A chameleon encryption scheme resistant to known-plaintext attack

Article · January 2010

DOI: 10.1145/1866870.1866876

Ee-Chien Chang
National University of Singapore
131 PUBLICATIONS **2,193** CITATIONS
SEE PROFILE

Chengfang Fang
Huawei Technologies
16 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Jia Xu
NUS-SingTel Cyber Security R&D Lab
20 PUBLICATIONS **375** CITATIONS
SEE PROFILE

A Chameleon Encryption Scheme Resistant to Known-Plaintext Attack

Ee-Chien Chang Chengfang Fang Jia Xu
Department of Computer Science
National University of Singapore
Republic of Singapore
{changec,c.fang,xujia}@comp.nus.edu.sg

ABSTRACT

From a ciphertext and a secret key assigned to a user, the decryption of a *Chameleon encryption* scheme produces a message which is the plaintext embedded with a watermark associated to the user. Most existing constructions of Chameleon encryption scheme are LUT (lookup table)-based, where a secret LUT plays the role of the master key and each user has a noisy version of the secret LUT. LUT-based methods have the limitation that the secrecy of the master key, under known-plaintext attack (KPA), relies on the difficulty in solving large linear system. In other words, with some knowledge of the plaintext, a dishonest user is able to derive the LUT, or an approximation of the LUT by solving a linear system. Resistance to such attack is crucial in the context of multimedia encryption since multimedia objects inherently contain high redundancies. Furthermore, for efficiency in decryption, the underlying linear system is likely to be sparse or not overly large, and hence can be solved using reasonable computing resource. In our experiment, a desktop PC is able to find a LUT (with 2^{16} entries) within 2 hours. We propose a scheme that is resistant to KPA. The core of the scheme is a MUTABLE-PRNG (Pseudo Random Number Generator) whereby different but similar sequences are generated from related seeds. We generate such sequence from multiple pseudo random sequences based on majority-vote, and enhance its performance using error-correcting code. The proposed scheme is very simple and it is easy to show that it is resistant to KPA under reasonable cryptographic assumptions. However, it is not clear how much information on the original plaintext is leaked from the watermarked copies. We analyze the scheme and quantify the information loss using average conditional entropy.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*

General Terms

Algorithm, Security

Keywords

Client-side watermark embedding, Chameleon Encryption, Known-Plaintext Attack, MUTABLE-PRNG

1. INTRODUCTION

With the boom of Internet, digital right protection becomes more and more crucial and challenging. Apart from legal penalties, many techniques are devised to resolve or mitigate the threat of pirate. Watermarking is an important technique among them. In a typical watermarking application, the content distributor first embeds watermarks into different copies of the multimedia data on the server side, and then distributes the watermarked copies to different subscribed users. Some scenarios require client-side watermarking, whereby the watermark is only embedded on the client side. One way to achieve secure client-side embedding is through trusted hardware [20, 21]. Each subscribed user will receive a tamper-proof physical decoder which decrypts and embeds watermark into the streamed data on the fly. However, securing hardware is costly. An alternative is Chameleon encryption. This framework consists of two phases: in the key setup phase, through a secure point-to-point channel, each user u receives a personalized decryption key $\mathcal{K}_u$ and the content distributor receives a master encryption key $\mathcal{K}$, from a trusted key generator. In the service phase, only an insecure broadcast channel is required. All data will be encrypted under the encryption key $\mathcal{K}$ by the distributor before being broadcasted to all subscribed users. Each subscribed user u can decrypt the received data using the assigned key $\mathcal{K}_u$. Instead of getting the original data, the user obtains a watermarked copy that is different from but perceptually similar to the original.

Most existing Chameleon encryption schemes [4, 6, 2] are lookup table(LUT)-based: A secret LUT is used during encryption, whereas each user is assigned a distinct noisy version of LUT for decryption. There are two main parameters, L and S, where L denotes the number of entries in LUT, and S is the number of LUT entries selected to encrypt/decrypt a single character. The LUT-based encryption has the limitation that the secrecy of the LUT relies on the difficulty in solving a large linear system under known-plaintext attack. Given plaintext-ciphertext pairs, an attacker is able to derive the master key, by solving a linear system. It is easy to extend the attack to a scenario where the attacker has a decryption key and only partial knowledge of the plaintext. In this scenario, by finding the solution to a linear least square system, the attacker can obtain an "approximate" master key that is significantly close to the original, and closer than any user's decryption key. Such attacks need to be addressed since multimedia data inherently contain high redundancies. Although compression may help to reduce redundancies, the watermark embedding typically has to be performed before the lossless compression. For instance, for JPEG, the watermark can be embedded in the DCT domain, but not after the lossless compression. Using larger LUT would force the attacker to solve a larger linear system and could deter attackers with limited resources. However, LUT is part of the key and thus it is desirable to keep it small. Furthermore, the parameter S is desired to be small for efficiency in encryption/decryption. This leads to a sparse linear system that can be handled using reasonable computing resource. In our experiment, for LUT with size $L = 2^{16}$, where each entry is a non-negative integer smaller than 2^{16} and $S = 80$, a "good" approximation of LUT can be found using a desktop PC within 2 hours.

We propose a new Chameleon encryption scheme which is simple but secure against known-plaintext attack. The scheme is a one-time pad cipher equipped with a Mutable-PRNG F: Given an input $\mathcal{S}$, one can find another short input $\mathcal{S}_u$, such that the Hamming distance between $\mathsf{F}(\mathcal{S})$ and $\mathsf{F}(\mathcal{S}_u)$ is small. Let $\mathcal{S}$ and $\mathcal{S}_u$ be the encryption key and decryption key respectively. One-time pad cipher is adopted to mask (unmask, respectively) the plaintext (ciphertext, respectively) with the generated pseudorandom bit-stream $\mathsf{F}(\mathcal{S})$ ($\mathsf{F}(\mathcal{S}_u)$, respectively). As a result, $\mathsf{F}(\mathcal{S}) \oplus \mathsf{F}(\mathcal{S}_u)$ is embedded into the decrypted data as watermark during the decryption, where $\oplus$ is the XOR operator.

To generate the sequence $\mathsf{F}(\mathcal{S})$, we use majority vote on multiple pseudorandom sequences. Specifically, $\mathcal{S}$ is a set of n seeds from which n pseudorandom sequences are generated. For each location i, the i-th bit of $\mathsf{F}(\mathcal{S})$ is set to the majority bit among all i-th bits in the n pseudorandom sequences. Similarly, we can generate the sequence $\mathsf{F}(\mathcal{S}_u)$ from $\mathcal{S}_u$, where $\mathcal{S}_u$ is a subset of $\mathcal{S}$. The sequence $\mathsf{F}(\mathcal{S}_u)$ generated from $\mathcal{S}_u$ correlates with, but is not identical to, the sequence $\mathsf{F}(\mathcal{S})$ generated from $\mathcal{S}$. To increase the correlations, we employ error correcting code (ECC).

The proposed scheme is very simple, and it is easy to show that the master key is secure under KPA, with the assumptions that each pseudorandom sequence is generated by some cryptographically secure PRNG. However, the same as LUT-based methods, our scheme suffers from collusion attack, in the sense that a few collusive users can easily derive the master key or a good approximation of the master key. In this paper, we do not address collusion attack. It is interesting to further investigate whether cryptographic techniques can be incorporated to "hide" the computation of majority and thus prevent collusion.

Our scheme reveals extra information of the original data besides the watermarked copy, due to information provided by the ECC and the internal states in the computation of majority bits. We analyze and quantify such information loss using the notion of average conditional entropy.

Contribution.

1. While it is already known that LUT-based Chameleon encryption schemes are vulnerable under KPA, previous Chameleon encryption schemes rely on large LUT to deter attack. We argue that for multimedia data, it is crucial to be secure under KPA and point out that the LUT can be easily found since the underlying linear system is likely to be sparse. Our experimental studies show that a rather large LUT (with $L = 2^{16}$ entries and $S = 80$) can be found using a desktop PC within 2 hours.
2. We propose a new Chameleon encryption scheme based on a specially designed Mutable-PRNG (Section 5). This Mutable-PRNG is in turn constructed using majority voting and a cryptographically secure PRNG. We show that this scheme is KPA-secure (Theorem 1) and give an upper bound on the entropy loss due to ECC and the internal states in the computation of majority voting.

Organization.

The rest of this paper is organized as follows: Section 2 discusses the related work and Section 3 defines the problem framework and security concerns. The known-plaintext attack on previous Chameleon encryption schemes is given in Section 4. We present the new Chameleon encryption scheme in Section 5 and analyze its security in Section 6. Section 7 presents a variant version of our proposed scheme and Section 8 closes this paper.

2. RELATED WORKS

Client side watermark embedding allows the distributor to distribute a unique copy of encrypted data, and the personalized watermark will be added into the data on the client side in a secure way. The hardware solution requires each legitimate user to have a dedicated secure hardware to embed the watermark, thus incurs a huge deployment cost. A secure software solution is much more preferable. Anderson *et al.* [4] proposed Chameleon Encryption scheme which binds decryption and watermark embedding together so that adversary cannot separate them. Adelsbach *et al.* [2] and Celik *et al.* [6] improved Chameleon Encryption. Adelsbach *et al.* briefly summarized previous works on joint fingerprinting and decryption [17, 13, 15]. Lemma *et al.* [14] proposed a similar method such that the content owner distributes personalized help data to each user for every decryption.

3. DEFINITIONS AND NOTATIONS

There are two different roles involved: a content distributor $\mathcal{D}$ and a set $\mathcal{U}$ of users. At the very beginning, $\mathcal{D}$ generates[1] a master encryption key $\mathcal{K}$ for himself/herself, and a personalized decryption key $\mathcal{K}_u$ for each user $u \in \mathcal{U}$. $\mathcal{D}$ delivers the decryption key $\mathcal{K}_u$ to user $u \in \mathcal{U}$ securely. After this setup, $\mathcal{D}$ can distribute his/her data or messages (e.g. digital movies, music, TV service) to all users (e.g. those who have paid $\mathcal{D}$ for such multimedia data) in $\mathcal{U}$ in this way:

1. $\mathcal{D}$ encrypts the message x using the encryption key $\mathcal{K}$:

[1]For simplicity, we assume the content distributor $\mathcal{D}$ is also the key generator.

$c \leftarrow \mathsf{Enc}(x;\mathcal{K})$, and distributes c to all users in $\mathcal{U}$ via (insecure) broadcast.

2. Each user $u \in \mathcal{U}$ decrypts c using the decryption key $\mathcal{K}_u$: $\widetilde{x}_u \leftarrow \mathsf{Dec}(c;\mathcal{K}_u)$.

Each decrypted message $\widetilde{x}_u$ is a (distinct) watermarked copy of the original message x. The watermark is embedded into $\widetilde{x}_u$ just in the same process of decryption on the client-side. The above model is a simplified "Joint Fingerprinting and Decryption" framework [2].

Security.

In this paper, the adversary is some user $u \in \mathcal{U}$ with a decryption key $\mathcal{K}_u$. We are concerned about the following security property:

1. Secrecy of master encryption key: A user $u \in \mathcal{U}$ wants to find a good approximation $\widetilde{\mathcal{K}}$ of the master encryption key $\mathcal{K}$, which is much closer to $\mathcal{K}$ than the decryption key $\mathcal{K}_u$. That is, $\mathsf{Dist}(\widetilde{\mathcal{K}},\mathcal{K}) \le \tau \cdot \mathsf{Dist}(\mathcal{K}_u,\mathcal{K})$ for some threshold[2] $\tau \in [0,1)$, where $\mathsf{Dist}(\cdot,\cdot)$ is some appropriate distance function. In Section 4.2.2, we choose L^2-Norm as the distance function. We consider two forms of known-plaintext attack. In the first form, the adversary has a decryption key and some plaintext-ciphertext pairs. We investigate whether the adversary can find the master encryption key with such information. In the second form, the adversary has a decryption key, some ciphertexts and knows the distribution of their corresponding plaintexts.
2. Robustness of watermark: For any watermarking scheme, the watermarked message inevitably reveals some information about the original. In joint decryption and watermarking framework, the combination of ciphertext C and user decryption key $\mathcal{K}_u$ may potentially reveal more information. We require the difference between the two, i.e. the extra information revealed by $(C,\mathcal{K}_u)$ to be small. However, it is very difficult to formulize this notion in a rigorous way. In our paper, as an approximation, we require $\mathbf{H}(X|X_u) - \mathbf{H}(X|C,\mathcal{K}_u)$ to be small, where $\mathbf{H}(\cdot|\cdot)$ denotes the *average conditional entropy* [19], the plaintext X is sampled from the plaintext domain with uniform distribution, C is the ciphertext of X, $\mathcal{K}_u$ is the decryption key for user $u \in \mathcal{U}$, and X_u is the decrypted message for user u.

4. CHAMELEON ENCRYPTION SCHEMES UNDER KNOWN-PLAINTEXT ATTACK

We first describe some constructions of Chameleon encryption [4, 6, 2], and then show that these constructions are vulnerable under known-plaintext attack.

4.1 Existing Constructions

4.1.1 Chameleon Encryption.

Anderson *et al.* [4] proposed the first Chameleon encryption scheme. The scheme has a parameter L, which represents the size of lookup table (LUT).

1. Encryption key is a $L \times 1$ table $\mathbf{E}$, and each table entry $\mathbf{E}[i]$, $0 \le i \le L-1$, stores a 64-bit bit string.
2. Decryption key for user u is a table $\mathbf{D}_u$ of the same size, and for $0 \le i \le L-1$, $\mathbf{D}_u[i] = \mathbf{E}[i] \oplus w_{u,i}$, where $w_{u,i}$ is a 64-bit personalized watermark with very small hamming weight and $\oplus$ is the XOR operator. Watermark $(w_{u,1}, w_{u,2}, \ldots, w_{u,L})$ identifies user u.
3. Encryption of $(x_0, x_1, \ldots, x_{m-1})$, where $x_i \in \{0,1\}^{64}$, $0 \le i \le m-1$:
   (a) Choose a random seed s, and from s generate a pseudorandom sequence $t_1, t_2, \ldots, t_{4m}$, where $t_j \in [0, L-1], 1 \le j \le 4m$.
   (b) For each $0 \le i \le m-1$, $c_i \leftarrow x_i \oplus \mathbf{E}[t_{4i}] \oplus \mathbf{E}[t_{4i-1}] \oplus \mathbf{E}[t_{4i-2}] \oplus \mathbf{E}[t_{4i-3}]$.
   (c) Output $(s, c_0, c_1, \ldots, c_{m-1})$.
4. Decryption of $(s, c_0, c_1, \ldots, c_{m-1})$ using decryption key $\mathbf{D}_u$ by user u
   (a) From s generate a pseudorandom sequence $t_1, \ldots, t_{4m}$, where $t_j \in [0, L-1], 1 \le j \le 4m$.
   (b) For each $0 \le i \le m-1$,

$$\begin{aligned}\widetilde{x}_i &\leftarrow c_i \oplus \mathbf{D}_u[t_{4i}] \oplus \mathbf{D}_u[t_{4i-1}] \oplus \mathbf{D}_u[t_{4i-2}] \oplus \mathbf{D}_u[t_{4i-3}] \\ &= x_i \oplus (\mathbf{E}[t_{4i}] \oplus \mathbf{D}_u[t_{4i}]) \oplus (\mathbf{E}[t_{4i-1}] \oplus \mathbf{D}_u[t_{4i-1}]) \oplus \\ &\quad (\mathbf{E}[t_{4i-2}] \oplus \mathbf{D}_u[t_{4i-2}]) \oplus (\mathbf{E}[t_{4i-3}] \oplus \mathbf{D}_u[t_{4i-3}]) \\ &= x_i \oplus (w_{u,4i} \oplus w_{u,4i-1} \oplus w_{u,4i-2} \oplus w_{u,4i-3}) \,.\end{aligned}$$

   (c) Output $(\widetilde{x}_0, \widetilde{x}_1, \ldots \widetilde{x}_{m-1})$.

4.1.2 Chameleon encryption with algebraic addition and large S.

Celik *et al.* [6] gave a variant version of Chameleon encryption, which is similar to the first Chameleon encryption scheme, except that it uses usual algebraic addition and subtraction to replace XOR, and real numbers to replace bit strings. Their scheme has two system parameters, L and S, where L is the size of LUT table and S is the number of LUT entries selected to encrypt/decrypt a single character. The scheme can be described as follows.

1. Encryption key is a $L \times 1$ table $\mathbf{E}$, and each table entry $\mathbf{E}[i]$ is independently and identically sampled from a fixed probability distribution (e.g. Gaussian distribution).
2. Decryption key for user u is a table $\mathbf{D}_u$ of the same size, and for $0 \le i \le L-1$,

$$\mathbf{D}_u[i] = -\mathbf{E}[i] + \mathbf{W}_u[i],$$

   where $\mathbf{W}_u$ is a $L \times 1$ personalized watermark LUT for user u and the entries of $\mathbf{W}_u$ are independently and identically sampled from some fixed distribution (e.g. Gaussian distribution).
3. Encryption of $(x_0, x_1, x_2, \ldots, x_{m-1})$, where $x_i \in \mathbb{R}$, $0 \le i \le m-1$:
   (a) Choose a random seed s, and from s generate a pseudorandom sequence $\{t_{i,j} \in [0, L-1] : 0 \le i \le m-1, 0 \le j \le S-1\}$.
   (b) For each $0 \le i \le m-1$,

$$c_i \leftarrow x_i + \sum_{j=0}^{S-1} \mathbf{E}[t_{i,j}].$$

4. Decryption of $(s, c_0, c_1, \ldots, c_{m-1})$ using decryption key $\mathbf{D}_u$ by user u

[2]The value of threshold τ is determined by applications.

(a) From s generate a pseudorandom sequence $\{t_{i,j} \in [0, L-1] : 0 \le i \le m-1, 0 \le j \le S-1\}$.

(b) For each $0 \le i \le m-1$,

$$\widetilde{x}_i = c_i + \sum_{j=0}^{S-1} \mathbf{D}_u[t_{i,j}] = x_i + \sum_{j=0}^{S-1} \mathbf{W}_u[t_{i,j}].$$

(c) Output $(\widetilde{x}_0, \widetilde{x}_1, \ldots \widetilde{x}_{m-1})$.

Celik *et al.* [6] remarked that larger S complicates cryptanalysis under known-plaintext attack.

4.1.3 Fingercasting.

Adelsbach *et al.* [2] gave a variant of Chameleon encryption with proof of security. Their scheme replaces XOR operation with modular addition and subtraction with a large prime modulus.

4.2 Vulnerabilities in Existing Constructions

All of the above three constructions are not KPA-secure: given several pairs of plaintext-ciphertext, a malicious user who has a decryption key, can find the long term encryption key or a good approximation of it. Section 4.2.1 studies the case that the adversary has full knowledge of the plaintexts corresponding to some given ciphertexts, and Section 4.2.2 studies the case that the adversary only has partial information of the plaintexts corresponding to some given ciphertexts.

4.2.1 KPA with Full Knowledge of Plaintext.

As mentioned by [6], the LUT based encryption can be represented in matrix form

$$\vec{c} = \vec{x} + \mathcal{T}\mathbf{E},$$

where $\vec{x}$ is a vector of plaintext, $\vec{c}$ is a vector of ciphertext, and $\mathcal{T}$ is a $m \times L$ matrix determined by the seed s, such that[3] $\mathcal{T}_{i,j} = 1$ if $j \in \{t_{i,\iota} : t_{i,\iota}$ is generated from seed $s\}$; otherwise $\mathcal{T}_{i,j} = 0$. Under known-plaintext attack, adversary knows $\vec{c}$, $\vec{x}$, and s (s implies $\mathcal{T}$). To find the master encryption table $\mathbf{E}$, the adversary just solves the linear system of equations $\mathcal{T}\mathbf{E} = \vec{c} - \vec{x}$ with $\mathbf{E}$ as unknowns.

XOR operation can be viewed as addition in $GF(2)$. So the above attack can also be applied to Anderson *et al.* [4].

Solving a large linear equation system.

Linear system over real numbers can be solved efficiently using many methods, including Gaussian elimination and iterative method (e.g. Gauss-Seidel method and the successive over-relaxation method). To solve a linear system with L equations and L unknowns, the complexity of Gaussian elimination is $O(L^3)$, and the successive over-relaxation method is in $O(tL^2)$ where t denotes the number of iterations.

Available tools for solving linear system include commercial softwares like MATLAB [1, 10], and free open source softwares like LAPACK [3], ScaLAPACK and PLAPACK. Particularly, ScaLAPACK and PLAPACK are extensions of LAPACK for distributed-memory systems and are suitable for grid computing.

Note that the encryption/decryption complexity is linear in S. So to achieve fast decryption, it is desireable to have a small S. For small S, the matrix $\mathcal{T}$ is sparse: Every row has exactly S non-zero entries, which are located at column $t_{i,j}$'s, where $1 \le j \le S$ and $t_{i,j}$'s are generated from seed s. For such matrix, the successive over-relaxation method is even more efficient: $O(tLS)$. Furthermore, if the adversary knows a decryption key $\mathbf{D}_u$, which is a good approximation of the unknowns $\mathbf{E}$, then the adversary has a good initial guess for the iterative method. Consequently, the number t of iterations is likely to be very small.

[3]Here, for simplicity, we assume that $t_{i,\iota}$'s generated from the seed s are all distinct.

4.2.2 KPA with Partial Knowledge of Plaintext.

Let the plaintext vector $\vec{x}$, ciphertext vector $\vec{c}$, matrix $\mathcal{T}$, parameter S, encryption key $\mathbf{E}$ and decryption key $\mathbf{D}_u$ of user u be as described in the above subsections, and let us denote with $\vec{x}_u$ the data obtained by decrypting $\vec{c}$ with key $\mathbf{D}_u$. We use the L^2-Norm as the distance function, and denote it with Dist: for two vectors $\vec{x}$ and $\vec{y}$ of the same dimension n, $\mathsf{Dist}(\vec{x}, \vec{y}) \overset{def}{=} \sqrt{\sum_{i=1}^{n} (\vec{x}[i] - \vec{y}[i])^2}$.

We now consider an attacker, who has a decryption key $\mathbf{D}_u$ and an approximation $\vec{y}$ of plaintext $\vec{x}$. The decryption table $\mathbf{D}_u$ is generated by independently adding a Gaussian noise with distribution $\mathcal{N}(0, \sigma_1)$ to each entry of encryption table $\mathbf{E}$ as described in [6]. As a result, the watermark, i.e. each component of vector $\vec{x}_u - \vec{x}$, follows a Gaussian distribution $\mathcal{N}(0, \sigma_1 S)$. We assume that each component of vector $\vec{y} - \vec{x}$ follows a Gaussian distribution $\mathcal{N}(0, \sigma_2)$ for some $\sigma_2 < \sigma_1 S$, to reflect the notion that the KPA-adversary has additional knowledge on $\vec{x}$ beyond $\vec{x}_u$. The goal of the attacker is to find a "better" decryption table $\widetilde{\mathbf{D}}$, such that $\mathsf{Dist}(\widetilde{\mathbf{D}}, \mathbf{E}) < \mathsf{Dist}(\mathbf{D}_u, \mathbf{E})$.

The attacker can find such $\widetilde{\mathbf{D}}$ by solving the (over-determined) linear equation system

$$\vec{y} + \mathcal{T}\widetilde{\mathbf{D}} \approx \vec{c},$$

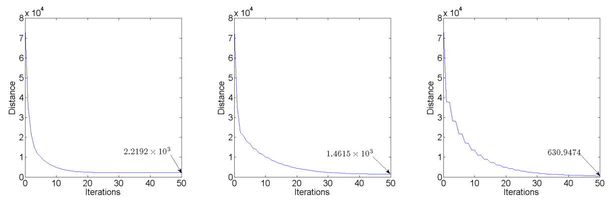
in a least square manner, that is, to find a vector $\widetilde{\mathbf{D}}$ such that the Euclidean length of the vector $(\vec{y} + \mathcal{T}\widetilde{\mathbf{D}} - \vec{c})$ is minimized. This could be solved using least square fitting [12]. Since the attacker already has a decryption table $\mathbf{D}_u$ that is close to the encryption table $\mathbf{E}$, he can use iterative method, for example, the LSQR method[16, 5] and speed up the computation with $\mathbf{D}_u$. When S is small, the matrix $\mathcal{T}$ is sparse, and thus the complexity in solving the linear system can be further reduced [9].

To illustrate the efficiency and effectiveness of the attack, we consider the following setting:

1. The encryption table $\mathbf{E}$ has $2^{16} = 65536$ entries of integers in $\mathbb{Z}_{65536}$.
2. The noise in decryption key $\mathbf{D}_u$ follows distribution $\mathcal{N}(0, \sigma_1)$ with $\sigma_1 = 300$.
3. The noise in vector $\vec{y}$ follows distribution $\mathcal{N}(0, \sigma_2)$ with $\sigma_2 = 30$.
4. The number of equations that the attacker obtained, i.e. $|\vec{y}|$, is $2^{18} = 262144$.
5. We simulate for three different values of S: $S = 4$ (Figure 1(a)), $S = 10$ (Figure 1(b)) and $S = 10$ (Figure 1(c)).

We run the iterative method using MATLAB 7.8.0 in a Windows XP SP3 system on a Dell desktop PC with Intel Core 2 Duo 2.33GHz CPU with 4GB of memory. The simulation utilizes only one CPU core and takes less than 2 hours to finish 50 iterations. Figure 1 shows the distance

$\mathsf{Dist}(\widetilde{\mathbf{D}}, \mathbf{E})$ for different iterations and S, where iteration 0 corresponds to the origin decryption key $\mathbf{D}_u$. From Figure 1, we can see that within 50 iterations, the adversary can obtain a good approximation $\widetilde{\mathbf{D}}$ of the master encryption key $\mathbf{E}$, such that $\mathsf{Dist}(\widetilde{\mathbf{D}}, \mathbf{E}) \le 0.05 \cdot \mathsf{Dist}(\mathbf{D}_u, \mathbf{E})$.



(a) $S = 4$ (b) $S = 10$ (c) $S = 80$

Figure 1: The distance $\mathsf{Dist}(\widetilde{\mathbf{D}}, \mathbf{E})$ of obtained decryption key $\widetilde{\mathbf{D}}$ from the encryption key $\mathbf{E}$ for 50 iterations for different values of S. Note: (1) The y-axis coordinate corresponding to Iteration 0 indicates $\mathsf{Dist}(\mathbf{D}_u, \mathbf{E})$. (2) The dimensions of vectors $\widetilde{\mathbf{D}}, \mathbf{E}$ and $\mathbf{D}_u$ are all $2^{16} = 65536$.

5. A NEW CHAMELEON ENCRYPTION SCHEME BASED ON MAJORITY VOTE

In this section, we first give an initial construction of our scheme based on majority voting. Then we augment the initial construction with ECC. We defer the analysis of the scheme and the discussion of choice of parameters to Section 6.

5.1 Initial Scheme

Our Chameleon encryption scheme is a one-time pad cipher combined with a special PRNG, called MUTABLE-PRNG.

5.1.1 Mutable Pseudo Random Number Generator.

Let $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^{poly(\kappa)}$ be a PRNG , and $\widetilde{\mathsf{G}}(s, i)$ be the i-th bit of the output of $\mathsf{G}(s)$. We construct a PRNG F based on G as follows: Let $s_i \in \{0,1\}^\kappa, 1 \le i \le n$, $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ be a (multi)set, and $\|$ denote the string concatenation.

$$\widetilde{\mathsf{F}}(\mathcal{S}, i) \overset{def}{=} \texttt{The majority bit among } \widetilde{\mathsf{G}}(s_1, i), \dots, \widetilde{\mathsf{G}}(s_n, i);$$

$$\mathsf{F}(\mathcal{S}) \overset{def}{=} \widetilde{\mathsf{F}}(\mathcal{S}, 1) \| \widetilde{\mathsf{F}}(\mathcal{S}, 2) \| \widetilde{\mathsf{F}}(\mathcal{S}, 3) \| \dots \| \widetilde{\mathsf{F}}(\mathcal{S}, poly(\kappa)). \quad (1)$$

We call F a MUTABLE-PRNG, since F is a PRNG and satisfies a special property:

- There exists an efficient algorithm, which takes as input $\mathcal{S}$ and $\mathcal{U}$ and outputs $\{\mathcal{S}_u : u \in \mathcal{U}\}$, such that for any u, (1) $\mathcal{S}_u$'s are distinct and (2) $|\mathsf{F}(\mathcal{S}_u)| = |\mathsf{F}(\mathcal{S})|$ and the Hamming distance between $\mathsf{F}(\mathcal{S})$ and $\mathsf{F}(\mathcal{S}_u)$ is small (say, less than 10% of the length of $\mathsf{F}(\mathcal{S})$) with high probability;
- $\mathcal{S}_u$ is short (e.g. $|\mathcal{S}_u| = O(|\mathcal{S}|)$) and the expansion factor of F is large (e.g. $poly(\kappa) = \Omega(\kappa^2)$).

From the security point of view, we require that: Given $\mathcal{S}_u$ and without $\mathcal{S}$, it is hard to find $\mathsf{F}(\mathcal{S}) \oplus \mathsf{F}(\mathcal{S}_u)$ (We defer the security analysis to Section 6.2).

The above property makes F very suitable for watermarking scheme: The content distributor can use seed $\mathcal{S}$ to generate an *encryption key stream*, which will be XORed with the plaintext to generate a ciphertext, and a user u may use seed $\mathcal{S}_u$ to generate a *decryption key stream*, which will be XORed with the ciphertext to generate a watermarked plaintext.

In practice, one may run a secure block cipher (e.g. AES [7]) or hash function (SHA1 [8]) in counter mode to simulate the PRNG G [22] .

5.1.2 The Construction.

1. (Content Distributor) $\mathsf{KGen}(1^\kappa)$: Key setup. Choose n seeds $s_1, s_2, \dots, s_n$ at random from $\{0,1\}^\kappa$. Keep $\mathcal{S} = \{s_1, s_2, \dots, s_n\}$ private as the encryption key. For $u \in \mathcal{U}$, choose at random a subset $\mathcal{S}_u$ of size k from $\mathcal{S}$, and distribute $\mathcal{S}_u$ to user u as the decryption key.
2. (Content Distributor) $\mathsf{Enc}(x; \mathcal{S})$: Encryption of $x = x_1 x_2 x_3 \dots x_\ell$, where $x_i \in \{0,1\}, 1 \le i \le \ell$. The content distributor maintains an integer state variable λ. Initially, $\lambda = 0$. For $1 \le i \le \ell$,
$$c_i = x_i \oplus \widetilde{\mathsf{F}}(\mathcal{S}, \lambda + i).$$
Let $c = c_1 c_2 \dots c_\ell$. Output (c, λ). Update the state variable λ: $\lambda \leftarrow \lambda + \ell$.
3. (User) $\mathsf{Dec}(c, \lambda; \mathcal{S}_u)$: Decryption of (c, λ) using decryption key $\mathcal{S}_u$. For $1 \le i \le \ell$,
$$x_{u,i} = c_i \oplus \widetilde{\mathsf{F}}(\mathcal{S}_u, \lambda + i).$$
Output $x_u = x_{u,1} x_{u,2} x_{u,3} \dots x_{u,\ell}$.

5.2 Extended Scheme with Error Correcting Code

To achieve good quality of the multimedia service from the watermarked data, we have to be able to control the Hamming distance between the watermarked data and the original, or equivalently the Hamming distance between the encryption key stream generated by $\mathsf{F}(\mathcal{S})$ and decryption key stream generated by $\mathsf{F}(\mathcal{S}_u)$. We can adjust this distance by changing the values of k and n. Furthermore, we are able to decrease this distance using ECC, without changing k and n.

The main idea is to divide the data into blocks and apply ECC encoding on them, and then take the resulting codewords as the input of Enc. Formally, let $\mathcal{E}_{\mathsf{ECC}}$ and $\mathcal{D}_{\mathsf{ECC}}$ be the (ℓ, t, d)-error correcting encoding and decoding algorithm based on Hamming distance (e.g. Reed-Solomon Code [18]), where t is the length of the origin code, ℓ is the length of the resulting code, and d is the number of errors that can be corrected. The new encryption and decryption functions are as follows:

$$\mathsf{EncECC}(x, \mathcal{S}) = \mathsf{Enc}(\mathcal{E}_{\mathsf{ECC}}(x), \mathcal{S}); \quad (2)$$

$$\mathsf{DecECC}(c, \mathcal{S}_u) = \mathcal{D}_{\mathsf{ECC}}(\mathsf{Dec}(c, \mathcal{S}_u)). \quad (3)$$

Thus, assuming the bit error rate in the original watermarking scheme $(\mathsf{Enc}, \mathsf{Dec})$ is γ, the block error rate (i.e. the probability that the number of bit errors in a block is more than threshold) of the extended scheme $(\mathsf{EncECC}, \mathsf{DecECC})$ is $1 - \Phi(d; \ell, \gamma)$, where Φ is the Cumulative Distribution Function (CDF) of binomial distribution. Note that when a block is decrypted wrongly, it is likely that some bits are still correct. Thus, the bit error rate of the decrypted message is less than the block error rate.

6. ANALYSIS OF THE NEW CHAMELEON SCHEME

We analyze the Hamming distance between watermarked data and original data in Section 6.1, and evaluate the security in Section 6.2.

6.1 Hamming Distance between the watermarked data and the original

We study the statistical property of the Hamming distance between the decrypted data and the original. For simplicity, we assume that G used in our scheme is an oracle which outputs true random numbers. We first analyze the probability that 1 bit message can be recovered after decryption. Then we suggest typical values for the system parameters in our scheme. After that, we quantify the probability that a data block can be recovered under the typical parameter setting, taking ECC into account. The expected hamming distance between the decrypted data and the original can be derived directly from such probabilities and the length of data.

6.1.1 *Probability to recover 1 bit.*

Let $X_1, X_2, \ldots, X_n$ be n independent random binary variables following Bernoulli distribution with success probability $p = 0.5$. Let $Y_k = \sum_{i=1}^{k} X_i$. Y_k follows Binomial distribution $\mathbf{B}(k,p)$. Let $Z_k = \chi_k(Y_k)$, where $\chi_k(\cdot)$ is defined in Eq (4):

$$\chi_k(x) = \begin{cases} 1 & (\text{If } x \geq \frac{k}{2}); \\ 0 & (\text{otherwise}). \end{cases} \tag{4}$$

Assume the plaintext is just one bit. Z_n represents encryption key stream of one-time pad cipher and Z_k represents the decryption key stream. We intend to quantify the probability $p_{n,k} \overset{def}{=} \mathsf{Pr}[Z_k = Z_n] = \mathsf{Pr}[\widetilde{\mathsf{F}}(\mathcal{S}, i) = \widetilde{\mathsf{F}}(\mathcal{S}_u, i)]$ that the decrypted message matches the plaintext, where $u \in \mathcal{U}$ and $1 \leq i \leq poly(\kappa)$. Assume n and k are odd numbers.

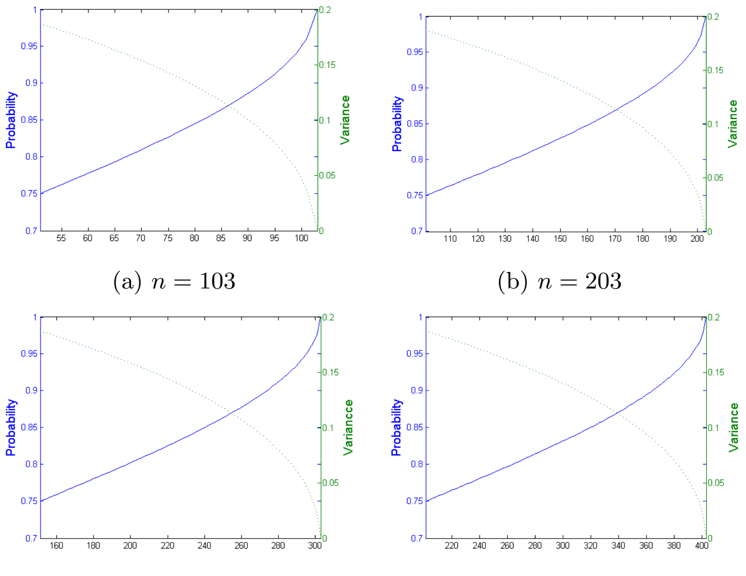
$$\begin{aligned} & \mathsf{Pr}[Z_k = 1, Z_n = 1] \\ = & \sum_{b=(n+1)/2}^{n} \left(\sum_{a=(k+1)/2}^{b} \mathsf{Pr}[Y_k = a,\ Y_n = b] \right) \\ = & \sum_{b=(n+1)/2}^{n} \left(\sum_{a=(k+1)/2}^{b} \mathsf{Pr}[Y_k = a]\mathsf{Pr}[Y_{n-k} = b - a] \right) \\ = & \sum_{b=(n+1)/2}^{n} \left(p^b (1-p)^{n-b} \sum_{a=(k+1)/2}^{b} \binom{k}{a} \binom{n-k}{b-a} \right). \end{aligned}$$

Let $f(n,k,p) \overset{def}{=} \mathsf{Pr}[Z_k = 1, Z_n = 1]$. Then $\mathsf{Pr}[Z_k = 0, Z_n = 0] = f(n,k,1-p)$. Thus, we can express $p_{n,k}$ as a function of n, k and p:

$$\begin{aligned} p_{n,k} &= \mathsf{Pr}[Z_k = 1, Z_n = 1] + \mathsf{Pr}[Z_k = 0, Z_n = 0] \\ &= f(n,k,p) + f(n,k,1-p). \end{aligned}$$

Let us fix the values of n and p, and view $p_{n,k}$ as a function of k. Figure 2 plots the values of $p_{n,k}$ and $p_{n,k}(1-p_{n,k})$ against k, for different values of n. Note that $p_{n,k}$ is approximately linear in $\frac{k}{n}$.

6.1.2 *Effects of parameters n and k.*



(a) $n = 103$ (b) $n = 203$

(c) $n = 303$ (d) $n = 403$

Figure 2: Plot of expectation $p_{n,k}$ (in solid line) and variance $p_{n,k}(1-p_{n,k})$ (in dashed line) against k, where $p = 0.5$ and $n = 103, 203, 303$ or 403, respectively. For each value of n, the domain of k is all odd numbers within the interval $[\frac{n-1}{2}, n]$.

The parameters n and k can impact the initial construction of scheme in Section 5, in various ways:

1. If n (or k) is even, it is hard to find a proper definition of "majority", such that the majority bit is always unambiguous and unbiased (assuming G outputs true random numbers). Therefore, we recommend that both n and k are odd numbers.
2. Smaller n will result in faster encryption and smaller k will result in faster decryption.
3. The key space size is $\binom{n}{k}$. For a fixed odd number n, $\binom{n}{k}$ achieves maximum when $k = (n-1)/2$. Note that $\binom{2k+1}{k} > 2^k$.
4. The values of n and k determine $p_{n,k}$ which indicates the probability that the decrypted 1-bit message matches the 1-bit plaintext. Figure 2 plots how $p_{n,k}$ varies with value of k for some fixed n.

If we choose $n = 203$ and $k = 101$, then the key space is larger than 2^{101} and the probability $p_{n,k} \approx 0.75$.

6.1.3 *Effect of ECC and Probability to recover a data block.*

We employ ECC to improve similarity between the decrypted message and the plaintext without changing n and k, and present the extended scheme in Section 5.2.

Let us consider the setting $n = 2k+1$, $n = 203$ and $k = 101$. Then the probability $p_{n,k} \approx 0.75$. We choose ECC code which generate a ℓ bits codeword from t bits plaintext, and can correct up to $(1-\alpha)\ell$ error bits in an ℓ bits codeword. The probability that a block of t bits plaintext could be recovered with probability $1 - \Phi(\alpha\ell; \ell, 0.75)$, where Φ is the Cumulative Distribution Function (CDF) of binomial distribution $\mathbf{B}(\ell, 0.75)$. We calculate this probability with different values of α and ℓ and show the result in Table 1.

Table 1: The probability that t bits plaintext can be recovered using ECC. In the extended scheme given in Section 5.2, this probability indicates the similarity between the watermarked decrypted message and the original message.

α	0.7	0.7	0.7	0.65	0.65
ℓ	64	128	256	64	128
$1-\Phi(\alpha\ell;\ell,0.75)$	0.8439	0.9058	0.9625	0.9662	0.9933

6.2 Security of the New Chameleon Encryption

6.2.1 Known-Plaintext Attack.

From a plaintext and its ciphertext, one can obtain the key stream used to generate this ciphertext by XORing the plaintext and ciphertext. However, this key stream will not be used any more according to our design. The adversary cannot predict the subsequent bits or guess previous bits from the obtained bit string. That means the obtained key stream cannot help the adversary to decrypt other ciphertexts or to obtain the encryption key. As a result, our scheme is resistant to known-plaintext attack as long as the pseudo random number generator used in our scheme is cryptographically secure. The following Theorem 1 states that the output of $\mathsf{F}(\mathcal{S})$ is unpredictable against adversaries with knowledge of $\mathcal{S}_u \subset \mathcal{S}$. From Theorem 1, the security of our proposed Chameleon encryption scheme against KPA is implied directly. See Corollary 1.

THEOREM 1. *Let F be as constructed in Section 5.1 based on PRNG $\mathsf{G} : \{0,1\}^\kappa \to \{0,1\}^{poly(\kappa)}$, where κ is the security parameter. Let $k > 0$ be an odd integer. Let $\mathcal{S}$ be a (multi)set of $n = 2k+1$ independent random seeds from $\{0,1\}^\kappa$ and $\mathcal{S}_u$ be an arbitrary subset of $\mathcal{S}$ of size k. Suppose G is cryptographically secure. Then, for any PPT algorithm $\mathcal{A}$, which takes as input security parameter κ, the size n of set $\mathcal{S}$, the size k of the subset $\mathcal{S}_u$, the subset $\mathcal{S}_u$, and an i-bit long prefix, denoted as $\mathsf{Prefix}(X,i)$, of $X \leftarrow \mathsf{F}(\mathcal{S})$, and outputs a guess for the next bit $X[i+1]$, for any $1 \le i < poly(\kappa)$, it holds that*

$$\mathsf{Pr}\left[\begin{array}{c}\mathcal{A}(\kappa,n,k,\mathcal{S}_u,\mathsf{Prefix}(X,i)) = X[i+1] \\ \textit{where } X \leftarrow \mathsf{F}(\mathcal{S})\end{array}\right] \le \frac{1}{2} + negl(\kappa),$$

where $negl(\cdot)$ is some negligible function. The probability is taken over all random coins of $\mathcal{A}$ and the random coins used to sample $\mathcal{S}$ and $\mathcal{S}_u$.

Note that we adopt the "unpredictability" version of definition of pseudorandomness [11], and unlike the standard setting of pseudorandomness, here adversary $\mathcal{A}$ has access to $\mathcal{S}_u$, which contains partial information of $\mathcal{S}$. The proof of Theorem 1 is given in Appendix A.

COROLLARY 1. *If G is a cryptographically secure PRNG, then the scheme described in Section 5.1 is secure against known-plaintext attack.*

6.2.2 Bound of Information Loss.

Recall that our ECC scheme can correct up to $(1-\alpha)\ell$ error bits at the cost of expending every t-bit message block to ℓ bits codeword. Such error correcting information might be exploited to defer more information about the original plaintext X than he can with the watermarked message $\widetilde{X}$. In addition, recall that our decryption reveals the computation steps, rather than only the outcome, of the majority voting. Thus, the adversary may defer additional information by analyzing the ECC expended ciphertext together with his key $\mathcal{S}_u$. In this section, we quantify this information leakage by analyzing the *average conditional entropy* [19] $\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S},i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u,i)\big)$ of the encryption bits, and with that, we bound the remaining entropy $\mathbf{H}\big(X \mid (\mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u)\big)$ from below and bound $\mathbf{H}\big(X \mid \widetilde{X}\big) - \mathbf{H}\big(X \mid \mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u\big)$ from above.

For simplicity, let us assume G generate true independent unbiased random bits sequence. From the construction of the function F (equation (1)), we will have (detailed computation is given in equation (9) in Appendix B) the entropy of $\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S},i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u,i)\big)$ is $-f(n,k,0.5)\log(f(n,k,0.5)) - (1-f(n,k,0.5))\log(1-f(n,k,0.5))$, where $f(n,k,0.5)$ is as define in Section 6.1. For the case $n = 203$ and $k = 101$, we have $f(n,k,0.5) = 0.5p_{n,k} \approx 0.375$ and $\mathbf{H}(\widetilde{\mathsf{F}}(\mathcal{S},i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u,i)) \approx 0.9544$. Thus, $\mathbf{H}(X|\widetilde{X}) \approx 0.9544t$.

Now we give an upper bound for the overall entropy loss $\mathbf{H}(X) - \mathbf{H}(X \mid \mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u)$. Recall the encoding function EncECC in equation (2), we want to quantify the remaining entropy of the original message X given $\mathsf{EncECC}(X,\mathcal{S})$ and $\mathcal{S}_u$. Since X and $\mathcal{S}_u$ are independent, we can view $\widetilde{\mathsf{F}}(\mathcal{S},i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u,i)$ as the randomness involved in $\mathsf{EncECC}(X,\mathcal{S})$. And note that $\widetilde{\mathsf{F}}(\mathcal{S},i)$ can be recovered from X and $\mathsf{EncECC}(X,\mathcal{S})$. Thus, we can have the the bound for the entropy loss due to observing the ECC code and a decryption key: $\mathbf{H}\big(X\big) - \mathbf{H}\big(X|(\mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u)\big)$ is $(1-f(n,k,0.5))\log(1-f(n,k,0.5))$ (detailed computation is given in equation (10) in Appendix B).

For the case $n = 203$ and $k = 101$, $\mathbf{H}(X|\mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u)$ is at least $t - 0.0456\ell$ and $\mathbf{H}(X|\widetilde{X}) - \mathbf{H}(X|\mathsf{EncECC}(X,\mathcal{S}),\mathcal{S}_u)$ is at most $0.0456(\ell - t)$.

7. EXTENSION

Our proposed scheme in Section 5 embeds hamming distance based watermark. We may modify it to support L^2-Norm. The modifications are briefed as follows:

1. Redefine $\widetilde{\mathsf{G}}(s,i)$ as the i-th byte of the output of $\mathsf{G}(s,i)$.
2. Redefine $\widetilde{\mathsf{F}}(\mathcal{S},i)$ as

$$\widetilde{\mathsf{F}}(\mathcal{S},i) \overset{def}{=} \frac{1}{n}\sum_{j=1}^{n}\widetilde{\mathsf{G}}(s_j,i),$$

where $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$. Namely, replace "majority vote" with "average".

Since s_j's are independent and $\widetilde{\mathsf{G}}(s_j,i)$'s independently identically distributed, we have $\mathsf{E}(\widetilde{\mathsf{F}}(\mathcal{S},i)) = \mathsf{E}(\widetilde{\mathsf{G}}(s_1,i))$ and $\mathsf{Var}(\widetilde{\mathsf{F}}(\mathcal{S},i)) = \frac{1}{n}\mathsf{Var}(\widetilde{\mathsf{G}}(s_1,i))$. Hence, for any subset $\mathcal{S}_u \subset \mathcal{S}$ with size $|\mathcal{S}_u| = k$, $\mathsf{E}(\widetilde{\mathsf{F}}(\mathcal{S}_u,i)) = \mathsf{E}(\widetilde{\mathsf{F}}(\mathcal{S},i))$ and $\mathsf{Var}(\widetilde{\mathsf{F}}(\mathcal{S}_u,i)) = \frac{n}{k}\,\mathsf{Var}(\widetilde{\mathsf{F}}(\mathcal{S},i))$.

8. CONCLUSIONS

We observed that most existing LUT-based Chameleon encryption schemes are vulnerable under known-plaintext attack: Adversary who has partial knowledge of the plaintext, could obtain a good approximation of the encryption key. We proposed a new Chameleon encryption scheme based on a special PRNG which we call MUTABLE-PRNG. We analyzed the scheme and showed that it is secure against known-plaintext attack. Our scheme leaks some information about the plaintexts. We quantified the information leakage by providing an upper bound. How to use cryptographic tools (like homomorphic cryptographic primitives) to avoid information leakage and how to relate conditional entropy to the robustness of watermark directly are in future work.

9. ACKNOWLEDGEMENT

This work is supported by Grant R-252-000-413-232/422/592 from Temasek Defence Systems Institute (TDSI).

10. REFERENCES

[1] MATLAB. www.mathworks.com.

[2] A. Adelsbach, U. Huber, and A. Sadeghi. Fingercasting-Joint Fingerprinting and Decryption of Broadcast Messages. *Transactions on data hiding and multimedia security II*, pages 1–34, 2007.

[3] E. Anderson, C. B. Z. Bai, S. Blackford, J. Demmel, J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen.

[4] R. J. Anderson and C. Manifavas. Chameleon - A New Kind of Stream Cipher. In *International Workshop on Fast Software Encryption*, pages 107–113, 1997.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. V. der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* 1994.

[6] M. Celik, A. Lemma, S. Katzenbeisser, and M. van der Veen. Lookup-Table-Based Secure Client-Side Embedding for Spread-Spectrum Watermarks. *IEEE Transactions on Information Forensics and Security*, pages 475–487, 2008.

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

[8] D. Eastlake and P. Jones. US secure hash algorithm 1 (SHA1), 2001.

[9] J. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in MATLAB: Design and implementation. *SIAM J. Matrix Anal. Appl*, 13(1):333–356, 1992.

[10] J. R. Gilbert, C. Moler, and R. Schreiber. Sparse matrices in matlab: design and implementation. *SIAM J. Matrix Anal. Appl.*, 13(1):333–356, 1992.

[11] O. Goldreich. *Foundations of Cryptography: Volume 1.* Cambridge University Press, New York, NY, USA, 2006.

[12] R. Hanson. *Solving least squares problems.* 1995.

[13] D. Kundur and K. Karthik. Video fingerprinting and encryption principles for digital rights management. In *Proceedings of the IEEE*, pages 918–932, 2004.

[14] A. Lemma, S. Katzenbeisser, M. Celik, and M. Van Der Veen. Secure Watermark Embedding through Partial Encryption. In *International Workshop on Digital Watermarking*, pages 433–445, 2006.

[15] W. Luh and D. Kundur. New paradigms for effective multicasting and fingerprinting of entertainment media. *IEEE Communications Magazine* , pages 77–84, 2005.

[16] C. Paige and M. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

[17] R. Parviainen and R. Parnes. Large Scale distributed watermarking of multicast media through encryption. In *International Conference on Communications and Multimedia Security Issues*, page 17, 2001.

[18] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, pages 300–304, 1960.

[19] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, pages 379–423, 1948.

[20] P. Tomsich and S. Katzenbeisser. Copyright Protection Protocols For Multimedia Distribution Based On Trusted Hardware. In *Protocols for Multimedia Systems*, pages 815–819, 2000.

[21] P. Tomsich and S. Katzenbeisser. Towards a Secure and De-centralized Digital Watermarking Infrastructure for the Protection of Intellectual Property. In *EC-Web*, pages 38–47, 2000.

[22] A. Young and M. Yung. *Malicious Cryptography: Exposing Cryptovirology.* 2004.

APPENDIX

A. PROOF OF THEOREM 1

Recall that in Section 5.1, we defined functions $\widetilde{\mathsf{G}}(\cdot,\cdot)$, $\mathsf{F}(\cdot)$, and $\widetilde{\mathsf{F}}(\cdot,\cdot)$, from some PRNG $\mathsf{G}(\cdot)$. For the sake of proof, let us define two functions $\widetilde{\mathcal{F}}$ and $\mathcal{F}$, in a similar way that we defined $\widetilde{\mathsf{F}}$ and F. Let set $\mathcal{S} = \{s_1, \ldots, s_n\}$. If n is even and $\sum_{i=1}^{n} \widetilde{\mathsf{G}}(s_i, j) = n/2$, $\widetilde{\mathcal{F}}(\mathcal{S}, j)$ outputs a random bit, otherwise $\widetilde{\mathcal{F}}(\mathcal{S}, j) = \widetilde{\mathsf{F}}(\mathcal{S}, j)$. $\mathcal{F}(\mathcal{S}) = \widetilde{\mathcal{F}}(\mathcal{S}, 1) \| \widetilde{\mathcal{F}}(\mathcal{S}, 2) \| \ldots$ Note that $\widetilde{\mathcal{F}}(\mathcal{S}, j)$ outputs an unbiased but potentially ambiguous majority bit.

In order to prove Theorem 1, we introduce and prove the following claim.

CLAIM 1. *Let* F *be as constructed in Section 5.1 based on PRNG* $\mathsf{G} : \{0,1\}^{\kappa} \to \{0,1\}^{poly(\kappa)}$*, where* κ *is the security parameter. Let* k *be an odd integer. Let* $\mathcal{T}$ *be a (multi)set of* $(k+1)$ *independent random seeds from* $\{0,1\}^{\kappa}$*. If* G *is cryptographically secure, then for any PPT algorithm* $\mathcal{B}$*, for any* $1 \le i < poly(\kappa)$*, it holds that*

$$\mathsf{Pr}\left[\begin{array}{c} \mathcal{B}(\kappa, k, 1^{|Y|}, y_1, \ldots, y_i) = Y[i+1] \\ \textit{where } Y \leftarrow \mathcal{F}(\mathcal{T}) \textit{ and } \forall 1 \le j \le i, \\ y_j \leftarrow \sum_{s \in \mathcal{T}} \widetilde{\mathsf{G}}(s, j) \end{array}\right] \le \frac{1}{2} + negl(\kappa),$$

where $negl(\cdot)$ *denotes some negligible function and the probability is taken over all random coins of* $\mathcal{B}$ *and the random coins used to sample* $\mathcal{T}$*.*

PROOF SKETCH OF CLAIM 1. Suppose there exists a PPT algorithm $\mathcal{B}$ which can break Claim 1 and guess the next bit correctly with probability $1/2 + \epsilon_1$. Then we can construct a PPT algorithm $\mathcal{C}$ which can break the pseudorandomness of

G, i.e. for some i^*, let U_κ denote a uniform random variable over $\{0,1\}^\kappa$, for non-negligible function $\epsilon_2(\kappa)$,

$$\Pr\left[\begin{array}{c}\mathcal{C}(\kappa, 1^{|X|}, \mathsf{Prefix}(X, i^*)) = X[i^*+1] \\ \text{where } X \leftarrow \mathsf{G}(U_\kappa)\end{array}\right] \geq \frac{1}{2} + \epsilon_2(\kappa). \tag{5}$$

Construction of algorithm $\mathcal{C}$.

1. The input of $\mathcal{C}$ is the security parameter κ, the length of X, and i^*-bit prefix of X.
2. Choose k elements $s_1, \ldots, s_k$ from $\{0,1\}^\kappa$ independently and uniformly randomly. Let (multi)set $\mathcal{T}_1 = \{s_1, \ldots, s_k\}$.
3. For each j, $1 \leq j \leq i^*+1$, compute $u_j \leftarrow \sum_{s \in \mathcal{T}_1} \widetilde{\mathsf{G}}(s, j)$.
4. For each j, $1 \leq j \leq i^*$, let σ_j be the j-th bit of $\mathsf{Prefix}(X, i^*)$ and set $y_j \leftarrow u_j + \sigma_j$.
5. Invoke algorithm $\mathcal{B}$ on input $(\kappa, k, 1^{|X|}, y_1, \ldots, y_{i^*})$ and obtain output $\zeta \in \{0,1\}$.
6. If $(k+1)/2 - 1 \leq u_{i^*+1} \leq (k+1)/2$, output ζ; otherwise output a random bit.

One can verify that the constructed algorithm $\mathcal{C}$ satisfies equation (5) and breaks the pseudorandomness of G with non-negligible advantage. Here we save the details. $\square$

The proof for Theorem 1 is given below.

PROOF OF THEOREM 1. Suppose that there exists a PPT algorithm $\mathcal{A}$ which has probability $\frac{1}{2} + \epsilon$ to guess the next bit correctly. More precisely, for some i^*, $1 \leq i^* < poly(\kappa)$, and some non-negligible function ϵ in κ,

$$\Pr\left[\begin{array}{c}\mathcal{A}(\kappa, n, k, \mathcal{S}_u, \mathsf{Prefix}(X, i^*)) = X[i^*+1] \\ \text{where } X \leftarrow \mathsf{F}(\mathcal{S})\end{array}\right] = \frac{1}{2} + \epsilon.$$

We intend to construct a PPT algorithm $\mathcal{B}$ based on $\mathcal{A}$, such that $\mathcal{B}$ can break Claim 1 with probability about $\frac{1}{2} + \frac{1}{4}\epsilon$: Let $\mathcal{T} = \mathcal{S} \setminus \mathcal{S}_u$,

$$\Pr\left[\begin{array}{c}\mathcal{B}(\kappa, k, 1^{|Y|}, y_1, \ldots, y_{i^*}) = Y[i^*+1] \\ \text{where } Y \leftarrow \mathcal{F}(\mathcal{T}) \text{ and } \forall 1 \leq j \leq i^*, \\ y_j \leftarrow \sum_{s \in \mathcal{T}} \widetilde{\mathsf{G}}(s, j)\end{array}\right] \approx \frac{1}{2} + \frac{1}{4}\epsilon.$$

Construction of Adversary $\mathcal{B}$ against Claim 1.

1. The input of $\mathcal{B}$ is $(\kappa, 1^{|Y|}, y_1, \ldots, y_{i^*})$ where $\forall 1 \leq j \leq i^*, y_j \leftarrow \sum_{s \in \mathcal{T}} \widetilde{\mathsf{G}}(s, j)$, and $\mathcal{T}$ is a set of $(k+1)$ independent random elements from $\{0,1\}^\kappa$.
2. Choose k elements $s_1, \ldots, s_k$ independently and uniformly randomly from $\{0,1\}^\kappa$. Let $\mathcal{T}' = \{s_1, \ldots, s_k\}$.
3. For each $1 \leq j \leq i^*$, compute $z_j \leftarrow \sum_{s \in \mathcal{T}'} \widetilde{\mathsf{G}}(s, j)$.
4. For each $1 \leq j \leq i^*$, if $y_j + z_j \geq k+1$, then set $\nu_j \leftarrow 1$, otherwise set $\nu_j \leftarrow 0$.
5. Invoke $\mathcal{A}$ on input $(\kappa, 2k+1, k, \mathcal{T}', \nu_1\nu_2 \ldots \nu_{i^*})$, and obtain output $b \in \{0,1\}$.
6. Compute $b' \leftarrow \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1) \quad \in \{0,1\}$.
7. If $b \neq b'$, then output b; otherwise output a random bit.

Success Probability of Adversary $\mathcal{B}$ against Claim 1.

Let $\mathbf{E}_\mathcal{A}$ ($\mathbf{E}_\mathcal{B}$, respectively) denote the event that $\mathcal{A}$ ($\mathcal{B}$, respectively) guesses the next bit correctly. The input that $\mathcal{B}$ feeds in $\mathcal{A}$ has identical distribution as the input of $\mathcal{A}$ in Theorem 1. Therefore, according to our hypothesis in the beginning of proof, $\Pr[\mathbf{E}_\mathcal{A}] = \frac{1}{2} + \epsilon$.

Let $\mathcal{W} = \mathcal{T} \cup \mathcal{T}'$, and $\Pr\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] = q$. We have to point out that through our calculation in Section 6.1 (See Figure 2), $q \approx 0.75$, and we will substitute q with 0.75 in the last step of calculation in order to avoid the potential amplification of numerical errors during the calculation.

Our calculation will utilize these facts:

1. $\Pr[b = \widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) \mid \mathbf{E}_\mathcal{A}] = 1$.
2. $\Pr[b = b' \mid \mathbf{E}_\mathcal{A}] = \Pr[b \neq b' \mid \neg\mathbf{E}_\mathcal{A}] =$ $\Pr\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] = q$.
3. $\Pr[\mathbf{E}_\mathcal{B} \mid \mathbf{E}_\mathcal{A}, b \neq b'] = 1$.

Now we calculate the probability $\Pr[\mathbf{E}_\mathcal{B}]$ that $\mathcal{B}$ guesses the next bit correctly.

$$\begin{aligned}
& \Pr[\mathbf{E}_\mathcal{B}] \\
= \; & \Pr[\mathbf{E}_\mathcal{B} \mid b = b'] \Pr[b = b'] + \Pr[\mathbf{E}_\mathcal{B}, b \neq b'] \\
= \; & \frac{1}{2}\Big(\Pr[b = b' \mid \mathbf{E}_\mathcal{A}] \Pr[\mathbf{E}_\mathcal{A}] + \Pr[b = b' \mid \neg\mathbf{E}_\mathcal{A}] \Pr[\neg\mathbf{E}_\mathcal{A}]\Big) \\
& + \Big(\Pr[\mathbf{E}_\mathcal{B} \mid \mathbf{E}_\mathcal{A}, b \neq b'] \Pr[b \neq b' \mid \mathbf{E}_\mathcal{A}] \Pr[\mathbf{E}_\mathcal{A}] + \\
& \Pr[\mathbf{E}_\mathcal{B} \mid \neg\mathbf{E}_\mathcal{A}, b \neq b'] \Pr[b \neq b' \mid \neg\mathbf{E}_\mathcal{A}] \Pr[\neg\mathbf{E}_\mathcal{A}]\Big) \\
= \; & \frac{1}{2}\Big(q \cdot (\frac{1}{2} + \epsilon) + (1-q) \cdot (\frac{1}{2} - \epsilon)\Big) + \\
& \Big(1 \cdot (1-q) \cdot (\frac{1}{2} + \epsilon) + \\
& \Pr[\mathbf{E}_\mathcal{B} \mid \neg\mathbf{E}_\mathcal{A}, b \neq b'] \cdot q \cdot (\frac{1}{2} - \epsilon)\Big)
\end{aligned} \tag{6}$$

We turn to calculate the probability $\Pr[\mathbf{E}_\mathcal{B} \mid \neg\mathbf{E}_\mathcal{A}, b \neq b']$.

$$\begin{aligned}
& \Pr[\mathbf{E}_\mathcal{B} \mid \neg\mathbf{E}_\mathcal{A}, b \neq b'] \\
& = \frac{\Pr[\mathbf{E}_\mathcal{B}, b \neq b' \mid \neg\mathbf{E}_\mathcal{A}]}{\Pr[b \neq b' \mid \neg\mathbf{E}_\mathcal{A}]} \\
& = \frac{\Pr\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) \neq \widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right]}{q} \\
& = \frac{1}{q}\Big(\Pr\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] - \\
& \Pr\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right]\Big) \\
& = 1 - \frac{1}{q}\Pr\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right]
\end{aligned} \tag{7}$$

Now we turn to calculate the probability $\Pr\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right]$. At first, we define k' as

$$k' = \begin{cases} k+1 & \text{(If } k \text{ is odd);} \\ k & \text{(otherwise).} \end{cases}$$

Note that $\mathcal{W} = \mathcal{T} \cup \mathcal{T}'$, $\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1) \Rightarrow \widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathcal{F}}(\mathcal{T}, i^*+1)$.

$$
\begin{aligned}
& \mathsf{Pr}\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] \\
= \; & \mathsf{Pr}\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] \\
= \; & \sum_{j \in \{0,1\}} \mathsf{Pr}\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = j, \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1) = j\right] \\
= \; & \sum_{j \in \{0,1\}} \mathsf{Pr}\left[\widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = j\right] \mathsf{Pr}\left[\widetilde{\mathsf{F}}(\mathcal{T}', i^*+1) = j\right] \\
= \; & \sum_{j \in \{0,1\}} \frac{1}{2} \cdot \frac{1}{2} \\
= \; & \frac{1}{2}. \qquad (8)
\end{aligned}
$$

Substituting $\mathsf{Pr}\left[\widetilde{\mathsf{F}}(\mathcal{W}, i^*+1) = \widetilde{\mathcal{F}}(\mathcal{T}, i^*+1) = \widetilde{\mathsf{F}}(\mathcal{T}', i^*+1)\right] = \frac{1}{2}$ into equation (7), we have $\mathsf{Pr}\left[\mathbf{E}_{\mathcal{B}} \mid \neg\mathbf{E}_{\mathcal{A}}, b \neq b'\right] = 1 - \frac{1}{2q}$. Substituting this result into equation (6), we have

$$\mathsf{Pr}\left[\mathbf{E}_{\mathcal{B}}\right] = \frac{1}{2} + \epsilon(1-q) \approx \frac{1}{2} + \frac{1}{4}\epsilon.$$

Since ϵ is non-negligible in κ, $\mathsf{Pr}\left[\mathbf{E}_{\mathcal{B}}\right] - \frac{1}{2} = \epsilon(1-q) \approx \frac{1}{4}\epsilon$ is non-negligible in κ.

Therefore, $\mathcal{B}$ breaks Claim 1. The contradiction implies that our hypothesis in the beginning of this proof is false and consequently Theorem 1 is correct. $\square$

B. MORE DETAILS ON THE COMPUTATION OF BOUND OF INFORMATION LOSS

We will now give the computation for the entropy of $\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S}, i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u, i)\big)$:

$$
\begin{aligned}
& \mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S}, i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u, i)\big) \\
& = -\sum_{y \in \{0,1\}} \frac{1}{2} \cdot \sum_{x \in \{0,1\}} \Big(\mathsf{Pr}\big(\widetilde{\mathsf{F}}(\mathcal{S}_u, i) = y \mid \widetilde{\mathsf{F}}(\mathcal{S}, i) = x\big) \cdot \\
& \qquad \log \mathsf{Pr}\big(\widetilde{\mathsf{F}}(\mathcal{S}_u, i) = y \mid \widetilde{\mathsf{F}}(\mathcal{S}, i) = x\big)\Big) \\
& = -f(n,k,0.5)\log(f(n,k,0.5)) - (1-f(n,k,0.5))\log(1-f(n,k,0.5)) \qquad (9)
\end{aligned}
$$

From $\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S}, i) \mid \widetilde{\mathsf{F}}(\mathcal{S}_u, i)\big)$, we can have:

$$
\begin{aligned}
& \mathbf{H}\big(x\big) - \mathbf{H}\big(x | (\mathsf{EncECC}(x, \mathcal{S}), \mathcal{S}_u)\big) \\
\leq \; & \mathbf{H}\big(x\big) - \Big(\mathbf{H}\big(x\big) - \log(2^{\ell}) + \ell\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S}, i) | \widetilde{\mathsf{F}}(\mathcal{S}_u, i)\big)\Big) \\
= \; & \ell + \ell \cdot \big(\; f(n,k,0.5)\log(f(n,k,0.5)) + \\
& (1-f(n,k,0.5))\log(1-f(n,k,0.5)) \;\big). \qquad (10)
\end{aligned}
$$

where the inequality holds since the information carried by $\mathsf{EncECC}(x, \mathcal{S})$ is at most ℓ bits, and we can view $\ell\mathbf{H}\big(\widetilde{\mathsf{F}}(\mathcal{S}, i) | \widetilde{\mathsf{F}}(\mathcal{S}_u, i)\big)$ as the randomness that can be recovered by x, $\mathsf{EncECC}(x, \mathcal{S})$ and $\mathcal{S}_u$).