf{EXP}_1^{\mathcal{A}}$, a real encryption scheme is not used, so this search is not possible. Hence a set Tok is maintained which contains pairs of tokens and associated data (t, d) and the uniqueness of tokens is ensured using this set Tok.

3. Game $\mathbf{EXP}_2^{\mathcal{A}}$ is obtained from game $\mathbf{EXP}_1^{\mathcal{A}}$ by replacing $\mathsf{RN}^{\mathcal{T}}[k_2]()$ by a procedure which on each invocation returns a random element in $\mathcal{T}$. This game also used the sets Ran_1 and Tok to ensure injectivity and the uniqueness of the tokens.

Game $\mathbf{EXP}_0^{\mathcal{A}}$

```
Initialization:
    01. CV ← NULL;
    02. $k_1 \xleftarrow{\$} \mathcal{K}_1$;
    03. $k_2 \xleftarrow{\$} \mathcal{K}_2$;

Query Phase
Respond to a query $(x, d)$
by $\mathcal{A}$ as follows
    10. $z \leftarrow \mathbf{E}_{k_1}(x, d)$;
    11. do
    12.     $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;
    13.     $t' \leftarrow \mathbf{E}_{k_1}(d, \mathsf{b}||t)$;
    14. while $\mathsf{SrchCV}(1, t') \neq \emptyset$;
    15. $c \leftarrow (t', z)$;
    16. $\mathsf{InsertCV}(c)$;
    17. return $(t, c)$ to $\mathcal{A}$

Challenge Phase
After $\mathcal{A}$ submits $(x_0, d_0), (x_1, d_1)$
do the following:
    51. $b \xleftarrow{\$} \{0, 1\}$;
    52. $z \leftarrow \mathbf{E}_{k_1}(x_b, d_b)$;
    53. do
    54.     $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;
    55.     $t' \leftarrow \mathbf{E}_{k_1}(d, \mathsf{b}||t)$;
    56. while $\mathsf{SrchCV}(1, t') \neq \emptyset$;
    57. $c \leftarrow (t', z)$;
    58. return $(t, c)$ to $\mathcal{A}$

Finalization Phase
After $\mathcal{A}$ outputs the bit $b'$
do the following:
    80. if $b = b'$ output 1
    81. else output 0
```

Game $\mathbf{EXP}_1^{\mathcal{A}}$

```
Initialization:
    01. CV ← NULL;
    02. $k_2 \leftarrow \mathcal{K}_2$;
    03. $\mathsf{Ran}_1 \leftarrow \emptyset$
    04. $\mathsf{Tok} \leftarrow \emptyset$

Query Phase
Respond to a query $(x, d)$
by $\mathcal{A}$ as follows
    10. $z \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    11. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{z\}$;
    12. do, $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;
    13. while $\mathsf{Tok} \cap \{(t, d)\} \neq \emptyset$;
    14. $\mathsf{Tok} \leftarrow \mathsf{Tok} \cup \{(t, d)\}$;
    15. $t' \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    16. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{t'\}$;
    17. $c \leftarrow (t', z)$;
    18. $\mathsf{InsertCV}(c)$;
    19. return $(t, c)$ to $\mathcal{A}$

Challenge Phase
After $\mathcal{A}$ submits $(x_0, d_0), (x_1, d_1)$
do the following:
    51. $b \xleftarrow{\$} \{0, 1\}$;
    52. $z \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    53. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{z\}$;
    54. do $t \leftarrow \mathsf{RN}^{\mathcal{T}}[k_2]()$;
    55. while $\mathsf{Tok} \cap \{(t, d_b)\} \neq \emptyset$;
    56. $\mathsf{Tok} \leftarrow \mathsf{Tok} \cup \{(t, d_b)\}$;
    57. $t' \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    58. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{t'\}$;
    59. $c \leftarrow (t', z)$;
    60. return $(t, c)$ to $\mathcal{A}$

Finalization Phase
After $\mathcal{A}$ outputs the bit $b'$
do the following:
    80. if $b = b'$ output 1
    81. else output 0
```

Game $\mathbf{EXP}_2^{\mathcal{A}}$

```
Initialization:
    01. CV ← NULL;
    02. $\mathsf{Ran}_1 \leftarrow \emptyset$
    03. $\mathsf{Tok} \leftarrow \emptyset$

Query Phase
Respond to a query $(x, d)$
by $\mathcal{A}$ as follows:
    10. $z \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    11. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{z\}$;
    12. do $t \xleftarrow{\$} \mathcal{T}$;
    13. while $\mathsf{Tok} \cap \{(t, d)\} \neq \emptyset$;
    14. $\mathsf{Tok} \leftarrow \mathsf{Tok} \cup \{(t, d)\}$;
    15. $t' \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    16. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{t'\}$;
    17. $c \leftarrow (t', z)$;
    18. $\mathsf{InsertCV}(c)$;
    19. return $(t, c)$ to $\mathcal{A}$

Challenge Phase
After $\mathcal{A}$ submits $(x_0, d_0), (x_1, d_1)$
do the following:
    51. $b \xleftarrow{\$} \{0, 1\}$;
    52. $z \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    53. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{z\}$;
    54. do, $t \xleftarrow{\$} \mathcal{T}$;
    55. while $\mathsf{Tok} \cap \{(t, d_b)\} \neq \emptyset$;
    56. $\mathsf{Tok} \leftarrow \mathsf{Tok} \cup \{(t, d_b)\}$;
    57. $t' \xleftarrow{\$} \mathbb{C} \setminus \mathsf{Ran}_1$;
    58. $\mathsf{Ran}_1 \leftarrow \mathsf{Ran}_1 \cup \{t'\}$;
    59. $c \leftarrow (t', z)$;
    60. return $(t, c)$ to $\mathcal{A}$

Finalization Phase
After $\mathcal{A}$ outputs the bit $b'$
do the following:
    80. if $b = b'$ output 1
    81. else output 0
```

Figure 5.10: The three games used to prove Theorem 5.3

It is easy to see that $\mathbf{EXP}_0^{\mathcal{A}}$ is a restatement of the experiment Exp-IND-TKR-CV$^{\mathcal{A}}$ in Figure 5.2. Hence,

$$\Pr[\text{Exp-IND-TKR-CV}^{\mathcal{A}} \Rightarrow 1] = \Pr[\mathbf{EXP}_0^{\mathcal{A}} \Rightarrow 1]. \tag{5.6}$$

Also we make the following claims:

Claim 5.1. *There exists a DET-CPA adversary $\mathcal{B}$ for* $\mathbf{E}$ *such that*

$$\Pr[\mathbf{EXP}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_{\mathbf{E}}^{\text{det-cpa}}(\mathcal{B}) + \frac{(q+1)^2}{2^s}$$

Proof. To prove this claim we construct a DET-CPA adversary $\mathcal{B}$ which has access to an oracle $\mathcal{O}$. This oracle is either the encryption scheme $\mathbf{E}_{k_1}$ for a random key k_1 or $\$(\cdot,\cdot)$ which on input (x,d) returns random strings of length s. $\mathcal{B}$ has the objective of distinguishing between these two scenarios. $\mathcal{B}$ runs $\mathcal{A}$ in the following way. First $\mathcal{B}$ initiates with an empty card-vault and selects a random key k_2 from $\mathcal{K}_2$, and also initializes a multi-set Dom to empty. Then, it answers queries of $\mathcal{A}$ according to the procedure TKR2a (shown in Figure 5.8). To answer the queries, whenever a call to the encryption scheme $\mathbf{E}_{k_1}$ is required, it is replaced by a call to its oracle $\mathcal{O}$. $\mathcal{B}$ also stores each output it gets from its oracle $\mathcal{O}$ in the set Dom. Note, as $\mathcal{A}$ does not repeat any query, hence all queries made by $\mathcal{B}$ to its oracle is distinct. After $\mathcal{A}$ stops querying and outputs a challenge pair $(x_0,d_0),(x_1,d_1)$, $\mathcal{B}$ selects a bit uniformly at random from $\{0,1\}$ and provides $\mathcal{A}$ with the pair (t,c). For responding to $\mathcal{A}$'s challenge, $\mathcal{B}$ makes another call to $\mathcal{O}$ and the output of $\mathcal{O}$ for this call is also inserted in Dom. Finally $\mathcal{A}$ outputs a bit b'. Now, $\mathcal{B}$ checks if there is a collision in Dom, i.e., if $\mathcal{O}$ ever returned two same values for two distinct queries. If there is a collision in Dom, then $\mathcal{B}$ outputs 0. On the other hand, if there is no collision in Dom and $b = b'$ then $\mathcal{B}$ outputs 1, otherwise it outputs a 0.

From the description above, we can easily see that if the oracle of $\mathcal{B}$ is the encryption scheme $\mathbf{E}_{k_1}(\cdot,\cdot)$, then there is never a collision in Dom as $\mathbf{E}_{k_2}(\cdot,\cdot)$ is injective, and in this scenario $\mathcal{B}$ is providing the exact environment of the game $\mathbf{EXP}_0^{\mathcal{A}}$, i.e.

$$\Pr[k_1 \xleftarrow{\$} \mathcal{K}_1 : \mathcal{B}^{\mathcal{E}_K(\cdot,\cdot)} \Rightarrow 1] \leq \Pr[\mathbf{EXP}_0^{\mathcal{A}} \Rightarrow 1]. \tag{5.7}$$

On the other hand, if the oracle of $\mathcal{B}$ is $\$(\cdot,\cdot)$, then $\mathcal{B}$ is providing the environment of $\mathbf{EXP}_1^{\mathcal{A}}$, given that there is no collision in Dom. If COLL be the event that there is a collision in Dom, then we have,

$$\begin{aligned}
\Pr[\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0] &= \Pr[(\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0) \wedge (\mathsf{COLL} \vee \overline{\mathsf{COLL}})] \\
&= \Pr[(\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0) \wedge \mathsf{COLL}] + \Pr[(\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0) \wedge \overline{\mathsf{COLL}})] \\
&= \Pr[(\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0)|\mathsf{COLL}] \Pr[\mathsf{COLL}] + \Pr[(\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 0)|\overline{\mathsf{COLL}}] \Pr[\overline{\mathsf{COLL}}] \\
&\geq \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 0](1 - \Pr[\mathsf{COLL}]).
\end{aligned}$$

Thus

$$\begin{aligned}\Pr[\mathcal{B}^{\$(\cdot,\cdot)} \Rightarrow 1] &\leq \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] + \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 0]\Pr[\mathsf{COLL}] \\ &\leq \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] + \Pr[\mathsf{COLL}] \qquad (5.8)\end{aligned}$$

Now from equations (5.7) and (5.8), and the definition of DET-CPA advantage of $\mathcal{B}$, we have

$$\mathbf{Adv}_{\mathbf{E}}^{\text{det-cpa}}(\mathcal{B}) \geq \Pr[\mathbf{EXP}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathsf{COLL}].$$

As, $\mathcal{A}$ asks q queries in the query phase, hence Dom has $q+1$ elements in it, and each element is a uniform random element in $\mathbb{C}$, and each element in $\mathbb{C}$ is s bits long. Hence,

$$\Pr[\mathsf{COLL}] = \binom{q+1}{2}\frac{1}{2^s} \leq \frac{(q+1)^2}{2^{s+1}}.$$

This completes the proof of the claim.

Claim 5.2. *There exists a* RND *adversary $\mathcal{B}'$ such that*

$$\Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{EXP}_2^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_{\mathsf{RN}^{\mathcal{T}}}^{\text{rnd}}(\mathcal{B}')$$

Proof. The proof of this claim is an easy reduction. Again we have an adversary $\mathcal{A}$ attacking TKR2a and we must construct a RND adversary $\mathcal{B}'$, which runs $\mathcal{A}$. $\mathcal{B}'$ has access to an oracle $\mathcal{O}$, that could be either $\mathsf{RN}^T[k_2]()$ or $\$^{\mathcal{T}}$, which on each invocation it returns a random element in $\mathcal{T}$. As in Claim 5.1, adversary $\mathcal{B}'$ do an initialization and a query phase, but now when a call to $\mathsf{RN}^T[k]()$ is required, it is substituted by a call to the oracle $\mathcal{O}$. Now we can see that

$$\Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{B}'^{\mathsf{RN}^{\mathcal{T}}[k]()} \Rightarrow 1] = \Pr[\mathbf{EXP}_1^{\mathcal{A}} \Rightarrow 1] \qquad (5.9)$$

in the case that the oracle of $\mathcal{B}$ is $\mathsf{RN}^T[k]()$, otherwise i.e., if $\mathcal{O}$ is $\$^{\mathcal{T}}$ then

$$\Pr[\mathcal{B}'^{\$^{\mathcal{T}}()} \Rightarrow 1] \leq \Pr[\mathbf{EXP}_2^{\mathcal{A}} \Rightarrow 1] \qquad (5.10)$$

Again from equations (5.9) and (5.10), the claim follows.

Claim 5.3. *For any arbitrary adversary $\mathcal{A}$*

$$\Pr[\mathbf{EXP}_2^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}$$

Proof. In game $\mathbf{EXP}_2^{\mathcal{A}}$, in the query phase $\mathcal{A}$ receives q tuples (t,c) where t and c are distinct random elements in $\mathcal{T}$ and $\mathbb{C}$, respectively. Finally in the challenge phase it receives (t,c) which is independent of $(x_0,d_0),(x_1,d_1)$. Hence, $\mathcal{A}$ cannot only guess the bit b with probability more than $\frac{1}{2}$.

Thus, from Claims 5.1, 5.2,

$$\Pr[\mathbf{EXP}_0^{\mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{EXP}_2^{\mathcal{A}} \Rightarrow 1] \leq \mathbf{Adv}_{\mathbf{E}}^{\text{det-cpa}}(\mathcal{B}) + \mathbf{Adv}_{\mathsf{RN}^{\mathcal{T}}}^{\text{rnd}}(\mathcal{B}') + \frac{(q+1)^2}{2^{s+1}} \quad (5.11)$$

Using equation (5.6) and claim 5.3,

$$\Pr[\text{Exp-IND-TKR-CV}^{\mathcal{A}} \Rightarrow 1] - \frac{1}{2} \leq \mathbf{Adv}_{\mathbf{E}}^{\text{det-cpa}}(\mathcal{B}) + \mathbf{Adv}_{\mathsf{RN}^{\mathcal{T}}}^{\text{rnd}}(\mathcal{B}') + \frac{(q+1)^2}{2^{s+1}}. \quad (5.12)$$

Finally, we have

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

as desired. □

Theorem 5.4. *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*

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

5.9 Summary

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 we proved their security. The constructions TKR2 and TKR2a make use of a primitive $\mathsf{RN}^{\mathcal{T}}[k]$ and an encryption scheme $\mathbf{E}$. In the next Chapter we will explain how to instantiate them.

Chapter 6

Practical instantiations of TKR2 and TKR2a

In the previous Chapter we started the study of tokenization systems. We stated three security notions for these systems and we also proposed three constructions, TKR1, TKR2 and TKR2a. The last two constructions use a primitive denoted as $\mathsf{RN}^{\mathcal{T}}[k]$, which ideally generates a token uniformly at random. In particular TKR2a encrypts the information in the card-vault, with a deterministic CPA secure encryption scheme, denoted as $\mathbf{E}$. In this Chapter we will describe how to implement these two primitives using specific cryptographic objects. We also prove the security of $\mathsf{RN}^{\mathcal{T}}[k]$ and $\mathbf{E}$, when they are instantiated with specific cryptographic objects. Finally, we will present the results that we obtained by implementing TKR2 and TKR2a, when the primitive $\mathsf{RN}^{\mathcal{T}}[k]$ is instantiated using a block cipher.

6.1 Notations

In this Section we introduce additional notation which will be used throughout this Chapter. Let AL be an arbitrary alphabet, and let $Y \in \mathsf{AL}^*$ be a string over this alphabet, 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. 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.

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

The heart of the procedures TKR2 and TKR2a is the keyed primitive $\mathsf{RN}^{\mathcal{T}}[k]$, which can be realized by standard cryptographic objects. 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 it is shown in Figure 6.1.

For the construction shown in Figure 6.1, 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 $\sigma : \mathsf{AL} \to \{0,1,2,\ldots,\ell-1\}$ 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

```
RN[$f_k$]()
 1. $X \leftarrow f_k(cnt)$;
 2. $X_1||X_2||\ldots||X_m \leftarrow \mathsf{format}_\lambda(X)$;
 3. $Y \leftarrow \epsilon$; (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 6.1: Construction of $\mathsf{RN}()$ using a pseudorandom function $f_k()$.

algorithm is to generate a "long" binary string using $f_k(cnt)$ and divide the string into blocks of λ bits. If the integer corresponding to a block is less than ℓ then it is accepted otherwise it is discarded. 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 6.1) 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 6.1. *Let $\mathcal{A}$ be an arbitrary adversary attacking $\mathsf{RN}[f_k]$ (as described in Figure 6.1) in the RND sense. Then there exists a PRF adversary $\mathcal{B}$ (which uses almost the same resources as of $\mathcal{A}$) such that*

$$\mathbf{Adv}^{\mathsf{rnd}}_{\mathsf{RN}[f_k]()}(\mathcal{A}) \leq \mathbf{Adv}^{\mathsf{prf}}_{f_k}(\mathcal{B}). \tag{6.1}$$

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

6.2.1 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 [66] 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 $\mathsf{SC}_K : \{0,1\}^{\ell} \rightarrow \{0,1\}^{L}$ 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 [11] 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 $E : \mathcal{K} \times \{0,1\}^n \rightarrow \{0,1\}^n$ 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** $\mathsf{take}_L(Z)$
```

$\mathbf{E1}_K(d, x)$	$\mathbf{E1}_K^{-1}(d, z)$	$\mathbf{E2}_K(d, x)$	$\mathbf{E2}_K^{-1}(d, z)$
1. $z_1 \leftarrow \mathsf{pad}_1(x)$;	1. $y \leftarrow E_K^{-1}(z)$;	1. $z_1 \leftarrow \mathsf{pad}_{\mathcal{X}}(x)$;	1. $y \leftarrow E_K^{-1}(z)$;
2. $z_2 \leftarrow \mathsf{pad}_2(d)$;	2. $z_1 \leftarrow \mathsf{take}_{n_1}(y)$;	2. $z_2 \leftarrow \mathsf{pad}_{\mathcal{D}}(d)$;	2. $z_2 \leftarrow \mathsf{pad}_{\mathcal{D}}(d)$;
3. $z \leftarrow E_K(z_1 \| z_2)$;	3. $x \leftarrow \mathsf{pad}_1^{-1}(z_1)$;	3. $z \leftarrow E_K(z_1 \oplus E_K(z_2))$;	3. $x \leftarrow y \oplus E_K(z_2)$;
4. **return** z	4. **return** x	4. **return** z	4. **return** x

Figure 6.2: The two instantiations of $\mathbf{E}_K$.

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 that the following holds

Proposition 6.1. *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*

$$\mathbf{Adv}_f^{\text{prf}}(\mathcal{B}) \leq \mathbf{Adv}_E^{\text{prf}}(\mathcal{B}') + \frac{m^2q^2}{2^n}.$$

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 6.1 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 [42, 44, 76]. 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 RN() as in Figure 6.1 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 RN[TR]. As such a generator gives us "true randomness", hence for any adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathsf{RN[TR]}}^{\text{rnd}} = 0$.

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

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

As discussed previously, $\mathbf{E}_{\mathbf{k}}(\cdot, \cdot)$ 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 6.2. Both the constructions uses a block cipher with a block length of n, and the padding functions defined above. In $\mathbf{E1}$, 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. $\mathbf{E2}$ is same as the construction of a tweakable block cipher proposed in [48]. 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 6.2. *Let $\mathcal{A}$ be an arbitrary det-cpa adversary attacking $\mathbf{E1}$, 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*

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

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

Proof. To prove this proposition, we construct a PRF adversary $\mathcal{B}$ (shown in Fig. 6.3) which runs an arbitrary adversary $\mathcal{A}$ who attacks the encryption scheme $\mathbf{E1}$ in the det-cpa sense. $\mathcal{B}$ being a PRF adversary has access to an oracle $\mathcal{O}$ which can be either be the block cipher E_k or a function ρ, chosen uniformly at random from $\mathsf{Func}(n)$.

We can easily see that if the oracle of $\mathcal{B}$ is the block cipher E_k then

$$\Pr[k \xleftarrow{\$} \mathbb{K} : \mathcal{B}^{E_k(\cdot)} \Rightarrow 1] = \Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathbf{E1}(\cdot,\cdot)} \Rightarrow 1]. \tag{6.2}$$

As $\mathcal{A}$ never repeats a query, so if the oracle of $\mathcal{B}$ is a random function ρ, then for each query $\mathcal{A}$ gets a uniform random n bit string as a response. Thus,

$$\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] = \Pr[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1] \tag{6.3}$$

```
**Adversary** $\mathcal{B}^{\mathcal{O}(\cdot)}$
    Whenever $\mathcal{B}$ gets a query $(d,x)$ from $\mathcal{A}$
    **do** the following until $\mathcal{A}$ stops querying
        $z_1 \leftarrow \mathsf{pad}_1(x)$ ;
        $z_2 \leftarrow \mathsf{pad}_2(d)$ ;
        $z \leftarrow \mathcal{O}(z_1||z_2)$;
        **return** $(z)$ to $\mathcal{A}$
    $\mathcal{A}$ returns a bit $b$ to $\mathcal{B}$;
    **return** $b$;
```

Figure 6.3: Adversary $\mathcal{B}$ for the proof of Proposition 6.2.

Thus from the equations above, and the definition of the det-cpa advantage of $\mathcal{A}$ and the PRF advantage of $\mathcal{B}$, we obtain

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

□

Proposition 6.3. *Let $\mathcal{A}$ be an arbitrary det-cpa adversary attacking* $\mathbf{E2}$*, 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*

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

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

Proof. As in the proof of Proposition 6.2, we construct a PRF adversary $\mathcal{B}$ (shown in Fig. 6.4) which runs an arbitrary adversary $\mathcal{A}$ who attacks the encryption scheme $\mathbf{E2}$. Adversary $\mathcal{B}$ has access to an oracle $\mathcal{O}$ which can be either a secure block cipher E_k or a pseudorandom function ρ, chosen uniformly at random from $\mathsf{Func}(n)$.

We can easily see that if the oracle of $\mathcal{B}$ is the block cipher E_k then

$$\Pr[k \xleftarrow{\$} \mathbb{K} : \mathcal{B}^{E_k(\cdot)} \Rightarrow 1] = \Pr[k \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathbf{E2}(\cdot,\cdot)} \Rightarrow 1] \qquad (6.4)$$

To analyze the situation when the oracle of $\mathcal{B}$ is a random function, we consider the game $\mathbf{G0}$ shown in Figure 6.5. The game $\mathbf{G0}$ describes a function $\mathbf{Choose}\text{-}\rho()$, which acts as a random function. It returns uniform random strings in $\{0,1\}^n$ when it is

```
**Adversary** $\mathcal{B}^{\mathcal{O}(\cdot)}$
    Whenever $\mathcal{B}$ gets a query $(d, x)$ from $\mathcal{A}$
    **do** the following until $\mathcal{A}$ stops querying
        $z_1 \leftarrow \mathsf{pad}_{\mathcal{X}}(x)$ ;
        $z_2 \leftarrow \mathsf{pad}_{\mathcal{D}}(d)$ ;
        $z \leftarrow \mathcal{O}(z_1 \oplus \mathcal{O}(z_2))$;
        **return** $z$ to $\mathcal{A}$
    $\mathcal{A}$ returns a bit $b$ to $\mathcal{B}$;
    **return** $b$;
```

Figure 6.4: Adversary $\mathcal{B}$ for the proof of Proposition 6.3.

invoked, but it returns the same string if invoked twice on the same input. It does this by maintaining a table ρ of outputs that it has already returned. Additionally in the set Dom, it maintains the points on which it has been queried. The function sets the bad flag to true if it is queried twice on the same input.

As **Choose-**ρ acts like a random function, hence it is immediate that

$$\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] = \Pr[\mathcal{A}^{G0} \Rightarrow 1] \qquad (6.5)$$

Now, we do a small change in game **G0**, i.e., we remove the boxed entry in the function **Choose-**ρ, we call this changed game as **G1**. Notice that games **G1** and **G0** are identical until the flag bad is set to true, hence we have

$$\Pr[\mathcal{A}^{G0} \Rightarrow 1] - \Pr[\mathcal{A}^{G1} \Rightarrow 1] \leq \Pr[\mathcal{A}^{G1} \text{ sets bad}] \qquad (6.6)$$

Also in game **G1**, the function **Choose-**ρ, returns random strings for any input it gets, thus $\mathcal{A}$ when interacts with **G1** gets random strings in $\{0,1\}^n$ in response to its queries. Hence,

$$\Pr[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1] = \Pr[\mathcal{A}^{G1} \Rightarrow 1]. \qquad (6.7)$$

Now, we do some small syntactic changes in the game **G1** to obtain the game **G2**, also shown in Figure 6.5. Game **G2** is only syntactically different from **G1**. In **G2** random strings are returned immediately as a response to a query of $\mathcal{A}$, and later in the finalization phase appropriate values are inserted in the multiset Dom, note as Dom is a multiset hence there can be several instances of the same element present here.

```
Game $\boxed{\mathbf{G0}}$, **G1**

**function Choose**-$\rho(X)$
    $Y \xleftarrow{\$} \{0,1\}^n$;
    **if** $X \in$ Dom **then**
        bad $\leftarrow$ true
        $\boxed{Y \leftarrow \rho[X]}$
    **else**
        $\rho[X] \leftarrow Y$
        Dom $\leftarrow$ Dom $\cup \{X\}$
    **end if**
    **return** $Y$
**Initialization**
    bad $\leftarrow$ false;
    Dom $= \emptyset$;
**Query Phase**
For a query $(d^{(i)}, x^{(i)})$ of $\mathcal{A}$ do the following
    $z_1^{(i)} \leftarrow$ pad$_{\mathcal{X}}(x^{(i)})$ ;
    $z_2^{(i)} \leftarrow$ pad$_{\mathcal{D}}(d^{(i)})$ ;
    **if** $z_2^{(i)} = z_2^{(j)}$ for some $j < i$ **then** $\mu^{(i)} \leftarrow \mu^{(j)}$
    **else** $\mu^{(i)} \leftarrow$ **Choose**-$\rho(z_2^{(i)})$
    **end if**
    $\lambda^{(i)} \leftarrow z_1^{(i)} \oplus \mu^{(i)}$
    $z^{(i)} \leftarrow$ **Choose**-$\rho(\lambda^{(i)})$
    **return** $z$
```

```
Game **G2**

**Query Phase**
For a query $(d^{(i)}, x^{(i)})$ of $\mathcal{A}$,
do the following
    $z^{(i)} \xleftarrow{\$} \{0,1\}^n$
    **return** $z^{(i)}$

**Finalization**
**for** $i \leftarrow 1$ to $q$
    $z_1^{(i)} \leftarrow$ pad$_{\mathcal{X}}(x^{(i)})$ ;
    $z_2^{(i)} \leftarrow$ pad$_{\mathcal{D}}(d^{(i)})$ ;
    **if** $z_2^{(i)} = z_2^{(j)}$ for $j < i$ **then**
        $\mu^{(i)} \leftarrow \mu^{(j)}$
    **else**
        $\mu^{(i)} \xleftarrow{\$} \{0,1\}^n$
        Dom $\leftarrow$ Dom $\cup \{z_2^{(i)}\}$
    **end if**
    $\lambda^{(i)} \leftarrow z_1^{(i)} \oplus \mu^{(i)}$
    Dom $\leftarrow$ Dom $\cup \{\lambda^{(i)}\}$
**endfor**
    if there is a collision in Dom then
        bad $\leftarrow$ true
```

Figure 6.5: Games **G0, G1, G2** used for the proof of Proposition 6.3.

As, there is no way that $\mathcal{A}$ can distinguish between **G1** and **G2**, hence

$$\Pr[\mathcal{A}^{G1} \Rightarrow 1] = \Pr[\mathcal{A}^{G2} \Rightarrow 1], \tag{6.8}$$

also

$$\Pr[\mathcal{A}^{G1} \text{ sets bad}] = \Pr[\mathcal{A}^{G2} \text{ sets bad}]. \tag{6.9}$$

Thus, using equations (6.5), (6.6), (6.7), (6.8) and (6.9) we get

$$\begin{aligned}\Pr[\rho \xleftarrow{\$} \mathsf{Func}(n) : \mathcal{B}^{\rho(\cdot)} \Rightarrow 1] &= \Pr[\mathcal{A}^{G0} \Rightarrow 1] \\ &\leq \Pr[\mathcal{A}^{G1} \Rightarrow 1] + \Pr[\mathcal{A}^{G1} \text{ sets bad}] \\ &\leq \Pr[\mathcal{A}^{G2} \Rightarrow 1] + \Pr[\mathcal{A}^{G2} \text{ sets bad}] \\ &\leq \Pr[\mathcal{A}^{\$(\cdot,\cdot)} \Rightarrow 1] + \Pr[\mathcal{A}^{G2} \text{ sets bad}] \qquad (6.10)\end{aligned}$$

Let COLLD be the event that there is a collision in the multiset Dom in game **G2**, then from the description of game **G2**, we have

$$\Pr[\mathcal{A}^{G2} \text{ sets bad}] = \Pr[\mathsf{COLLD}]$$

Now we concentrate on finding an upper bound for $\Pr[\mathsf{COLLD}]$. The elements present in Dom are d's and λ's. Let Dom $= Q_d \cup Q_\lambda$, where $Q_d \subseteq \{d^{(i)} : 1 \leq i \leq q\}$, and $Q_\lambda = \{\lambda^{(i)} = z^{(i)} \oplus \mu^{(i)} | 1 \leq i \leq q\}$.

Note, that the way the game **G2** is designed, all elements in Q_d are distinct, thus there can be no collision among two elements in Q_d. Additionally we claim the following

Claim 6.1. *For $1 \leq i, j \leq q$, $i \neq j$, $\Pr[\lambda^{(i)} = \lambda^{(j)}] \leq 1/2^n$.*

Proof. We have two cases to consider:
Case 1. If $d^{(i)} = d^{(j)}$, then $x^{(i)} \neq x^{(j)}$, as $\mathcal{A}$ does not repeat any query. This makes $z^{(i)} \neq z^{(j)}$. According to the game **G2**, if $d^{(i)} = d^{(j)}$, then $\mu^{(i)} = \mu^{(j)}$. Thus we have $\lambda^{(i)} \neq \lambda^{(j)}$. Thus, making $\Pr[\lambda^{(i)} = \lambda^{(j)}] = 0$.
Case 2. If $d^{(i)} \neq d^{(j)}$, then $\mu^{(i)}$ and $\mu^{(j)}$ are uniform and independent random elements in $\{0,1\}^n$, thus making

$$\Pr[\lambda^{(i)} = \lambda^{(j)}] = \Pr[z_1^{(i)} \oplus \mu^{(i)} = z_1^{(j)} \oplus \mu^{(j)}] = \frac{1}{2^n}.$$

Claim 6.2. *For any $d \in Q_d$ and any $\lambda \in Q_\lambda$, $\Pr[\lambda = d] \leq 1/2^n$.*

Proof. Any $\lambda \in Q_\lambda$ is a uniform random string in $\{0,1\}^n$, and is independent of any $d \in Q_d$.

Now, as $\#Q_d \leq q$ and $\#Q_\lambda = q$, using Claims 6.1, 6.2 and the union bound, we have

$$\Pr[\mathsf{COLLD}] \leq \frac{1}{2^n}\binom{q}{2} + \frac{q^2}{2^n} < \frac{2q^2}{2^n}.$$

Now, using the definition of det-cpa advantage of $\mathcal{A}$ and equations (6.4) and (6.10), we have the proposition. $\square$

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

6.4 Experimental Results

We performed some preliminary experiments to determine the efficiency and functionalities of the proposed constructions in a practical environment. All experiments reported used the following computing resources:

CPU: Four-core i5-2400 Intel processor (3.1GHz).

OS: Ubuntu 12.04.4 LTS.

DataBase: PostgreSQL 9.2.6

Compiler: gcc 4.7.3

We implemented both $\mathsf{TKR2}[f_k]$ and $\mathsf{TKR2a}[f_k]$, instantiated with $\mathsf{RN}[f_k]$ (described in Figure 6.1), where f_k was instantiated with block cipher based construction described in Section 6.2.1.

We implemented the card-vault in a PostgresSQL database. For $\mathsf{TKR2}$ we considered the card-vault to be a relation with three attributes: the token (TKN), the associated data (ASD) and the PAN. For this construction the primary key is composed by the token and the associated data. For $\mathsf{TKR2a}$ we considered the card-vault to be a relation with two attributes EPAN and ETKN, representing the encrypted PAN and token respectively. We encrypt these data using the construction $\mathbf{E1}$ described in Section 6.3. In this case ETKN was considered as the primary key.

For implementation of $f_k()$ we used AES with 128 bit key, and the implementation was done by using the new Intel AES-NI instruction set, which provides a very efficient and secure implementation of the AES. We assumed that $\mathcal{X}$ contains strings of 16 characters where each character is a decimal digit, and $\mathcal{T} = \mathcal{X}$. Thus, in accordance to our notations introduced before, we had $\mu = 16$, $\mathsf{AL} = \{0, 1 \ldots, 9\}$, thus $\lambda = \lceil \lg(\#\mathsf{AL}) \rceil = 4$, and $\mathcal{X} = \mathcal{T} = \mathsf{AL}^{\mu}$.

Experiment	**Time**(ms)	
	TRK2	TKR2a
Run1	0.19	0.26
Run2a	0.30	0.37
Run2b	0.83	0.96
Run2c	1.27	1.30
Run2d	1.49	1.52
Run2e	1.69	1.98

Table 6.1: Summary of the experimental results: The descriptions of **Run1**, **Run2a**,$\cdots$ **Run2e** are provided in the text.

The reported times are based on an $-$`O3` optimized code. The time was measured by first measuring the number of cycles necessary for a specific operation using the `rdtsc` instruction. This cycle counts we converted to real time using the processor frequency.

We summarize our experiments and the results below:

1. The first experiment was to verify how many block cipher calls are necessary for each call of $f_k()$ and the efficiency of $\mathsf{RN}^{\mathcal{T}}[f_k]$. In Section 5.7, we discussed that if the range of f_k is $\{0,1\}^L$, then $L \leq 3\lambda\mu$ would be sufficient. Note, that the number of block cipher calls required for each invocation of f_k is $m = \lceil L/\lambda \rceil$. We made 1,000,000 independent calls to f_k, and in all cases, in each call we required at most two block cipher calls. In fact in only 5% of the cases two calls were necessary. In all others only one call was sufficient. The average time required for each invocation of $\mathsf{RN}^{\mathcal{T}}[f_k]$ was 0.1 microseconds.

2. The second experiment was to see if TKR2 implemented without the uniqueness test (as described in Figure 5.5) would be sufficient. Again, we generated 1,000,000 tokens using TKR2 and they were all unique. Thus, in a practical scenario, where the card-vault would be stored in a database, the uniqueness test (as included in the description in Figure 5.7) is not required to be explicitly included. Once a token is generated and when the system tries to insert it in the database, if the uniqueness condition is violated then the database would generate an error message, and then the process may be repeated until a unique token is generated.

3. Finally we measured efficiency of the tokenization procedures TKR2 and TKR2a. In Table 6.1 we summarize the results, which are described as below:

   - **Run1** denotes the average time required to generate one token, including the

	IND-TKR	IND-TKR-CV	IND-TKR-KEY
TKR1	$\surd$	$\surd$	
TKR2[f_k]	$\surd$		
TKR2[TR]	$\surd$		$\surd$
TKR2a[$f_k, \mathbf{E}$]	$\surd$	$\surd$	
TKR2a[TR, $\mathbf{E}$]	$\surd$	$\surd$	$\surd$

Table 6.2: Summary of Security

insertion in the card-vault. But here primary keys in the card-vault relations are not specified, i.e., this run does not do any uniqueness test. The average is computed over 1,000,000 tokens.

- **Run2** denotes the scenario where the primary keys are specified, i.e., the database checks for the uniqueness. As it is obvious, in this case the time required to tokenize (including the insertions in the card-vault) would increase with the current size of the card-vault. To measure this difference we divided this run into five different runs which we call **Run2a**, **Run2b**,$\cdots$, **Run2e**. For **Run2a** we started with an empty card-vault, and generated 1,000,000 tokens. In **Run2b** we started with a card-vault already containing 1,000,000 tokens, and we generated 1,000,000 more tokens. Similarly, in runs **Run2c**, **Run2d** and **Run2e**, we started with a card-vault containing 2,000,000; 3,000,000 and 4,000,000 tokens, respectively. In each run we generated 1,000,000 more tokens. The Table 6.1 shows the average time required for generating one token for each scenario.

The basic component for both TKR2 and TKR2a is the procedure $\mathsf{RN}^{\mathcal{T}}$, as mentioned, a call to $\mathsf{RN}^{\mathcal{T}}$, costs only 0.1 micro seconds. But the times reported in Table 6.1 (which are in milliseconds) are more realistic, and it shows that the database insertions dominate the cost of tokenization. Thus, further optimization in this regard may be possible. But, still our experimental results confirm that the schemes proposed in this work can be implemented and used in a real tokenization environment.

6.5 Discussions

SECURITY. The security properties of the various schemes as stated in the previous security theorems are summarized in Table 6.2. 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 pseudorandom 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 [13] becomes vacuous when the number of queries exceeds $2^{\#\mathcal{T}/4}$, whereas the scheme in [53] can tolerate up to $2^{\#\mathcal{T}-\epsilon}$ queries where ϵ is inversely related to the number of rounds in the construction. Some recent constructions in [36, 64] achieve much better bounds, specially in [64] 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 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 $\mathsf{RN}[f_k]$ is instantiated as in Figure 6.1, and in turn f_k is constructed using a block cipher, then using Proposition 6.1 and Theorem 5.2, for any IND-TKR adversary $\mathcal{A}$ who asks at most q queries, we have

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

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 6.2.1, $m = L/n$, 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 $\mathbf{Adv}_{\mathsf{TKR2a}[f_k]}^{\text{ind-tkr-cv}}(\mathcal{A})$, 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 $\mathsf{TKR2}[f_k]$ and $\mathsf{TKR2a}[f_k]$ 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).

6.6 Summary

As we mentioned in the previous Chapter, in the guidelines of the PCI SSC, regarding the tokenization paradigm, it is not clear which is the best way to generate the tokens. These guidelines neither explain which cryptographic objects should be used nor how to use them. Here we answered these questions. We showed how to instantiate $\mathsf{RN}^{\mathcal{T}}[k]$ using specific cryptographic objects, to generate the tokens. We also proposed two ways to implement the encryption scheme $\mathbf{E}$ to protect the data in the card-vault. In addition we also analyzed the security of these instantiations. Finally, seeing the results that we obtained in our experiments, we have shown the feasibility of our constructions.

Chapter 7

Conclusions and Future Work

The results that we described in the previous chapters constitute an effort to formulate suitable security notions for two different practical scenarios. We have not only proposed new notions of security, but also novel encryption schemes suitable for the applications, which are secure in the sense of the proposed notions of security. As it is always the case, these encryption schemes can be improved in different ways. In the following paragraphs, we provide a summary of the contributions of this thesis. Also we point out some limitations of the proposed schemes and highlight some directions by which the schemes can be improved. The trust of this thesis was to study some notions of security which may be applicable to some practical security problems. We addressed two application areas, namely the problem of profiling adversaries and the problem of encrypting credit card numbers. Needless to say there are many other scenarios where a strong encryption scheme reduces the usability. We discuss some application areas which we do not consider in this thesis but we plan to study in the near future.

7.1 Summary of Contributions

Given that we analyzed two different scenarios: protecting communications against profilers and the study of the tokenization paradigm, we will also describe our contributions in two parts.

7.1.1 Securing Information from Profilers

1. We studied the exact requirements for an encryption algorithm to be secure only against profilers and gave a precise notion of security for such schemes. We called

this as the PROF-EAV (security against eavesdropping profilers) notion of security. This definition follows the paradigm of concrete provable security but is completely new. Moreover, we were able to show that the PROF-EAV security is strictly weaker than the more popular IND-CPA security notion.

2. We proposed a full protocol for secure communication against non-human profiling adversaries. Our protocol neither requires a key exchange nor a public key infrastructure. The main components of this protocol are a PROF-EAV secure encryption scheme, a CAPTCHA puzzle generator and a secret sharing scheme. We were able to prove security of the protocol with mild assumptions on the encryption scheme and the CAPTCHA generator. Among other things, our protocol demonstrates a novel security application for CAPTCHAs which were traditionally used only to differentiate humans from machines.

3. The proposed protocol uses a PROF-EAV secure encryption scheme, and as PROF-EAV security is weaker than IND-CPA security, hence any traditional encryption scheme which provides IND-CPA security can be used in the protocol. But, it would be more interesting to construct a PROF-EAV encryption scheme. We achieved this by using a less known scheme by Rivest which is known as Chaffing and Winnowing (CW) [65]. Our scheme uses some novel ideas of document classification, and we experimentally demonstrated that the scheme can really fool classifiers (profilers). We also provided a preliminary theoretical analysis of our scheme.

7.1.2 Cryptographic Treatment of Tokenization

Tokenization systems are currently in use, but a formal cryptographic study of such systems does not exist in the current literature. We initiated a study in this direction, and contributed in the following ways:

1. We formally fixed a general syntax for tokenization systems and specified the requirements for its various components.

2. We formally defined security of tokenization systems and formulated three different security notions called IND-TKR, IND-TKR-CV, and IND-TKR-KEY. These three definitions are based on three different realistic threat models. We amply discussed the adequacy of these new notions of security in practical scenarios.

3. We introduced some constructions of tokenization systems, and proved their security in the proposed security models. We proposed three generic constructions

namely TKR1, TKR2 and TKR2a and discussed 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 format preserving encryption. Both TKR2 and TKR2a use off the shelf cryptographic primitives, in particular we showed how to instantiate them using ordinary block ciphers, stream ciphers supporting initialization vectors (IV) and physical random number generators. We also proved security of our constructions in the proposed security models.

7.2 Future Work

Here we note down some ways in which the proposed schemes can be improved and extended, and we also point out some other application areas where weak security notions are applicable.

7.2.1 Profiling Adversaries

In Chapter 3, we proposed the protocol $\mathbb{P}'$ which uses multiple CAPTCHAs and thus gives the user multiple chances to recover the key. This functionality is important, as a human user may sometimes fail to solve a CAPTCHA. Actually, we recommended a specific instantiation of $\mathbb{P}'$, where a human has to solve 2 out of 5 CAPTCHAs to decrypt a message. However this solution, though correct, is inconvenient. Imagine a user who needs to read a large quantity of emails in a single day, he has to solve two CAPCHAs per email that he reads. A better solution would be that the user solves only one CAPTCHA, and only if the user fails, then the system automatically generates another CAPTCHA and so on. This is in line with the normal usage of CAPTCHAs and may be a bit more convenient than our solution in $\mathbb{P}'$. The basic idea involved in implementing this functionality, would be to have multiple CAPTCHAs, each of whose solution can be mapped to a single key K. We give some initial ideas in this direction next.

One option is using the roots of the equation $x^r \equiv a \bmod n$, where a is some representation of k in our protocol $\mathbb{P}$. We can convert the values of the roots x_i into a CAPTCHA, then if we assume that the equation has r roots, there are r possible values that generate the same value a. A possibility is that $r = 2$, then we have the equation $x^2 \equiv a \bmod n$, where n is the product of two distinct prime numbers p and q, i.e. $n = pq$. We already know that this equation has four possible solutions, and it is possible to choose p, q and a in such a way that these solutions are distinct. Then each of these four square roots x_i, where $1 \leq i \leq 4$, can be converted (via some encoding) into a CAPTCHA. The receiver must solve the CAPTCHA, convert it into the original value x_i and compute $x_i^2 \bmod n$ to obtain a. If the receiver fails to solve the first CAPTCHA, still he would have three more chances.

Another option is to consider an error-correcting code. Error-correcting codes are used to detect and correct the errors that occur in a communication channel. Depending on the type of code that is used, it is possible to detect and correct a specific number of errors. Let us consider a linear code. In a linear code if we consider a message m as a binary string of length ℓ, then we can transform it into a *code word* m' of n bits, where $n > \ell$. Thus we will have a set of 2^ℓ code words. The $n - \ell$ denominated *redundancy bits* help us to detect possible errors in the communication. Now let us explain how we can apply a linear code to solve our problem. If we consider the string k in our protocol $\mathbb{P}$, we can convert it into a code word k' of length n by applying a linear code. Then we can add some error e_i to k', and thus obtain $s_i = k' + e_i$, $1 \leq i \leq r$. These s_i can be used to generate the CAPTCHAs. The receiver must solve the CAPTCHA, and decode s_i to obtain k and use this k as input to the hash function to generate the encryption key K. Again if a user fails in solving a CAPTCHA, the system can offer $r - 1$ other options. Here r depends on the number distinct errors the code can tolerate. Naturally, we still need to determine the specific code that we can apply.

As we mentioned in Chapter 3, another important issue with our protocol, which uses CAPTCHA, is the key length. The size of the key space is around 2^{48}, which is not so convenient. It would be interesting to find a mechanism that let us increase the size of key space.

7.2.2 Tokenization Systems

The security notions of tokenization systems that we propose in Chapter 5 does not take into account a malicious tokenizer. Consider a scenario, where a merchant receives a token from a tokenizer and later the tokenizer denies to detokenize the token by saying that this token was not generated by it. Our security notions do not consider this scenario. But the definitions can be strengthened to incorporate security against such malicious tokenizers. A secure tokenization system which can provide such kind of security may have to use a digital signature in a suitable way. We plan to explore this option in near future.

The novelty of our tokenization systems is that they by-pass the need for format preserving encryption. Format preserving encryption or, specifically the problem of "small domain encryption" is largely open. None of the few solutions available in the literature are efficient, and most have an unacceptable security bound. In the near future we want to take up this problem to design encryption schemes for small messages which are efficient and provides acceptable security in provable terms. There is an increased need for such encryption schemes. As there are many identifiers similar to credit card numbers, for example social security numbers, patient identification numbers, etc. which need to be encrypted.

7.3 Other Scenarios where Weak Notions of Security are Applicable

In this thesis we discussed just two practical scenarios where the security usability tradeoff is important, but there are many more problems of practical interest where strong cryptography cannot be directly applied. In this section we discuss two such scenarios.

7.3.1 Deduplication

Recently applications such as Dropbox, Google Drive, Bitcasa among others, have gained popularity. Not only big companies but common users have started to store their information in the cloud. These applications offer remote backup services to their users. An important issue for the companies that offer these services is how to save storage space and network bandwidth. Privacy aware users generally encrypt the information that they upload to a cloud, this can lead to an increase in the amount of storage required in the servers. We explain it with an example: Consider that a user $\mathbb{A}$ created a video M and shared it with another user $\mathbb{B}$. Both $\mathbb{A}$ and $\mathbb{B}$ uploaded the video M to a cloud server which provides the service of storage. To maintain confidentiality of the data, both $\mathbb{A}$ and $\mathbb{B}$ encrypted the video M before uploading it to the cloud. As $\mathbb{A}$ and $\mathbb{B}$ would have different encryption keys, hence the encryption of the same message M, would be different and there would be no way for the server to know that the two things that it got from the two users $\mathbb{A}$ and $\mathbb{B}$ correspond to the same message. *Secure deduplication* is an encryption procedure that enables the server to detect duplicates of the same information and thus it stores only one copy of a message. It is obvious that the main aim of deduplication is to save storage space at the server side at the cost of sacrificing some security.

A first solution to this problem came from Douceur et. al [27] who proposed a cryptographic scheme to avoid duplicating files, this cryptographic scheme is called *convergent encryption*, and it works as follows. The first time that a user wants to upload an encrypted file M, he gets the key from the message $K \leftarrow H(M)$, where H is a public hash function. Then using this key K and a deterministic symmetric encryption scheme E, he enciphers M as $C \leftarrow E_K(M)$. It is important to note that E must be deterministic, to obtain the same ciphertext for equal messages. This property makes possible to detect duplicates, even when they are encrypted. Finally a *tag*, T of the ciphertext C is computed as $T = H(C)$ and it is sent to the server. This tag is an index, used to store the encrypted file C and also it determines if an encrypted file has been already stored. When a second user wants to upload the same file, the user needs to perform the same steps already explained above, and when he sends the tag T to the server, it will know whether the file M has already been uploaded by some other user.

Recently a formal cryptographic study of deduplication systems was initiated in [7], where strict security definitions for deduplication schemes were proposed. After this, several schemes were proposed [1, 6, 22]. Unfortunately some of the solutions, proposed in these papers, require a complicated machinery, which is not adequate given the practicality of the problem. Some other schemes make an attempt to reduce the computation involved but they introduce security problems. For example, let us consider the following variation in the original scheme. Instead of calculating the tag T from the ciphertext, it is computed as $T = H(K)$. This introduces a security problem called *duplicate faking attack*. In this attack, an adversary $\mathcal{A}$ chooses two different messages M and M', derives the key $K = H(M)$ and $C' = E_K(M')$ and $T = H(K)$. When a second user wants to store the file M, he does the following $K = H(M)$, $C = E_K(M)$ and $T = H(K)$. Since the server determines if a file is duplicated by seeing only the tag T, the server will not store the file of the second user. Later, when this second user tries to recover his file, he will obtain $M' \neq M$. We believe that to solve this problem, it would be better to develop a scheme to compute the ciphertext and the tag in one pass.

Another interesting problem in *secure deduplication* is related with the deduplication of files which are not equal but similar. A preliminary solution is based on error-correcting codes, see [22]. However we have the intuition that other mechanisms such as *locality-preserving hashing (LSH)* [3] can be employed to this problem. A hash function of this kind works as follows, if two points p and q in a plane are neighbours, then the probability that $h(p) = h(q)$ is high, where the neighbourhood is determined by the distance between these two points. If we could find a way to transform two files or documents into points and determine a measure to establish the similarity of the documents, then it will be possible to establish that they are similar. Once that we do this, then we can encrypt only the difference between the two files, and instead of storing the whole file, we just need to store this difference.

7.3.2 Searchable Encryption

Searchable encryption denotes the set of the encryption schemes which allows several kinds of searches within encrypted data. The two goals of this kind of encryption are contradictory, as a good encryption scheme should destroy all structure present in the data and make it look random, whereas the more structured the data is more efficient search on it can be performed. Thus, it is evident that a weak notion of security for such encryption schemes is required. There have been some theoretical work describing the proper security notion for such encryption schemes, for example see [20, 33]. Also there is a variety of applications where such schemes can be useful. We discuss two application areas:

1. **Encrypting Outsourced Databases:** With the advent of cloud computing, many corporations outsource their databases to third party server. In most cases the data owner would not like to reveal the data to the server but at the same time he would want the server to answer the specific queries that he poses on the outsourced data. This, is a classic example where searchable encryption may be applied. There have been many works to date to address this problem [62], but still this problem is largely open as the proposed schemes have many limitations concerning both security and functionality.

2. **Encrypted Email Search:** Nowadays it is very common that we use a web mail service. As a part of this service we can make searches through our emails by keywords and dates, but how can we be sure that the search results are correct? Given that the web email service lets us maintain a huge number of emails, it is difficult for a user to know if the search results are correct. This, problem falls in the domain of authentication. An initial solution to this problem has been proposed in [57]. Their protocol allows a user to verify that the list of messages in the search results is correct, but it does not allow to verify if the emails that are not queried are still present at the email server. The problem becomes more interesting and complex if we consider that the emails are encrypted.

These problems and the application areas fall directly within the scope of our work, and we aim to study them in near future.

Publications of the Author Related to the Thesis

1. S. Díaz-Santiago and D. Chakraborty. On Securing Communications from Profilers. In P. Samarati, W. Lou and J. Zhou, editors, *SECRYPT 2012-Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE-The International Joint Conference on e-Business and Telecommunications*, pages 154-162. SciTePress, 2012.

2. S. Díaz-Santiago and D. Chakraborty. Encryption Schemes Secure against Profiling Adversaries. In M.S. Obaidat and J. Filipe, editors, *E-Business and Telecommunications*, volume 455 of *Communications in Computer and Information Science*, pages 172-191. Springer Berlin Heidelberg, 2014.

3. S. Díaz-Santiago, L. M. Rodríguez-Henríquez and D. Chakraborty. A Cryptographic Study of Tokenization Systems. In M. S. Obaidat, A. Holzinger and P. Samarati, editors, *SECRYPT 2014-Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, pages 393-398. SciTePress, 2014.

4. S. Díaz-Santiago, L. M. Rodríguez-Henríquez and D. Chakraborty. A Cryptographic Study of Tokenization Systems. *IACR Cryptology ePrint Archive, 2014:602, 2014.*

Bibliography

[1] M. Abadi, D. Boneh, I. Mironov, A. Raghunathan, and G. Segev. Message-locked encryption for lock-dependent messages. In Canetti and Garay [15], pages 374–391.

[2] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In D. Naccache, editor, *CT-RSA*, volume 2020 of *Lecture Notes in Computer Science*, pages 143–158. Springer, 2001.

[3] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[4] M. Bellare and A. Boldyreva. The security of chaffing and winnowing. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 517–530. Springer, 2000.

[5] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In N. Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1996.

[6] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. *IACR Cryptology ePrint Archive*, 2013:429, 2013.

[7] M. Bellare, S. Keelveedhi, and T. Ristenpart. Message-locked encryption and secure deduplication. In T. Johansson and P. Q. Nguyen, editors, *EUROCRYPT*, volume 7881 of *Lecture Notes in Computer Science*, pages 296–312. Springer, 2013.

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

[9] M. Bellare and P. Rogaway. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

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

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

[12] J. Black, S. Halevi, H. Krawczyk, T. Krovetz, and P. Rogaway. Umac: Fast and secure message authentication. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 216–233. Springer, 1999.

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

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

[15] R. Canetti and J. A. Garay, editors. *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*. Springer, 2013.

[16] R. Canetti, S. Halevi, and M. Steiner. Hardness amplification of weakly verifiable puzzles. In J. Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 17–33. Springer, 2005.

[17] CardHub. Number of credit cards and credit card holders, 2012. Available at `http://www.cardhub.com/edu/number-of-credit-cards/`.

[18] R. Clayton and G. Danezis. Chaffinch: Confidentiality in the face of legal threats. In F. A. P. Petitcolas, editor, *Information Hiding*, volume 2578 of *Lecture Notes in Computer Science*, pages 70–86. Springer, 2002.

[19] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

[20] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, 19(5):895–934, 2011.

[21] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 2002.

[22] G. I. Davida and Y. Frankel. Efficient encryption and storage of close distance messages with applications to cloud storage. In D. Naccache, editor, *Cryptography and Security*, volume 6805 of *Lecture Notes in Computer Science*, pages 465–473. Springer, 2012.

[23] S. Diaz-Santiago and D. Chakraborty. On securing communication from profilers. In P. Samarati, W. Lou, and J. Zhou, editors, *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 154–162. SciTePress, 2012.

[24] S. Diaz-Santiago and D. Chakraborty. Encryption schemes secure against profiling adversaries. In M. S. Obaidat and J. Filipe, editors, *E-Business and Telecommunications*, volume 455 of *Communications in Computer and Information Science*, pages 172–191. Springer Berlin Heidelberg, 2014.

[25] S. Diaz-Santiago, L. M. Rodriguez-Henriquez, and D. Chakraborty. A cryptographic study of tokenization systems. *IACR Cryptology ePrint Archive*, 2014:602, 2014.

[26] S. Diaz-Santiago, L. M. Rodriguez-Henriquez, and D. Chakraborty. A cryptographic study on tokenization systems. In M. S. Obaidat, A. Holzinger, and P. Samarati, editors, *SECRYPT 2014 - Proceedings of the 11th International Conference on Security and Cryptography, Vienna, Austria, 28-30 August, 2014*, pages 393–398. SciTePress, 2014.

[27] J. R. Douceur, A. Adya, W. J. Bolosky, D. Simon, and M. Theimer. Reclaiming space from duplicate files in a serverless distributed file system. In *ICDCS*, pages 617–624, 2002.

[28] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[29] M. Dürmuth. Useful password hashing: how to waste computing cycles with style. In M. E. Zurko, K. Beznosov, T. Whalen, and T. Longstaff, editors, *NSPW*, pages 31–40. ACM, 2013.

[30] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In E. F. Brickell, editor, *CRYPTO*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. Springer, 1992.

[31] S. Dziembowski. How to pair with a human. In J. A. Garay and R. D. Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 200–218. Springer, 2010.

[32] O. Goldreich. *The Foundations of Cryptography - Volume 1, Basic Applications*. Cambridge University Press, 2001.

[33] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553. Springer, 2013.

[34] S. Goldwasser and S. Micali. Probabilistic encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[35] P. Golle and A. Farahat. Defending email communication against profiling attacks. In V. Atluri, P. F. Syverson, and S. D. C. di Vimercati, editors, *WPES*, pages 39–40. ACM, 2004.

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

[37] R. Impagliazzo, R. Jaiswal, and V. Kabanets. Chernoff-type direct product theorems. *J. Cryptology*, 22(1):75–92, 2009.

[38] R. Impagliazzo and D. Zuckerman. How to recycle random bits. In *30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, North Carolina, USA, 30 October - 1 November 1989*, pages 248–253. IEEE Computer Society, 1989.

[39] ISO/IEC 7812-1. Identification cards-identification of issuers-part 1: Numbering system, 2006.

[40] ISO/IEC 9797:1989. *Data cryptographic techniques-Data integrity mechanism using a cryptographic check function employing a block cipher algorithm*, 1989.

[41] C. S. Jutla. Almost optimal bounds for direct product threshold theorem. In D. Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2010.

[42] S. P. Karanam. Tiny true random number generator. Master's thesis, George Mason University, Department of Electrical and Computer Engineering, 2006.

[43] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/ CRC, 2008.

[44] W. Killmann and W. Schindler. A design for a physical RNG with robust entropy estimators. In E. Oswald and P. Rohatgi, editors, *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, volume 5154 of *Lecture Notes in Computer Science*, pages 146–163. Springer, 2008.

[45] N. Koblitz and A. Menezes. Another Look at "Provable Security". II. *IACR Cryptology ePrint Archive*, 2006:229, 2006.

[46] N. Koblitz and A. Menezes. Another Look at "Provable Security". *J. Cryptology*, 20(1):3–37, 2007.

[47] X. Lai and J. L. Massey. A proposal for a new block encryption standard. In *EUROCRYPT*, pages 389–404, 1990.

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

[49] C. Mancillas-López. *Studies on Disk Encryption*. PhD thesis, CINVESTAV-IPN, 2013.

[50] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[51] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering, 1996. `http://www.cs.cmu.edu/~mccallum/bow`.

[52] D. Mladenic and M. Grobelnik. Feature selection for unbalanced class distribution and naive bayes. In I. Bratko and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning (ICML 1999), Bled, Slovenia, June 27 - 30, 1999*, pages 258–267. Morgan Kaufmann, 1999.

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

[54] M. Naor. Verification of a human in the loop or identification via the turing test, 1997. http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.pdf.

[55] NIST. Data Encryption Standard. Federal Information Processing Standard (FIPS) 46-3, 1999.

[56] NYT. Congress begins deep packet inspection of internet providers, 2009. Available at: http://bits.blogs.nytimes.com/2009/04/24/congress-begins-deep-packet-inspection-of-internet-providers/.

[57] O. Ohrimenko, H. Reynolds, and R. Tamassia. Authenticating email search results. In A. Jøsang, P. Samarati, and M. Petrocchi, editors, *Security and Trust Management - 8th International Workshop, STM 2012, Pisa, Italy, September 13-14, 2012, Revised Selected Papers*, volume 7783 of *Lecture Notes in Computer Science*, pages 225–240. Springer, 2012.

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

[59] PCI Security Standards Council. Information supplement: PCI DSS tokenization guidelines, 2011. Available at https://www.pcisecuritystandards.org/documents/Tokenization_Guidelines_Info_Supplement.pdf.

[60] C. Percival. Stronger key derivation via sequential memory-hard function. BSDCan'09, pages 1-19, 2009.

[61] B. Pinkas and T. Sander. Securing passwords against dictionary attacks. In V. Atluri, editor, *ACM Conference on Computer and Communications Security*, pages 161–170. ACM, 2002.

[62] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: processing queries on an encrypted database. *Commun. ACM*, 55(9):103–111, 2012.

[63] N. Provos and D. Mazières. A future-adaptable password scheme. In *USENIX Annual Technical Conference, FREENIX Track*, pages 81–91. USENIX, 1999.

[64] T. Ristenpart and S. Yilek. The mix-and-cut shuffle: Small-domain encryption secure against n queries. In Canetti and Garay [15], pages 392–409.

[65] R. L. Rivest. Chaffing and winnowing: Confidentiality without encryption, 1998. http://people.csail.mit.edu/rivest/Chaffing.txt.

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

[67] P. Rogaway. Nonce-based symmetric encryption. In B. K. Roy and W. Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 348–359. Springer, 2004.

[68] P. Rogaway and D. Coppersmith. A software-optimized encryption algorithm. *J. Cryptology*, 11(4):273–287, 1998.

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

[70] Securosis White Paper. Tokenization guidance: How to reduce PCI compliance costs, 2011. Available at `http://gateway.elavon.com/documents/Tokenization_Guidelines_White_Paper.pdf`.

[71] Securosis White Paper. Tokenization vs. encryption: Options for compliance, 2011. Available at `https://securosis.com/research/publication/tokenization-vs.-encryption-options-for-compliance`.

[72] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[73] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal, Vol 28, pp. 656âĂŞ715*, Oktober 1949.

[74] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

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

[76] B. Sunar, W. J. Martin, and D. R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Trans. Computers*, 56(1):109–119, 2007.

[77] V. Toubiana, A. Narayanan, D. Boneh, H. Nissenbaum, and S. Barocas. Privacy preserving targeted advertising. In *Proceedings of annual network and distributed systems security symposium*, 2010. `http://www.isoc.org/isoc/conferences/ndss/10/pdf/05.pdf`.

[78] Voltage Security White paper. Payment security solution - processor edition, 2012. Available at `http://www.voltage.com/wp-content/uploads/Voltage_White_Paper_SecureData_PaymentsProcessorEdition.pdf`.

[79] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 294–311. Springer, 2003.