SECURITY IN WATERMARKING AND AUTHENTICATION OF NOISY DATA

LI QIMING

(B.Eng.(Hons), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

I would like to express my greatest gratitude to my Ph.D. supervisor, Dr. Chang Ee-Chien. Without his guidance in the past few years this thesis would not be possible. Dr. Chang has always been working closely with his students. He is always ready to discuss with the students about any difficulty they meet in their research, and he would spend hours of his precious time with his students. This is the part that I appreciated most because most of the inspirations and ideas on our research works came from such discussions, and I learnt a lot about how research should be conducted from these discussions.

I have been very grateful to my dearest wife, Mary, who has been giving me endless love and support, and took good care of me during the time I worked on my research problems. I would not have accomplished so much without her. I also owe gratitude to our lovely daughter, Lucy, whose birth gave us endless joy. I would also thank my parents, who concerned about me and have been supporting me in whatever way they could from the time I entered the National University of Singapore as an undergraduate, although they are thousands of miles away from Singapore.

Finally I would thank many of my friends who cared about me, or discussed research problems with me, during the past few years. I wish they are all doing well in their career and lead wonderful lives.

Contents

Summary

Many types of data, such as images and biometric templates, are noisy in nature in the sense that, small noises can be added to the data without changing their semantics and authenticity. This leads to the research of robust authentication schemes that can tolerate random noises. On the other hand, additional information (*watermarks*) can be purposely embedded into noisy data to carry secret messages. Such digital watermarking techniques have been intensively studied.

In this thesis, we study the security aspect of digital image watermarking and biometric authentication. Many previous works focus on the robustness of the schemes, that is, the noises are assumed to be random. In contrast, we consider "smart" attackers, who know the algorithms that are used in the systems, and insert carefully designed noises that aim to fail the systems. For these attackers, we do not assume in anyway their algorithms, but only specify their capabilities in different scenarios.

We consider three scenarios in digital image watermarking: (1) We consider watermark detection with proxies, where the owner wants to designated the watermark detection routine to third parties, in such a way that the secrecy of the watermark is preserved. We propose a new framework in which the security can be achieved with a group of proxies, as long as the majority of them are honest. (2) We also consider oracle attacks, where the attackers try to remove the watermark in an image by repeatedly probing a publicly available watermark detector with carefully chosen images. We study a class of watermarking schemes and give a lower bound of the effort required by the attackers. We also give an attack algorithm that matches the lower bound. (3) Lastly, we consider inversion attacks, where the attackers try to create ambiguities on the ownership of a work. Adelsbach et al. [2] give the first formal definition of such an attack. We make subtle and important modifications to the definition of a successful attacker, and give a provably secure scheme.

Fault-tolerant authentication can be achieved using *helper information*, which is to be made public and is used to guide the authentication. An important requirement on the helper information is that it should not reveal too much information about the original data (low *entropy loss*). We study the problem of minutiae-based fingerprint authentication using helper information, where the templates can be modeled as point-sets under a combination of different noises. Clancy et al. [14] propose to add random minutiae points into templates to confuse the attackers, but they only considered random attackers. We give several schemes and prove bounds of the entropy loss.

List of Tables

List of Figures

List of Symbols

I	Original image. It is often represented by a vector of real numbers. That is, we write $I = \langle x_1, \cdots, x_n \rangle$.
W	Watermark. This is also often represented by $W = \langle w_1, \cdots, w_n \rangle$.
$\widetilde{I}$	Watermarked image. In the spread-spectrum based watermarking schemes, $\widetilde{I}$ is often computed as $\widetilde{I} = I + kW$ for some factor k.
X	Original biometric data. In this thesis, we assume X is a set of s points, and we write $X = \{x_1, \cdots, x_s\}$.
Y	Biometric data that are corrupted by noise. We write $Y = \{y_1, \cdots, y_s\}$.
P	Helper information.
P_H	Helper information for white noise.
P_S	Helper information for set difference.
$\mathcal{L}$	Entropy loss.
$\mathcal{L}_H$	Entropy loss of the helper information for white noise.
$\mathcal{L}_{SD}$	Entropy loss of the helper information for set difference.

Chapter 1

Introduction

Many types of data are noisy in nature in the sense that, some small noises can be added to the data, either purposely or accidentally, without changing their semantics and authenticity. The possibility of hiding messages in noisy data leads to numerous research papers in digital watermarking, whereas challenges have been posed in designing robust authentication schemes that can tolerate such noises.

In many previous works on digital watermarking and robust authentication, robustness is the focus. In other words, the noises are assumed to follow some random distributions. Although important, these results are not sufficient to show the security of the schemes. In fact, we would never know what an attacker would do in practice, and random attacks are probably the last thing we would expect. In contrast with previous approaches, we focus on "smart" attackers, who know the algorithms and public parameters that we use in the schemes, and carefully insert arbitrary noises into the data so as to fail the systems. In this thesis, we give rigorous treatments for such smart attackers in different scenarios. We do not assume the algorithms the attackers use, and only specify the capabilities of the attackers in those scenarios.

In a digital image watermarking scheme, a piece of information (i.e., a watermark) is embedded into a digital image by slightly modifying the image. Such a watermark is essentially an intentional noise, under which the quality of the image remains the same, for example, as perceived by human eyes. The hidden watermarks can later be detected or extracted from the watermarked image. This technique can be used in many applications. For example, the information about the creator of an image can be embedded into the image before it is published. The presence of the watermark serves as an evidence of the ownership of the image. Note that a robust watermarking scheme may not be *secure* in many circumstances. For example, we should consider *removal attacks*, where the a smart attacker tries to remove the watermark by carefully modifying the image. In that case, the noise added to the image would not necessarily be random.

In this thesis, we consider three scenarios in digital image watermarking: watermark detection with proxies, oracle attacks, and inversion attacks. In these scenarios, the attackers have different capabilities, and the security goals that we want to achieve are different. Our approach differs from many previous works on secure watermarking schemes in that, we do not presume the attack algorithms in any way, but only make assumptions on the capabilities of the attackers. We focus on provable security in all these scenarios, and make use of well accepted techniques in cryptography.

The first scenario we consider is watermark detection with proxies (Section 2.1). Under this scenario, the application requires a publicly available watermark detector, such that any user, including attackers, can send arbitrary images for watermark detection. In this case, it is desirable that the watermark detector reveals as little information about the secret watermark as possible. One approach is to use asymmetric watermarking scheme [27, 24, 25], where different keys are used to embed and to detect the watermarks. However, as shown in [25], known schemes are not satisfactory. Another simple method is to let a Trusted Third Party to perform the watermark detection. Recently, [3, 19] give an interesting method to remove the assumption of the existence of a TTP. Their methods employs zero-knowledge interactive proofs, which can be quite expensive in terms of network communications, and does not allow the owner of the watermark to designate the watermark detection to a third party. We propose an alternative setting that can be considered as a modification to the above approach. Instead of using a TTP or zero-knowledge proofs, we give a secure and efficient scheme that makes use of multiple proxies. Although each individual proxy cannot be trusted, we assume that the group as a whole can be trusted in the sense that the majority of them are always honest.

In the second scenario we consider oracle attacks, or sensitivity attacks (Section 2.2). Similar to the first scenario, there is a publicly available watermark detector acting as a black-box. Unlike the first scenario, we assume that the detector can be trusted, and study how attackers can make use of the very limited information revealed by the detector to remove the embedded watermark with small modification to the image. We assume that the attackers, based on previous queries and the responses from the watermark detector, are able to perform some computations and choose carefully the next (modified) image and send it to the detector for watermark detection. We study a class of watermarking schemes for binary sequences, and relate it to the Twenty Questions Game, which is originally studied by Ulam in 1976 [46]. We give a variant of the game that corresponds to the game between the watermark detector and the attackers. We give a lower bound of the effort required by the attackers, which is measured by the number of queries sent by the attackers. We also give an attack algorithm that matches the lower bound.

The third scenario we consider is inversion attacks, or ambiguity attacks, where the attackers try to create ambiguities on the ownership of a work (Section 2.3). This type of attacks are first considered in [20] for digital images with spread spectrum watermarking, where a remedy is proposed based on a secure hash function. This problem is further studied in [40, 41] for video and audio, and new schemes are proposed. After that, there are a number of works [42, 1, 2] that exploit the weaknesses of known schemes that are claimed to be non-invertible. As mentioned in [1, 2], most proposed scheme either do not come with a satisfactory proof of security, or the proofs are flawed. Due to the difficulty in designing such non-invertible watermarking schemes, [2] proposed a provably secure scheme by making use of a Trusted Third Party. We discuss this type of attacks in detail in Section 2.3, and show that it is possible to construct non-invertible watermarking schemes without a TTP by giving a construction. We make subtle but important changes to the formal definition of a successful attacker given in [2], and prove the security of the scheme using well accepted methods in cryptography.

In a typical biometric authentication system, biometric *templates* are extracted from biometric samples of the users during the registration and stored in a database. These templates are matched with the newly extracted templates at the time of authentication. In such a system, the secrecy of the biometric data stored in the system becomes the most vulnerable link. If the biometric data of a legitimate user is leaked out, it is possible that they will be used by an attacker to impersonate that legitimate user.

In traditional password-based systems, similar problems of lost or revealed passwords can be resolved by applying a secure one-way hash function on the stored passwords, and by regularly changing the passwords. But this is not applicable in biometric authentication systems because biometric data are usually permanently associated with their owner, and it is difficult to design a secure hash function that can tolerate the noises.

Such problems can be solved using *helper information*, which is also referred to as helper data, fuzzy commitment, fuzzy vault, secure sketch, shielding function, and so on [34, 33, 23, 9]. Given some noisy data, some helper information can be computed in such a way that when the data are corrupted by a small noise, the original data can be reconstructed from the corrupted data and the helper information. After that, an almost-uniform cryptographic key can be extracted from the original data and used in traditional cryptographic schemes. An important requirement on the helper information is that it should not reveal too much information about the original data. Dodis et al. [23] give a notion of *entropy loss* that measures the information revealed by the helper information.

Hence, in biometric authentication systems, helper information with low entropy loss can be computed from biometric templates and stored in the database. In this way, even if the database is compromised, not much information about the templates would be revealed.

Not surprisingly, the construction of helper information highly depends on the underlying distance metrics. The constructions of helper information with respect to Hamming distance and set difference are closely related to error-correcting codes, and are being intensively studied. However, these metrics are not sufficient to describe the noises commonly encountered in biometric applications.

Let us take minutiae-based fingerprint verification systems as an example. In such systems, minutiae points are extracted from the fingerprint images as the templates. Each point can be represented by its location in the 2-dimensional space, and probably some other attributes, such as type and orientation. Under noise, each point can be shifted by a small distance, be removed, and new points can be added in. We use *point-set difference* as the distance measure under this combination of noises. We say that two points are close if they are near each other in the 2-dimensional space, and we say that two sets of points are close if the number of pairs of close points between these two sets exceeds a certain threshold.

Clancy et al. [14] proposed a scheme that hides the actually minutiae points by adding random points into the space, and considered random attackers. However, it is very difficult to analyze the entropy loss of the scheme to give a rigorous proof of the security. In Section 3.1, we give several provably secure helper information schemes for point-set difference, which can be extended to higher dimensions, and give upper bounds on the entropy loss.

Chapter 2

Security in Digital Image Watermarking

2.1 Scenario 1: Watermark Detection with Proxies

2.1.1 Scenario Settings

In this scenario (Figure 2.1), the owner of an image I embeds a secret watermark W into I and publishes the watermarked $\widetilde{I}$. The owner wants to designate the task of watermark detection to a group of proxies. There is a verifier, who can ask the proxies to perform watermark detection for any image J. We assume that none of the individuals (which include the owner, the proxies, and the verifier) can be trusted.

We want to design a scheme such that, as long as the majority of the individuals are honest, (1) during the watermark detection the verifier does not obtain any information about W, except whether J is watermarked by W or not, (2) no proxy can learn the secret watermark W, and (3) the validity of the watermark W and the correctness of the watermark detection can be verified.

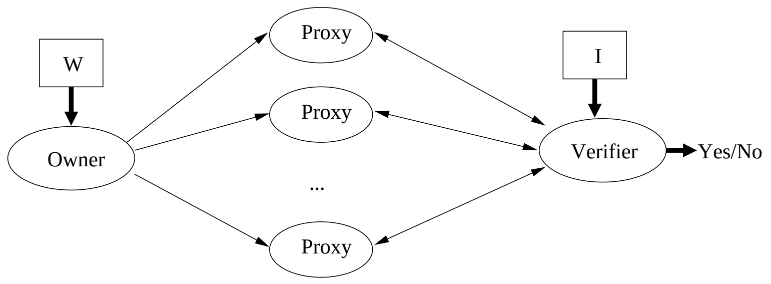

Figure 2.1: Scenario 1: Watermark detection with proxies

2.1.2 Introduction

To achieve the second requirement that the detector should not reveal too much information about the watermark, one possible approach is asymmetric watermarking schemes [27, 24, 25], where the key to embed watermarks is different from the key required during detection. However, the detection keys of known methods do reveal some crucial information, which leads to a number of successful attacks (for e.g., [25] listed a few attacks on specific schemes).

A simple approach achieves secrecy by introducing a trusted third party T. In which case the owner of the watermark gives his watermark W to T, and distributes images. To check if an image is watermarked, a user sends it to T via the Internet, and the result is sent back from T. In this case both the secrecy of W and the interest of the users are protected assuming that T is honest.

Recently, [3, 19] gave interesting methods to remove the assumption of a trusted third party. The methods employ a *prover* (the owner) who proves the existence of the watermark in given images to *verifiers* (the users). The prover is prevented from cheating by the means of commitment schemes, and the secrecy of the watermark is maintained through zero-knowledge interactive proofs. Although these schemes are cryptographically secure, a main drawback is the large number of rounds and bandwidth required in the communications, and they are not easy to implement in practice. Furthermore, if the owner wishes to designate another party to perform the checking and proving, he has to reveal the secret key to this trusted party.

Here, we propose an alternative setting that can be viewed as a modification of the above approach. In this setting, we remove the expensive zero-knowledge interactive proofs without assuming the existence of a trusted third party. Instead, we replace the trusted third party T with a group of *proxies*. Security is maintained if the majority of the proxies are honest.

The individuals in our setting are an owner, a few proxies, and a verifier. At the beginning, the owner generates a secret watermark W and performs a *registration* with the proxies. During registration, the owner distributes some information about the watermark W to the proxies, so that they can carry out watermark detections on their own. After that, the owner embeds the watermark W into his images, which are then released to the public. (The owner could also request the proxies to perform the embedding, without revealing the watermark. We will not describe this operation here.) On the other hand, when a verifier wants to determine if a given image is watermarked by W, he requests the proxies to perform a *detection*. During detection, the verifier sends information about the image to the proxies, who then perform some computations and return the results back to the verifier. Based on these results, the verifier would be able to decide whether the image is watermarked. The registration and detection processes are illustrated in Fig. 2.1.2 below.

Since we do not assume the existence of a trusted third party, no individual can be trusted. Therefore, during registration, the owner does not trust any individual proxy, so he cannot simply send W to each of them. Instead, he needs to distribute the watermark in such a way that it is information-theoretically impossible to compute W even if some of the proxies collude. During detection, the verifier does not trust all proxies because some of them might give wrong results, either intentionally or accidentally. Therefore, there has to be some mechanisms to allow the verifier to detect errors, or even correct them. Furthermore, the verifier does not trust the owner either. A dishonest owner might distribute a watermark that correlates with many images (for instance, in the well-known spread spectrum method [17], a watermark with very high energy would likely give a high correlation value with a randomly chosen image). The dishonest owner might also collude with a few proxies to mislead the verifier.

Note that in this setting, the role of the proxies (as a group) is similar to that of a trusted third party, who will not leak any information of the secret W, and will not cheat the verifier. The main difference is that, in this setting, we only require the integrity of the proxies as a group, which is a much weaker requirement than having a trusted third party. It is also noted that our setting relieves the owner from performing the detections. This is a secondary advantage of using proxies for detection.

The proposed setting naturally suggests the use of secret sharing schemes [44] as a basic building block. A secret sharing scheme breaks a secret z into *shares* and distributes each to a server. No individual server will know the secret unless a number of dishonest servers collude. There are many secret sharing schemes, satisfying various useful properties. For example, Shamir's scheme is also a threshold scheme, and with further modifications it can be verifiable [12, 26] and proactive [30]. An important property required in our setting is that, both multiplications and additions can be supported on the shares. That is, if secrets z_1, z_2 and z_3 are integers and are shared among n severs, the shares of $z_1z_2 + z_3$ can be generated without revealing the values of z_1, z_2, z_3, and $z_1z_2 + z_3$. We give a scheme based on secret sharing. This scheme achieves public watermarking as long as not too many individuals collude. This scheme is arguably easy to implement and is efficient in terms of computation and communication cost.

In Section 2.1.3, we describe the basic watermarking method (spread-spectrum method) used for discussion. Section 2.1.3 gives our proposed multiple proxies setting, and the security requirements. Section 2.1.4 gives a brief description of secret sharing schemes. Our scheme is described in Section 2.1.5, followed by the security analysis in Section 2.1.6. Some discussions on the error-correcting capability of the scheme will be given in Section 2.1.7.

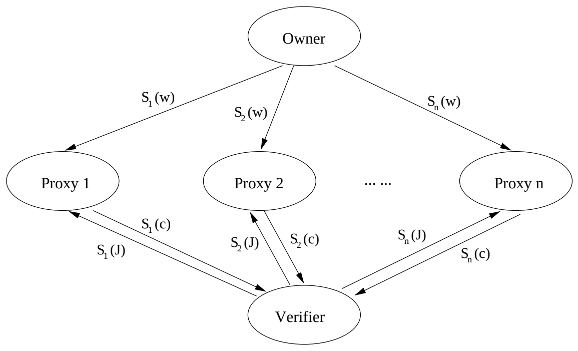

Figure 2.2: The proposed setting in scenario 1.

2.1.3 Notations and Models

Watermarking Model

We employ a variant of the well-known spread spectrum method [17] to embed and detect watermarks. Other watermarking schemes can also be employed as long as the detection involves only multiplication and addition.

Our images and watermarks are “discretized”. An image I is a vector $I = \langle x_1, x_2, \ldots, x_m \rangle$ where each $x_i \in \{0, 1, 2, \ldots, d-1\}$ and d is some integer determined by the media/image representation. For example, d could be 256 if each x_i represents a pixel. The watermark W is also a vector $W = \langle w_1, w_2, \ldots, w_m \rangle$ where each $w_i \in \mathbf{Z}$ is an integer. In addition, the energy of the watermark W is fixed, that is $W \cdot W = E$ where E is some predefined threshold, and $\cdot$ is the vector inner product. The constant E is made known to the public.

During embedding, given an image I and the watermark W, the watermarked image $\widetilde{I}$ is computed as $\widetilde{I} = \text{trunc}(I + W)$, where the function trunc() truncates/rounds the coefficients where values are not in the range $\{0, 1, 2, \ldots, d-1\}$. During detection, given an image J, the correlation value $(J \cdot W)$ is computed. If the correlation value exceeds certain threshold, then J is declared to be watermarked. [1]

Owner, Proxies & Verifier

The individuals in the proposed setting are the owner, n proxies, namely $T_1, T_2, \ldots, T_n$, and the verifiers. The number of verifiers is not important, so we assume that there is only one verifier.

During *registration*, the owner generates a secret watermark W which

[1] We have omitted the normalization of images in the embedding and detection for simplicity. Normalization is not required in our discussion, but still can be incorporated if required.

satisfies $(W \cdot W) = E$, where E is a constant that every individual knows. The owner sends information of W to the n proxies. Let $S_i^{(0)}(W)$ be the data the owner sends to the proxy T_i. Let $S_i(W)$ be the data that T_i keeps and uses in subsequent computations. Note that it is not necessary that $S_i^{(0)}(W) = S_i(W)$. To guard against dishonest owner, the proxies might wish to transform the original $S_i^{(0)}(W)$'s distributed by the owner.

During *detection*, a verifier wishes to know the correlation value $c = J \cdot W$ of a given image J with the secret watermark W. Firstly, the verifier splits J into n pieces, $S_i(J)$, $1 \leq i \leq n$, such that each of them contains partial information about J. Then it sends $S_i(J)$ to proxy T_i respectively. Next, each proxy T_i computes and sends the verifier partial information $S_i(c)$ of the correlation value c. After receiving data from all proxies, the verifier reconstructs the correlation value $(J \cdot W)$ (Fig. 2.1.2).

In our proposed scheme, the partial information communicated through the network, namely $S_i^{(0)}(W)$, $S_i(W)$, $S_i(J)$ and $S_i(c)$, corresponds to the "shares" in secret sharing schemes (see Section 2.1.4). Therefore, we will refer to those pieces of information as shares in the rest of this section, even if some of them may be fraud sent by dishonest individuals.

Security Requirements

A parameter for the security requirement is the *security threshold* t where $t \leq n$. Vaguely, the scheme should tolerate at most $(t-1)$ dishonest individuals (including the proxies, the verifier and the owner). The security requirements can be roughly classified into (a) maintaining the secrecy of W, and (b) protecting the interest of the the verifier. In the following, the first requirement $S1$ belongs to the first class and the remaining belong to the second class.

***S1. Secrecy of* W.** The owner generates and keeps the watermark W. Recall that the energy $(W \cdot W)$ is a predefined constant E, which is known by everyone. For any $(t-1)$ proxies, even if they collude, they should not know W. Specifically, to any group of $(t-1)$ proxies, any vector W' satisfying $(W' \cdot W') = E$ is a possible candidate for the secret watermark.

On the other hand, after detection, from the n sets of data $S_i(c)$'s obtained, the verifier should not know W. Specifically, to the verifier, any vector W' satisfying $(W' \cdot W') = E$ and $(J \cdot W') = c$ is a possible candidate for the secret watermark.

In general, any $(t-1)$ individuals, including the proxies and verifier, should not know W. That is, by combining all the data held by them, any vector W' satisfying $(W' \cdot W') = E$ and $(J \cdot W') = c$ is a possible candidate for the secret watermark.

S2. Dishonest owner during registration. During registration, instead of honestly sending $S_i(W)$ to the proxies, an owner may send other values so as to mislead the proxies to give high correlation value during detection. Here is an example of dishonest owner: the dishonest owner chooses a watermark w and embeds it into images according to the scheme, but registers with the proxies another watermark Aw where A is a very large constant. Thus, in the subsequent detections, whatever correlation value determined will be based on Aw, instead of w. Owe to the large value of A, the probability that a randomly chosen image is wrongly declared as watermarked is higher. Therefore, we require that, after registration, with at most $(t-1)$ dishonest proxies, any malicious behaviour of the owner can be detected. Specifically, the proxies can check that, indeed, $(W \cdot W) = E$.

S3. Dishonest proxies during detection. During detection, some of the proxies may collude so as to mislead the verifier. Thus, we require that, if at most $(t-1)$ proxies are dishonest, the verifier can detect that.

S4. Collusion among proxies and the owner during detection. A more interesting case is collusion among the owner and proxies. The owner may collude with a few proxies and mislead the verifier. The main difference of this case from the case in previous paragraph is that: here, the owner can reveal the watermark to the proxies. With this extra information, it is easier for the proxies to influence the detection. Thus, we require that, if the owner colludes with at most $(t-2)$ proxies (so the total number of dishonest individuals is at most $t-1$), the verifier should be able to detect that.

S5. Collusion among the owner, proxies and a verifier. It is interesting that we should consider collusion among dishonest owner and verifier. The owner may collude with the verifier and a few proxies, so as to obtain the $S_i(W)$ held by an honest proxy T_i. After obtaining the information, they can use it to influence subsequent detections. Thus we require that, by combining information from a verifier, $(t-3)$ proxies, and the owner, no sufficient information on data held by honest proxies can be derived.

Remarks. Note that currently we do not consider verifier/proxies who keep the history of communications. For example, a verifier who probes the proxies by sending in a series of images. This type of attacks is generally known as sensitivity attacks. We will address this in Section 2.1.8.

Requirement on Error-Correcting

A secondary requirement is error-correcting capability. More specifically, even if some proxies are dishonest or failed, detection can still be carried out. Note the difference between security and error-correcting capability. A scheme that immediately shuts down when malicious activities are detected is considered to be secure, but it is not capable of correcting errors. We say that the *error-correcting threshold* of a scheme is R, if all detection operations can be carried out when there are at least R honest proxies.

2.1.4 Backgrounds on Secret Sharing Schemes

A (t, n) secret sharing scheme splits a secret into n *shares*, which are distributed to n servers respectively. The knowledge of any $t-1$ shares will not reveal any information about the secret but the secret is reconstructible by putting together any t shares. Shamir gave such a scheme in 1979 [44]. For a secret $z \in \mathbb{Z}_p^*$, where p is a public large prime, the share for the i-th server is $f(i) \pmod p$ where $f(x)$ is a random polynomial of degree $(t-1)$ whose free coefficient $f(0) = z$. No individual server knows the coefficients of $f(x)$, thus any $t-1$ servers can not derive z from their shares. However, if t servers put their shares together, they can solve for the coefficients of $f(x)$ and thus reconstruct the secret z.

Shamir's scheme can be modified to achieve useful properties. For example, the schemes can be verifiable [12, 26] and proactive [30]. It is also known that certain arithmetic operations of the secrets can be performed on their shares, such that the shares of the result can be obtained without revealing any of the secrets [7, 31]. In our scheme, we mainly make use of arithmetic operations on the shares. Proactive schemes can also be employed to enhance security.

Notations on Secret Sharing. A secret is an integer in $\mathbb{Z}_p^*$ where p is a prime. In this section, all arithmetic operations (multiplications and additions) performed are followed by modulo p. Thus, for simplicity, we omit the notation (mod) when writing an arithmetic expression. For example, we simply write $z_1 + z_2 z_3 \pmod p$ as $z_1 + z_2 z_3$.

With respect to a secret sharing scheme, let $S_i(z)$ be the i-th share of z, the secret [2]. Let $S(V)$ be the i-th share of a vector $V = (v_1, v_2, \ldots, v_m)$. Note that the "secret" in the secret sharing scheme is an integer, whereas the watermark W and image I are vectors. To compute $S(V)$, we treat each v_i as an independent secret. That is, $S_i(V) = (S_i(v_1), S_i(v_2), \ldots, S_i(v_m))$. Since the shares are associated with the proxies, we also call $S_i(V)$ the share of V for the proxy T_i.

Arithmetic Operations on Shares. Consider two secrets α and β, which are encoded by $f(x)$ and $g(x)$ respectively as in Shamir's scheme, and the shares are distributed to $2t-1$ servers. Now, suppose the servers want to compute the shares of $\alpha+\beta$ without revealing the values of α, β or $\alpha+\beta$. This can be easily done by instructing each server to locally construct the new share by adding the two shares it holds.

The shares for $\alpha\beta$ can be computed similarly by computing $s_{i,1}s_{i,2}$, which is the share of $\alpha\beta$ encoded by $k(x) = f(x)g(x)$. However, the degree of $k(x)$ is raised to $2t-2$, thus $2t-1$ servers are required to reconstruct the secrets.

In our application, instead of general combinations of multiplications and additions, we require only inner products. That is, given the shares of $x_1, x_2, \ldots, x_m$, and $v_1, v_2, \ldots, v_m$, we want to compute the shares of the inner product $c = \sum_{i=1}^{m} x_i v_i$, without revealing the secrets $x_1, x_2, \ldots x_m$,

[2] Note that the shares are computed based on some randomly chosen numbers. Thus to be more precise, we should write $S_i(R, z)$ for the share where R is the chosen sequence of random numbers. For simplicity, we omit R in the notation.

$v_1, v_2, \ldots, v_m$, and the inner product c. For each server, its share of c can be easily computed locally by simply computing the summation of products on its shares of x_i, v_i's.

2.1.5 Watermark Detection Using Secret Sharing

The multiple proxies setting naturally suggests the use of secret sharing as a basic construction block. Assume that there are n proxies. Suppose the security threshold we want to achieve is t, that is, if not more than $(t-1)$ individuals collude, the security is maintained. We also require that $(2t-1) \leq n$. We choose a (t,n) secret sharing scheme where $(2t-1) \leq n$. Recall that n is the number of proxies, and $(t-1)$ is the number of dishonest individuals the system can tolerate.

The scheme consists of two parts: registration and detection.

Registration

§1. ***Distributing watermark.*** The owner, using the secret sharing scheme (with the notations defined in Section 2.1.3 and 2.1.4), computes $S_i^{(0)}(W)$, the share of W for each proxy T_i, $1 \leq i \leq n$. The owner then sends $S_i^{(0)}(W)$ to T_i secretly for all proxies.

§2. ***Refreshing the shares.*** After receiving the shares from the owner, the proxies refresh the shares of W using the mechanisms described in Section 2.1.4. At the end of this step, each proxy T_i has $S_i(W)$ as a new share of W, and old shares $S_i^{(0)}(W)$'s are discarded.

§3. ***Checking*** W ***is genuine.*** Each proxy T_i computes the value $(S_i(W) \cdot S_i(W))$, and broadcasts it to other proxies. Note that this value is also the share of the inner product $(W \cdot W)$ for each proxy. After receiving all data from other proxies, each proxy reconstructs $(W \cdot W)$ and confirms that indeed $(W \cdot W) = E$. If not, the registration fails.

The detection is initiated by a verifier. The verifier wants to know whether an image J is embedded with the watermark claimed by the owner.

Detection

§1. ***Distributing the image.*** The verifier computes the shares of J and sends the shares to the respective proxies.

§2. ***Computing the shares of the correlation value.*** Each proxy T_i computes the inner product $(S_i(J) \cdot S_i(W))$ and sends it back to the verifier.

§3. ***Reconstructing the correlation value.*** After receiving all the shares, the verifier reconstructs the correlation value $(J \cdot W)$. Recall that $(2t-1)$ shares are necessary and sufficient for reconstruction. Therefore, if $(2t-1) = n$, there is only one possible way to reconstruct such value. Otherwise, $(2t-1) < n$, and there are more than one group of $(2t-1)$ shares. For each group, the verifier reconstructs the value.

§4. ***Checking for corrupted data.*** Since the error-correcting threshold is $(2t-1)$, all proxies must be honest if $(2t-1) = n$. In this case, the only value the verifier reconstructed must be correct. Otherwise, the verifier checks whether there is any inconsistency among the values reconstructed from different groups of $(2t-1)$ shares. If so, it declares that some proxies are cheating.

To enhance security, the proxies can refresh their shares regularly. This is to guard against sensitivity attacks. We will revisit these issues later.

2.1.6 Security Analysis

We want to show that the proposed scheme satisfies the requirements stated in Section 2.1.3. The requirements are generally in this form: if at most $(t-1)$ individuals collude, then either no extra information on W is revealed, or no sufficient information is revealed so that the colluders can manipulate the results.

In the following analysis, we treat each share as an equation, where the unknowns are the random numbers used to generate the share. To illustrate, consider a proxy T_i who is holding the share $S_i^{(0)}(W)$, and it wants to guess the watermark W. This share is generated by the owner using $(t-1)m$ random numbers (note that W is a vector of m coefficients). For example, let us consider only the first coefficient w_1, the proxy T_i can expresses what it has as the equation

$$S_i^{(0)}(w_1) = w_1 + r_1 i + r_2 i^2 + \ldots + r_{t-1} i^{t-1}$$

where $w_1, r_1, \ldots, r_{t-1}$ are the unknowns. If a proxy manages to gather t such equations or more, he would be able to solve for w_1. Otherwise, **any** value is a possible candidate for w_1.

Note that the security is achieved unconditionally. That is, even if the colluders have infinite computing power, they can not compute the secret. This is in contrast to schemes that are *computationally secure*, where the security is based on the assumption that certain problem is computationally difficult. We will omit the details for security requirements ***S4*** and ***S5***.

S1. Secrecy of W***.*** First, we investigate the case where all the $(t-1)$ colluders are proxies. Without loss of generality, let the colluders be the proxies $T_1, T_2, \ldots, T_{t-1}$. Note that each T_i has the shares $S_i(W)$ and $S_i^{(0)}(W)$, and all proxies know $S_j(E)$ for all j, and that $(W \cdot W) = E$. Now, we want to know whether combining the information from $(t-1)$ proxies will reveal additional information on W.

Let us consider only $S_i^{(0)}(W)$ first. For each coefficient w_i in vector W, $t-1$ random numbers are used to generate the shares. For w_i, there are t unknowns. On the other hand, the corresponding entry in each $S_i^{(0)}(W)$ proxy T_i possesses is equivalent to 1 equation. Therefore, for $t-1$ proxies, there are only $t-1$ equations, and any integer (in $\mathbb{Z}_d^*$) is a possible solution for w_i, as shown in [44]. This shows that these equations do not give the proxies any advantages in computing W. Furthermore, the new shares $S_i(W)$ are obtained after refreshing, and no information about W or $S_i^{(0)}(W)$ is exchanged among the proxies during this process. Therefore these values do not give any advantages in computing W either. Lastly, after computing $S_j(E) = S_i(W) \cdot S_i(W)$, information about the elements of $S_i(W)$ is hidden in the inner product, given that m is sufficiently large (which is true for most practical applications).

Next, suppose $(t-1)$ colluders are the verifier and $(t-2)$ proxies $T_1, \ldots, T_{t-2}$. The verifier knows the shares $S_i(c)$ for all i, where $S_i(c) = S_i(W) \cdot S_i(J)$. Similar to the above argument, given sufficiently large m, the information contained in the inner product is useless in attempts to obtain W.

S2. Dishonest owner during registration. The owner could be dishonest and try to mislead the proxies so that they give false results in the detection. For instance, he could give a false watermark with high energy, so that the correlation value of the watermark and any randomly chosen image would be large with high probability. This is prevented in the registration because the proxies compute the energy of the watermark and compare it to a known constant E (Step §3 in registration). If the energy is not E, the watermark would be rejected by the proxies, and the registration would fail.

S3. Dishonest proxies during detection. The proxies could also send false results of their inner products $S_i(W) \cdot S_i(J)$ to mislead the verifier. However, since we have more than $2t - 1$ proxies in the system, we can perform reconstruction of $J \cdot W$ multiple times from different set of shares (Step §4 in detection). It is unlikely that the results are consistent if some proxies cheat. In this case, we can employ the method mentioned in Section 2.1.7 to both detect and correct the error.

2.1.7 Analysis on Error-Correcting

Besides the security requirement that no more than $t - 1$ proxies collude, we also require that the verifier can detect errors from proxies and correct them if there are at least $(2t - 1)$ honest proxies [3].

Here are two methods of error correction. The first method is to let the verifier compute a new image $J' = kJ$, where k is some integer chosen by the verifier. If the proxies are honest, the resulting correlation value $c' = J' \cdot W = k(J \cdot W)$ would have the integer k as its factor. If some proxies are dishonest, it is highly unlikely that the reconstructed correlation will still have k as its factor.

The other method let the verifier repeat the detection using the same image J, but using different random numbers to generate the shares of J. Thus, a group of $2t - 1$ proxies would be able to give consistent results only if all of them are honest. By repeating the detections, the correct results

[3] If an accidental error happens, say, during network transmission from a proxy to the verifier, we consider it as a dishonest behaviour (of the proxy).

can be obtained with arbitrarily high probability. It is noted that the above two methods can be used together.

2.1.8 Sensitivity Attacks

A dishonest verifier might probe the proxies for the watermark. By designing the probes carefully, it may be able to get a good approximation of, or erase, the watermark, using small numbers of probes. This is generally known as sensitivity attacks. Some general attacks are given in [16, 35]. Practical attacks usually target at the image representation. For example, the well-know Stir-Mark provides a list of attacks [38].

We classify these attacks into two types. The first type is specific to our proposed scheme and not applicable to others schemes, for instance the zero-knowledge detector [3]. The second type of attacks are designed for general public watermarking schemes. For example, the attacks described in [16, 38, 35]. In this analysis, we focus on the first type. Further research is required to handle the second type of attacks.

Let us consider a dishonest verifier. The verifier may collude with $(t-2)$ proxies in attempt to get the secret watermark. Let $S = (s_1, s_2, \ldots, s_m)$ be the shares of W kept by an honest server. In each detection, the verifier knows the inner product of $S \cdot V$ where V is some vector chosen by the verifier. Although knowing $S \cdot V$ will not reveal any useful information of the watermark W, by sending in many different vectors, the verifier can determine S. If the verifier knows the inner product of $S \cdot V_i$ for $i = 1, \ldots, m$ and the V_i's are independent, then the verifier can solved for S. By knowing the shares in t proxies, the verifier can solve for W. Note that this attack is specific to our scheme. To prevent this, we can require the proxies to refresh their shares regularly, for example, after every $m - 1$ detections.

2.1.9 Communication Cost

We measure the communication cost by the number of rounds of communication and the amount of data transmitted. The size of a coefficient is not more than $\lceil \log p \rceil$ bits, where p is the prime used in the secret sharing scheme. Note that we only require $p > n$ and p is larger than the range of the original image coefficient. Thus, it is not required to be very large. In contrast, for the zero-knowledge detector in [3], the size of one coefficient has to be large (for e.g., more than 200 bits), so that it is computationally infeasible to break the commitment scheme.

Let us assume that the size of each coefficient is 1 unit. Thus, the size of W and J is m. The size of each share is also m. During detection, the verifier sends the share $S_i(J)$ to each proxy T_i, and T_i returns the share $S_i(c)$ of the correlation value. Thus, only 1 round of communication is required. Since the size of each share $S_i(J)$ is m, and the size of each share $S_i(c)$ is 1, the total amount of data transmitted is $(mn + n)$. The zero-knowledge detector in [3] invokes an interactive proof protocol during detection. Due to the "probabilistic" nature of interactive proof, many rounds are required for high level of confidence.

Higher communication cost is required during registration. This is due to the communication required in refreshing. Fortunately, registration is only performed once for each watermark. During registration, refreshing without verification can be done in 1 round with mn^2 units of data. If verification is required, then the communication cost depends on the commitment schemes and the interactive proof protocol employed.

2.2 Scenario 2: Oracle Attacks

2.2.1 Scenario Setting

In this scenario (Figure 2.3), we assume that there is a publicly available watermark detector, anyone can send queries to the detector with any image, and the detector will answer `1` if the the image is watermarked by a secret watermark known only to the detector, and `0` otherwise.

The goal of an attacker is the following: Given a watermarked image $\widetilde{I}$, find a non-watermarked image I', such that I' is close to $\widetilde{I}$, with as few queries sent to the detector as possible. We study the security of a class of watermarking schemes for binary sequences, which is measured by the minimum effort needed by the attacker.

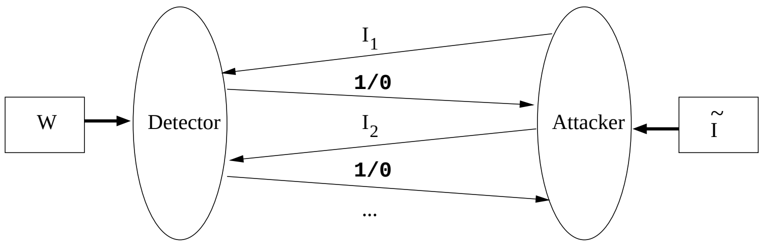

Figure 2.3: Scenario 2: Oracle attacks

2.2.2 Introduction

Under the above setting, Cox et al [16] give a heuristic for general watermarking schemes and an estimated number of queries required. The well-known Stir-mark [38] provides a list of practical attacks, many of which are based on image properties. Here, we view the attacks as games between the attacker and the watermarking scheme. We focus on a few schemes for binary sequences, and take the Hamming distance as the measure.

Our problem is related to the Twenty Questions Game proposed by Ulam in 1976 [46]. In the original game, a target is an integer between 1 and 2^{20}, and a player is to guess the target by asking twenty questions. There are several variants of the Twenty Questions Game since then. For example, the Twenty Questions Game with Genes[39], and [5, 21]. We give a variant that corresponds to the game between the watermarking scheme and the attacker. In this game, the player corresponds to the attacker of the watermarking scheme, and the player's questions correspond to the queries sent to the detector. We give a randomized player who uses expected $O(d(1 + \log(n/k)))$ questions, where d and k are parameters of the game. This player is optimal when $k = o(n)$. The number of queries required by the attacker can serve as a measure of watermarking security. This can be traded-off with the requirements on false alarm and distortion.

In Section 2.2.3, we describe the notations used in this section. We describe a watermarking scheme in Section 2.2.4. In Section 2.2.5, we focus on the Twenty Questions Game. We first give a lower bound in Section 2.2.5, followed by the randomized player in Section 2.2.5, and how the game relates to the original watermarking problem in Section 2.2.5. In Section 2.2.6, we give a few variations.

2.2.3 Notations and Models

A watermarking scheme consists of an *encoder* and a *detector*. The encoder of a watermarking scheme takes a sequence $I = \langle a_1, a_2, \ldots, a_n \rangle$ as input and gives an encoded sequence $\widetilde{I}$. Let $\mathcal{K}$, the *kernel*, be the set of all possible encoded sequences. The encoder satisfies the *distortion* constraint, which requires the distance of $\widetilde{I}$ from I to be bounded by a predefined distortion ϵ. In the other end, the detector takes a sequence as input and outputs a 1 or 0 indicating whether the sequence is watermarked. Let $\mathcal{W}$ be the set of all watermarked sequences. The detector satisfies the constraint on the *false alarm* ratio F, that is, the probability of a randomly selected sequence being watermarked is bounded by F. If the underling distribution is the uniform distribution, then $F = 2^{-n}|\mathcal{W}|$. Besides the above constraint, the encoded sequence $\widetilde{I}$ should remain watermarked even under attacks. There are many different models of attacks. Many previous works, e.g., [15, 11, 13], focus on the robustness of the scheme. We consider smart attackers in this scenario.

2.2.4 A Watermarking Scheme

Now we describe a class of watermarking schemes for binary sequences of length n. This watermarking scheme is analogous to that in [10]. Each scheme is parameterized by the integers d, ϵ and k. The value of d, ϵ and k is made known to the public, including the potential attackers. What are kept secret by the encoder is a secret *key* K and a secret *codebook* C. The codebook C is a collection of *codewords*, which are binary sequences of length k. The codebook satisfies the distortion requirement ϵ in the sense that the distance between any two codewords is at most $2\epsilon + 1$. The secret key $K = \{h_1, h_2, \ldots, h_k\}$ is a set of k indices, where $1 \leq h_i \leq n$ for all $1 \leq i \leq k$. For a sequence I, call the sequence $\langle a_{h_1}, a_{h_2}, \ldots, a_{h_k}\rangle$ the *watermarking coefficients* of I. Given a sequence I to be watermarked, the encoder quantizes the watermarking coefficients of I to the nearest codeword in C. For example, if $\langle a_1, a_2, \ldots, a_k\rangle$ is the watermarking coefficients, and $\langle a'_1, a'_2, \ldots, a'_k\rangle$ is the codeword in C that is nearest to $\langle a_1, a_2, \ldots, a_k\rangle$, then the watermarked sequence $\widetilde{I}$ is the same as I except its watermarking coefficients are replaced by $\langle a'_1, a'_2, \ldots, a'_k\rangle$.

In the other end, the detector declares a sequence I to be *watermarked* if and only if the watermarking coefficients are within a distance d from a codeword in C. Thus, the kernel $\mathcal{K}$ of this scheme consists of sequences whose watermarking coefficients are in C, and the watermarked sequences $\mathcal{W}$ are all the sequences whose watermarking coefficients are within a distance of d from C.

Define $V_{N,R}$ to be the volume of a sphere in N-dimensional space with radius R, where the distance is measured as Hamming distance. That is, $V_{N,R} = \sum_{i=0}^{R} \begin{pmatrix} N \\ i \end{pmatrix}$. The false alarm F satisfies the following bound,

$$F \geq \frac{V_{k,d}}{V_{k,\epsilon}}. \tag{2.1}$$

The equality holds if and only if C is an ϵ-error-correcting perfect code. In this case, the distortion D is:

$$D = \epsilon. \tag{2.2}$$

For $k \gg d \gg \epsilon$, the right-hand-side in (2.1) is approximately $k^{d-\epsilon}$. Note that the false alarm (2.1) and distortion (2.2) do not depend on the size n. The size n plays an important role in security. To see how the security requirement affects the choice of d and k, let us assume that low false alarm and small distortion are the only desirable properties. Then, with fixed distortion, k should be as large as possible and d should be 0. Since $d = 0$, the watermarked sequences are isolated "points" in $[0,1]^n$. This amounts to finding a good source code for the binary sequence. By bringing in the security requirement, each sequence in the kernel should be surrounded by watermarked sequences. If not, an attacker can easily find a non-watermarked sequence by random perturbation. Intuitively, d should be as large as possible to enhance security. However, larger d will raise the false alarm (from (2.1)). Thus an important question is how to choose d and k for given requirements of false alarm, distortion and security. Next section gives an analysis on security that provides a trade-off for the watermarking requirements.

2.2.5 Twenty Questions Game with Watermark Attacker

Before we describe a watermark attacker, let us consider this guessing game involving a *player* and a *target*. The target K is a set containing k integers from $U = \{1, 2, \ldots, n\}$. The player knows the size of K and U before the game starts. The goal of the player is to determine at least $d+1$ elements in K, using as few queries as possible. A query is represented by a set $Q \subseteq U$. The outcome of a query Q, denoted by $\mathcal{Q}(Q)$, is `Yes` if and only if

$$|Q \cap K| > d.$$

This game can be considered as a variant of the Ulam's game [46], and is similar to the Twenty Questions Game with Genes in [39]. In the Twenty Questions Game with Genes, the query is of the form "does a given interval contain an integer from K". The goal is to reconstruct K using as few queries as possible. The lower bound for a deterministic player of the Twenty Questions Game with Genes is $\log \begin{pmatrix} n \\ k \end{pmatrix}$, which is approximately $k \log(n/k)$ for $k \ll n$.

Our game differs from the Twenty Questions Game with Genes in a few ways. Our player has an easier job because he only needs to determine $d+1$ elements in K. On the other hand, our queries are more general, and thus might provide less information.

Lower Bound

A lower bound for any deterministic player in our game is

$$\log\left(\binom{n}{d+1} \Big/ \binom{k}{d+1}\right). \tag{2.3}$$

In the guessing game, the player wins if he can identify $d+1$ elements in the target K. Before the game starts, from the player point of view, all the $\binom{n}{k}$ sets of k elements are possible targets. This class of possible targets reduces as the player asks questions. When all the possible targets contain $d+1$ common elements, the player can confidently outputs these $d+1$ elements and wins the game.

Let us look at the decision tree where each node is a class of possible targets. Thus, the root is the class of size $\binom{n}{k}$. In the best scenario for the player, each leaf is a class with largest possible number of targets, which is $\binom{n-(d+1)}{k-(d+1)}$ (this is the number of possible targets where $d+1$ elements are fixed). Therefore, the height of the tree is at least

$$\log\left(\binom{n}{k} \Big/ \binom{n-(d+1)}{k-(d+1)}\right),$$

which is equal to (2.3). This gives the claimed lower bound. Note that the bound is in $\Omega(d\log(n/k))$, and for small k and d, the bound is approximately $d\log(n/k)$.

By assigning each node with equal probability, and using the Yao's principle[48], we can also show that any randomized player requires expected

$\Omega(d \log(n/k))$ questions.

A Player (Deterministic and Probabilistic)

The job of a player is to identify at least $d+1$ elements in K. Our strategy is to first find a small subset $U_0 \subset U$ that contains at least $d+1$ elements in K. Next, the size of U_0 is gradually reduced in a way similar to binary search, until its size becomes $d+1$, which is what we want. To find the small U_0, the deterministic player uses step §1 in the algorithm below. However, this step requires $(k-1)/d-1$ queries in the worst case. The randomized player improves this step to expected constant number of queries by first shuffling the coefficients (§0).

Deterministic Algorithm. Here we present a deterministic algorithm for the guessing game.

§1. Divide U evenly into $(k-1)/d$ groups, $U_1, U_2, \ldots, U_{(k-1)/d}$. Find an i such that $\mathcal{Q}(U_i)$ gives `Yes`. Let $Q_0 = U_i$.

§2. Divide Q_0 evenly into $2d+2$ groups, $G_1, G_2, \ldots, G_{2d+2}$. Let $L = \phi$ and $G_0 = \phi$, where ϕ is the empty set.

§3. Find the largest $i \in \{0, 1, 2, \ldots, 2d+2\}$ such that $\mathcal{Q}((G_0 \cup G_1 \cup G_2 \cup \ldots \cup G_i) \cup L)$ gives `No`. Update L to be $L \cup G_{i+1}$. Repeat step §3 until no such i exist.

§4. Update Q_0 to be L. If Q_0 contains only $d+1$ elements, Q_0 is the result. Otherwise repeat from step §2.

By the pigeon-hole principle, there exists one group U_i in step §1 that contains at least $d+1$ elements from K, and $\mathcal{Q}(U_i)$ gives `Yes`. Therefore, the number of queries needed for this step is at most $(k-1)/d-1$.

Since each U_{i+1} identified in step §3 contains at least 1 element from K, the repeat-loop in step §3 repeats for at most $d+1$ rounds before $\mathcal{Q}(L)$ gives `Yes`. Therefore, step §2 to §3 identify at most $d+1$ groups among $G_1, \ldots, G_{2d+2}$, which in total contain at least $d+1$ elements from K. It follows that the size of L is at most $|Q_0|/2$. These two steps can be completed using a total of $2d+2$ queries.

Step §2 to §4 are repeated until $|Q_0|$ is reduced to $d+1$. Thus, the total number of rounds is at most $\max(1, \log(n/k))$ and the total number of queries required to complete the outer-loop is $O\left(d(1+\log(n/k))\right)$.

In the worst case, the number of queries needed by the player is $O(k/d + d(1+\log(n/k)))$.

Randomized Algorithm. When k is small, the above is dominated by the term $d\log(n/k)$. However, if k is large, the term k/d would dominate, which is undesirable. Now we introduce a probabilistic player, who uses expected $O(d(1+\log(n/k)))$ queries.

§0. Permutes the set U uniformly at random.

This probabilistic player performs step §0, and then proceeds from step §1 of the deterministic player.

Recall that the size of a group U_i in step §1 is $dn/(k-1)$. Since the input U is randomly shuffled in step §0, each element in U_i has the probability k/n to be from K. Let Z be the number of elements in U_i that are from K. Then the expected value of Z is $E(Z) = dk/(k-1)$. Since $d < dk/(k-1) < d+1$, the probability $Pr[Z \geq (d+1)] = Pr[Z > E(Z)]$, which is greater than some constant that is approximately $1/2$.

Since we are doing selection without replacement in step §1, if the group we select contains less than $d+1$ elements from K, the following groups would have greater probability to contain at least $d+1$ elements from K.

Thus, step §1 can be completed in expected $O(1)$ queries. This gives expected $O(d(1+\log(n/k)))$ for the randomized algorithm. When $k = o(n)$, we have a tight $O(d \log(n/k))$ algorithm.

A Watermark Attacker

For a set X of indices, let I_X be the sequence whose i-th coefficient is 1 if and only if $i \in X$. Given a sequence I and a (n, k, d, ϵ) scheme, the task of the attacker is to find a non-watermarked sequence I' such that $\|I' - I\| \leq d+1$. The attacker knows the values of n, k, d, and ϵ. What he does not know is the codebook and the secret key K. Here, we assume that the codebook is a perfect binary code.

Without loss of generality, we can assume that the given sequence I consists of only 0's, that is $I = \langle 0, 0, \ldots, 0 \rangle$, and the codebook contains $\langle 0, 0, \ldots, 0 \rangle$. Now, it suffices for the attacker to find a set of indices X such that $|X| = d+1$ and $X \subseteq K$. Since $|X \cap K| = d+1$ and C is a perfect code, I_X is non-watermarked.

The watermark attacker corresponds to the player in the Twenty Questions Game in Section 2.2.5, the secret key K corresponds to the target, and the detector corresponds to the query. The sequence I_X is watermarked if $\mathcal{Q}(X)$ gives `No`. Note, however, the two problems are not completely equivalent. Consider a X' where $|X' \cap K| > d$. It is possible that $I_{X'}$ is still watermarked, although $\mathcal{Q}(X')$ gives `Yes`. However, the number of such X' is insignificant comparing to the number of $\tilde{X}$ where $|\tilde{X} \cap K| > d$ and $I_{\tilde{X}}$ is not watermarked.

Trade-off with False Alarm and Distortion. For a given false alarm F and distortion D, we want to know how to choose d, k, and ϵ to achieve the highest security. By taking the approximate lower bound on the number of calls to the detector required as a measure of the security S,

$$S = \log\left(\binom{n}{d+1} \Big/ \binom{k}{d+1}\right), \tag{2.4}$$

combining with the equation for false alarm (2.1) and distortion (2.2), we can determine the right parameters. For simplicity, use the approximations $S \approx d \log(n/k)$ and $F \approx k^{d-\epsilon}$. Together with (2.2) and (2.4), it can be shown that S has the maximum value

$$\begin{aligned} S_{max} &= (\sqrt{D \log n} - \sqrt{\log F^{-1}})^2 && (2.5) \\ \text{when } d &= D - \sqrt{D \log_n F^{-1}}. && (2.6) \end{aligned}$$

We can trade-off the requirements on security, the false alarm (2.1) and the distortion (2.2). For instance, in order to make the watermarking scheme more secure, we can increase the value of d or decrease the value of k. But by doing that, we will increase the false alarm F. In order to compensate for F, we can also choose to increase the value of ϵ, which would in turn raise the distortion D. If the requirements for the false alarm and distortion are fixed, we can then use (2.5) to find out the maximum security that we can achieve.

2.2.6 Variations of the Game

In this section we will examine some variations of the game and the corresponding watermarking schemes. These variations try to confuse the player by introducing a liar and multiple targets into the game. However, as we will see, although these mechanisms make the game more difficult, they degrade the performance on false alarm and distortion. In the overall trade-off, they do not improve the security.

Twenty Questions Game Between Watermark Attacker and Liar

The Twenty Questions Game with watermark attacker can be extended to a game with a liar. That is, with some constant probability $p < 1/2$, the answer to the query would be wrong. The error can be two-sided: a type-1 error with probability p_1, when $|Q \cap K| > d$ but the answer is `No`; and a type-2 error with probability p_2, when $|Q \cap K| \leq d$ but the answer is `Yes`.

If $p_2 = 0$, our algorithm will still give a correct solution. However, because of the effect of p_1, the expected number of groups identified in step §2 and §3 will be increased to $(d+1)(1+p_1)$. So the factor by which U_0 is reduced is not $1/2$ but $(1+p_1)/2$. Thus the expected cost of our randomized algorithm will be increased by a constant factor $1/(1-\log(1+p_1))$, but is still $O(d\log(n/k))$. To take p_2 into consideration, we need to slightly modify step §3 as the following.

§3. Find the largest $i \in \{0, 1, 2, \ldots, 2d+2\}$ such that $\mathcal{Q}((G_0 \cup G_1 \cup G_2 \cup \ldots \cup G_i) \cup L)$ gives `No`. Update L to be $L \cup G_{i+1}$. Repeat step (3) until no such i exist. If $L = Q_0$, stop with no solution.

Now our algorithm becomes a Monte Carlo algorithm, which gives a correct solution with certain probability. Obviously, if no errors occur in all the queries, the result would be correct. The probability of such cases is $P = (1-p)^{c_1 d \log(n/k)}$, where c_1 is some positive constant.

If we repeat the same query for T times and take the majority answer, the new probability of error $p' < e^{-c_2 T}$, for some positive constant

c_2. Now the probability for our algorithm to give a correct solution is $P = (1-p')^{c_1 d \log(n/k)}$, which is approximately $1 - p' c_1 d \log(n/k)$ for small p'. Let $p' c_1 d \log(n/k) < e^{-c_2 T} c_1 d \log(n/k) < 1/2$, then $T > (1/c_2) \ln(2 c_1 d \log(n/k))$. Thus for $P > 1/2$, the expected number of queries required by our algorithm is $O(d \log(n/k) \log(d \log(n/k)))$. Therefore, by repeating the algorithm for an expected constant number of times, we will have a correct solution.

The liar in the Twenty Questions Game corresponds to a detector that gives a wrong answer in the watermarking scheme. With probability p_1, the sequence is not watermarked but the detector says that it is; with probability p_2, the sequence is watermarked but the detector says that it is not. We can see that in practice p_2 should be negligible, otherwise we could just randomly select a sequence near the watermarked one, and make the detector say that it is not watermarked by repeatedly sending the sequence to it. Because of p_1, the false alarm F will be increased to $F' = F + p_1$. In order for p_1 to be significant enough to our algorithm, p_1 has to be greater than $c_3/d \log(n/k)$, for some constant c_3. However, since the original false alarm $F \approx k^{d-\epsilon} \ll 1/d \log(n/k)$, it is very difficult, if not impossible, to compensate for the false alarm by adjusting the values of k and d. Even if we want to do so, the number of queries will increase because of the changes to d and k.

Modified Twenty Questions Game with Multiple Targets

We can also extend the Twenty Questions Game to have two secret sets $\mathcal{K}_1$ and $\mathcal{K}_2$. The answer to the query would be `Yes` if $|Q \cap \mathcal{K}_1| > d$ and $|Q \cap \mathcal{K}_2| > d$, and `No` otherwise. The player is still required to identify more than d elements from $\mathcal{K}_1$.

Interestingly, the algorithm and analysis in Section 2.2.6 are still applicable, with $p_2 = 0$. Therefore it also can be solved in expected $O(d \log(n/k))$ queries. This variation can be easily extended further to more than two secret sets, where different secret sets may have different values of d and k. However, those variations will not make the game more difficult.

The corresponding watermarking scheme would have multiple codebooks, and only use one of them to watermark a sequence. The choice of the codebook to be used can be random, or based on sequence specific information, such as the nearest distances from the codewords of each codebook. Similar to the watermarking scheme in Section 2.2.6, the false alarm F increases significantly due to p_1, and the number of calls to the detector increases if we want to compensate for F.

2.2.7 Remarks

We have explored other watermarking schemes on binary sequences. It turns out that the simple watermarking scheme in Section 2.2.4 outperforms them. This leads to a general question: given the requirements on false alarm and distortion, what is the highest security (measured in term of number of calls to the detector) we can achieve. We do not know the solution to this general question. We suspect that the security of a watermarking scheme is closely related to the *critical distance*, that is, the radius of the smallest sphere centered at the kernel, whose surface contains roughly half watermarked sequences. Note that our randomized player given in Section 2.2.5 uses this distance to obtain the set U_0. We also do not know any non-trivial bound of this distance with a given false alarm and distortion. Many interesting problems remain open.

2.3 Scenario 3: Inversion Attacks

2.3.1 Scenario Settings

Under this scenario (Figure 2.4), Alice has the original image I and a secret watermark $W_{\mathtt{A}}$. She releases the watermarked image $\widetilde{I} = I + W_{\mathtt{A}}$ into the public domain. Given $\widetilde{I}$ and not knowing $W_{\mathtt{A}}$, Bob (who is an attacker) wants to find a watermark $W_{\mathtt{B}}$ that is present in both $\widetilde{I}$ and I. If such a watermark $W_{\mathtt{B}}$ is found, Bob can create confusion of the ownership by claiming that: (1) $\widetilde{I}$ is watermarked by his watermark $W_{\mathtt{B}}$, and (2) the image $I' = \widetilde{I} - W_{\mathtt{B}}$ is the original. If Bob can successfully and efficiently find such $W_{\mathtt{B}}$, we say that the scheme is *invertible*.

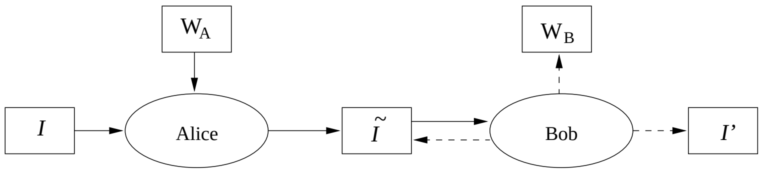

Figure 2.4: Scenario 3: Inversion attacks

In this section we give a construction of a non-invertible watermarking scheme, and prove its security using well accepted cryptographic techniques.

2.3.2 Introduction

There are many discussions on the uses of watermarking schemes in resolving ownership disputes. An interesting and well-known scenario is the *inversion attacks* studied by Craver et al. [20], which gives an attacker when the underlying watermarking scheme is the well-known spread spectrum method. To overcome such attackers, they propose a protocol that employs a secure hash, and claim that it is non-invertible. Qiao et al. [40, 41] also give watermarking schemes for video and audio which are claimed to be non-invertible.

Subsequently, there are a number of works [42, 1, 2] exploiting weaknesses of known non-invertible schemes. Ramkumar et al. [42] give an attack for the scheme by Craver et al. [20], and they also give an improved scheme. On the other hand, [1, 2] give a formal definition of ambiguity attacks and mention that most proposed non-invertible schemes either do not come with a satisfactory proof of security, or the proofs are flawed. They also point out that if the false alarm of the underlying watermarking scheme is high (for e.g. 2^{-10}), then successful ambiguity attacks are possible. However, there is no mention of cases when the false alarm is low. Thus, it is interesting to know whether non-invertibility can be achieved when false alarm is low. Due to the difficulty of obtaining a non-invertible scheme, [2] propose to use a *trusted third party* (TTP) to issue valid watermarks. Although using a TTP is provably secure, there is still a question of whether it can withstand attackers that probe the system. The development of the studies of non-invertibility seems to lead to the conclusion that a stand-alone (in the sense that there is no TTP) non-invertible scheme does not exist. Here, in contrast, we argue that with low false alarm, it is possible to have a non-invertible scheme. We support our argument by giving a provably secure protocol that employs a *cryptographically secure pseudo-random number generator* (CSPRNG). The main idea is to show that if the scheme is invertible, then the CSPRNG is not secure, and thus lead to a contradiction.

Our protocol requires a computationally secure one-way function, whose existence is a major open problem in computer science. Nevertheless, it is well accepted that such functions exist. In practice, many cryptographic protocols rely on this unproven assumption.

We show that our protocol is secure against *ambiguity attacks*, of which inversion attacks are a special case. Given a work $\widetilde{I}$, a successful ambiguity attack outputs a watermark W that is embedded in $\widetilde{I}$, and a key K that is used to generate W. In a weaker form, the attack is also required to output an original I. In our discussion, we do not require the attacker to do so.

There are two components in our scheme. The first component addresses the issue of robustness, false alarm and distortion. This component is often called the *underlying* watermarking scheme. Due to the theoretical nature of this problem, we adopt the usual assumption that the hosts and noise are Gaussian, and distortion is measured by Euclidean 2-norm. We employ the well-known spread spectrum method as the underlying scheme.

The second component consists of key-management and watermark generation. In our setting, Alice (the owner) has a secret key $K_{\mathtt{A}}$, and she generates a watermark $W_{\mathtt{A}}$ using a CSPRNG with $K_{\mathtt{A}}$ as the seed. Next, she watermarks the original I using $W_{\mathtt{A}}$. To prove the ownership, Alice needs to reveal (or show that she knows) $K_{\mathtt{A}}$ and $W_{\mathtt{A}}$. Interestingly, our scheme does not use the original I to derive the key $K_{\mathtt{A}}$, nor the watermark $W_{\mathtt{A}}$. Hence the watermark is statistically independent from the original. This is in contrast to the method given by Craver et al. [20], where Alice computes the hash of the original I, and uses the hash value $h(I)$ to generate the watermark $W_{\mathtt{A}}$. Hence, to achieve non-invertibility, it is not necessary to enforce a relationship between the watermark and the original work.

We give our main idea of our protocol in Section 2.3.3. We further give precise notations and describe the models that we use in Section 2.3.4. The details of the non-invertible scheme will be given in Section 2.3.5, followed by a proof of security in Section 2.3.6. At the end of this section we give some remarks (Section 2.3.7).

2.3.3 Main Idea

In our scheme, a watermark W is a sequence of -1 and 1 of length n, i.e. $W \in \{-1, 1\}^n$. We call W a *valid watermark* if it is generated by a CSPRNG using some m-bit seed, where $m < n$. Thus, the number of valid watermarks is not more than 2^m, and not all sequences in $\{-1, 1\}^n$ are valid watermarks.

Suppose we have a probabilistic polynomial-time algorithm B such that given any work $\widetilde{I}$ that is embedded using some valid watermark W, B can successfully find a valid watermark $\widehat{W}$ embedded in $\widetilde{I}$ with probability that is not negligible[4].

Now, we want to use B to construct a polynomial statistical test $\mathcal{T}$ that distinguishes a truly random sequence from a sequence generated by the CSPRNG, thus lead to a contradiction.

Given a sequence W, $\mathcal{T}$ carried out the following steps:

1. Embed W in I to get $\widetilde{I}$, where I is a randomly chosen work.
2. Ask B for a valid watermark $\widehat{W}$ embedded in $\widetilde{I}$.
3. Declare that W is from the random source if B fails to find such a watermark, and declare that W is generated by the CSPRNG otherwise.

By carefully choosing parameters for the underlying watermarking scheme, the probability that a valid watermark exists in a randomly chosen $\widetilde{I}$ can be exponentially small.

Hence, if W is generated by the truly random source, then it is very unlikely that a valid watermark exists in $\widetilde{I}$, and thus most of the time, B fails and the decision by $\mathcal{T}$ is correct. On the other hand, if W is indeed generated from the CSPRNG, the chances that a valid $\widehat{W}$ can be found is

[4] W and $\widehat{W}$ can be different.

not negligible since B is a successful attacker. So, with probability that is not negligible, the decision made by $\mathcal{T}$ is correct.

Combining the above 2 cases leads to the conclusion that $\mathcal{T}$ can distinguish the two distributions. This contradicts with the assumption that the pseudo random number generator is secure. Therefore, no such B exists, and the scheme is non-invertible as a consequence.

2.3.4 Notations and Models

Overall Setting

A *work* is a vector $I = \langle x_1, \ldots, x_n \rangle$ where each x_i is a real number. A *watermark* W is a sequence in $\{-1,1\}^n$. A *key* K is a sequence of m binary bits. A watermark generator $f : \{0,1\}^m \rightarrow \{-1,1\}^n$ maps a key to a watermark. We say that a watermark W is *valid* if and only if w is in the range of f, i.e., it is generated from some key K by f.

The *underlying* watermarking scheme consists of an embedder and a detector. Given an original work I and a watermark W, the embedder computes a watermarked work $\widetilde{I}$. Given a work $\widetilde{I}$ and a watermark W, the detector declares whether W is embedded in $\widetilde{I}$, or not.

Before watermarking an original work I, Alice chooses a secret key $K_{\mathtt{A}}$ and generates a watermark $W_{\mathtt{A}} = f(K_{\mathtt{A}})$. Alice then embeds $W_{\mathtt{A}}$ into I. To resolve disputes of ownership, Alice has to reveal both the secret key $K_{\mathtt{A}}$ and the watermark $W_{\mathtt{A}}$. (In zero-knowledge watermarking setting [3, 18], Alice only has to prove that she knows $K_{\mathtt{A}}$ and $W_{\mathtt{A}}$).

In a successful ambiguity attack, given $\widetilde{I}$, Bob (the attacker) manages to find a pair $K_{\mathtt{B}}$ and $W_{\mathtt{B}}$ such that $f(K_{\mathtt{B}}) = W_{\mathtt{B}}$ and $W_{\mathtt{B}}$ is already embedded in $\widetilde{I}$. A formal description of ambiguity attacks will be presented in Section 2.3.4.

It is unreasonable to require a successful attacker to be always able to find the pair K_{B} and W_{B} for every work $\widetilde{I}$. Thus, we consider an attacker *successful* as long as the probability that he succeeds, on a randomly chosen $\widetilde{I}$, is non-negligible (greater than $1/p(n)$ for some positive polynomial $p(\cdot)$). Note that the probability distribution to be used in the definition of a successful attacker is important in the formulation. In Section 2.3.4 we will give more details on this.

We measure computational efficiency with respect to n, the number of coefficients in a work. Thus, an algorithm that runs in polynomial time with respect to n is considered efficient.

Statistical Models of Works and Watermarked Works

In this section, we give the statistical models of works. Recall that a work I is expressed as $I = \langle x_1, \ldots, x_n \rangle$, where each x_i is a real number. We assume that I is Gaussian. That is, the x_i's are statistically independent and follow zero-mean normal distribution. Thus, to generate a random I, each x_i is to be independently drawn from the normal distribution $\mathcal{N}(0, 1)$. Note that the expected energy $E(\|I\|^2)$ is n.

Although the distribution of the original works is Gaussian, the distribution of the watermarked works is not necessarily Gaussian. Consider the process where an $\widetilde{I}_r$ is obtained by embedding a randomly chosen W_r from $\{-1, 1\}^n$ into a randomly chosen original work I. If the embedder simply adds the watermark to the original work, then the distribution of such watermarked work $\widetilde{I}_r$ is the convolution of the distribution of the watermarks and that of the original works, which is not necessarily Gaussian. Let us denote the distribution of $\widetilde{I}_r$ as $\mathbf{X_r}$ and call it the distribution of *randomly watermarked works*.

Now, consider the process where a **valid** watermark W_v is uniformly chosen (by uniformly choosing the key for the watermark generator), and then the watermarked work $\widetilde{I_v}$ is obtained by embedding W_v into a randomly chosen original work I. Let us denote the distribution of such $\widetilde{I_v}$ as $\mathbf{X_v}$, and call it the distribution of *valid watermarked works.*

For clarity in notation, we use the symbol I to denote an original work, and add the tilde $\widetilde{I}$ to denote a work drawn from either $\mathbf{X_r}$ or $\mathbf{X_v}$ [5].

Formulation of Ambiguity Attacks

We follow the formulation of ambiguity attacks given in [2] with slight but important modification.

Let B be a probabilistic polynomial-time algorithm. Given some watermarked work $\widetilde{I}$, we say that B successfully attacks $\widetilde{I}$ if it outputs a pair (W, K) s.t. $\widetilde{I}$ contains the watermark W and $W = f(K)$, or outputs a symbol $\perp$ to correctly declare that such pair does not exist. Let us write $B(\widetilde{I}) = \texttt{PASS}$ when the attack is successful. We denote $\Pr[B(\widetilde{I}) = \texttt{PASS}]$ to be the probability that B successfully attacks a particular $\widetilde{I}$. The probability distribution is taken over the coin tosses made by B. Note that for $\widetilde{I}$ there does not exist such a pair (W, K), B has to output $\perp$ and hence is always successful.

We further denote $\widetilde{\mathcal{I}_n}$ to be a work that consists of n coefficients, and that is randomly drawn from the distribution of valid watermarked works $\mathbf{X_v}$. Let $\Pr[B(\widetilde{\mathcal{I}_n}) = \texttt{PASS}]$ to be the probability that an attack by B is successful. In this case, the probability distribution is taken over the coin tosses made by B, as well as the choices of watermarked $\widetilde{\mathcal{I}_n}$. Then we have

[5]Clearly these two distributions $\mathbf{X_r}$ and $\mathbf{X_v}$ are different. However, by an argument similar to that in Section 2.3.6, it is not difficult to show that these two distributions are computationally indistinguishable.

the

DEFINITION 1 *Let B be a probabilistic polynomial-time algorithm. We say that B is a successful attacker if, there exists a positive polynomial $p(\cdot)$, s.t. for all positive integer n_0, there exists an integer $n > n_0$, and*

$$\Pr[B(\widetilde{\mathcal{I}_n}) = \texttt{PASS}] > 1/p(n).$$

In other words, B is a successful attacker if B successfully output a watermark-key pair with probability that is not negligible.

Note that our definition is a slight modification from [2]. The definition in [2] does not take into account cases where there is no valid watermark in a work. Moreover, the distribution of the watermarked work $\widetilde{I}$ is taken over the random choices of the original works. In our formulation, the watermarked work is drawn from $\mathbf{X}_{\mathtt{v}}$, and we differentiate the case where there are some valid watermarks in the given work from the case where there is not any.

This modification is important. We cannot simply say that an attacker is successful if $\Pr[B(\widetilde{\mathcal{I}_n}) = \texttt{PASS}]$ is high. This is because we observe that, it is possible to design a watermarking scheme such that for a randomly chosen work $\widetilde{I}$, the probability that it does not contain a valid watermark is very high. In that case, a trivial algorithm that always declares "can not find a valid watermark" is correct with high probability, and thus by definition is a successful attacker. Due to this consideration, we decide to consider $\mathbf{X}_{\mathtt{v}}$ in the definition, and separate the two cases where valid watermarks do or do not exist.

Cryptographically Secure Pseudo-Random Number Generator

Loosely speaking, a *pseudo-random number generator* (PRNG) takes a seed of a certain length as input and outputs a string, which is of a longer length than that of the seed.

A *cryptographically secure pseudo-random number generator* (CSPRNG) is a PRNG whose output string cannot be computationally distinguished from a truly random distribution. Formal definition of the security of CSPRNG is done in terms of polynomial statistical tests [49]. We follow a simplified definition of statistical tests used in [8].

Let $\{0,1\}^n$ be the set of binary strings of length n, and $\{0,1\}^*$ denotes the set of all binary strings of all lengths. Formally, we have the following definitions.

Definition 2 *A PRNG g is a deterministic polynomial-time algorithm $g : \{0,1\}^m \rightarrow \{0,1\}^{q(m)}$, for some positive integer m and positive polynomial $q(m)$.*

Definition 3 *A probabilistic polynomial-time statistical test $\mathcal{T}$ is a probabilistic polynomial-time algorithm that assigns to every input string in $\{0,1\}^*$ a real number in the interval $[0,1]$.*

In other words, $\mathcal{T}$ can be considered as a function $\mathcal{T} : \{0,1\}^* \rightarrow [0,1]$, which terminates in polynomial time, and whose output depends also on the coin tosses during execution. Let r_n be the expected output of $\mathcal{T}$ over all truly random n-bit strings drawn uniformly from $\{0,1\}^n$, and all coin tosses made by $\mathcal{T}$. We have

Definition 4 *A PRNG g passes test $\mathcal{T}$ if, for every positive integer t, and every positive polynomial $q(m)$, there exists a positive integer m_0, such that* *for all integers $m > m_0$, the expected output of $\mathcal{T}$, given a $q(m)$-bit string generated by g, lies in the interval $(r_{q(m)} - m^{-t}, r_{q(m)} + m^{-t})$, assuming the seed of g is uniformly distributed over $\{0,1\}^m$.*

If a PRNG g does not pass a test $\mathcal{T}$, we say that $\mathcal{T}$ has an *advantage* in distinguishing g from a truly random source. Then we can define CSPRNG as

Definition 5 *A CSPRNG is a PRNG g that passes every probabilistic polynomial-time statistical test $\mathcal{T}$.*

In other words, no test $\mathcal{T}$ can have an advantage in distinguishing a CSPRNG g from a truly random source.

Here, we employ the CSPRNG due to Blum et al. [8]. A Blum number N is an integer that is the product of two primes, each congruent to 3 (mod 4). Let QR_N be the set of all quadratic residues in $\mathbb{Z}_N^*$. That is, $x \in QR_N$ if and only if there exists an $x_0 \in \mathbb{Z}_N^*$ such that $x_0^2 \equiv x \mod N$. Let $s \in QR_N$ be a seed to the Blum CSPRNG, the i-th bit b_i in the output string is computed as

$$b_i = (s^{2^i} \mod N) \mod 2. \tag{2.7}$$

In other words, we compute the output string by squaring the current number (starting from the seed) to get the next number, and take the least significant bit as the output. Following the above notations, we have the

Definition 6 *A Blum PRNG is a function $g : QR_N \rightarrow \{0,1\}^{q(m)}$ defined as $g(s) = b_0, b_1, \cdots, b_{q(m)-1}$, where $b_i = (s^{2^i} \mod N) \mod 2$, N is a Blum number of length m, and $q(m)$ is a positive polynomial of m.*

It is proved in [8] that, under the well accepted assumption that integer factorization is hard, this PRNG is secure. That is, it passes every polynomial statistical test $\mathcal{T}$. We shall refer to it as the Blum CSPRNG.

2.3.5 A Non-Invertible Scheme

Now, we describe the proposed secure protocol. The parameters for the protocol are three constants T, k and m. In the proof of security, the parameters should be expressed in terms of n. We will choose

$$k = 1/100, \qquad T = nk/2 = n/200, \qquad m = \sqrt{n}. \tag{2.8}$$

Underlying Watermarking Scheme

The underlying watermarking scheme is essentially the spread spectrum method. For completeness and clarity, we describe the embedding and detection processes.

Embedding: Given an original I and a watermark W, the watermarked $\widetilde{I}$ is computed as $\widetilde{I} = I + kW$, where k is a predefined parameter. [6]

Detection: Given a work $\hat{I}$ and a watermark W, declare that $\hat{I}$ is watermarked if and only if $\hat{I} \cdot W \geq T$, where $\cdot$ is the vector inner product and T is a predefined parameter.

False Alarm, Robustness and Distortion (Parameters T and k)

The performance of a watermarking scheme is measured by its false alarm, robustness and distortion. Detailed analysis can be found in [17]. Here, we are more concerned with the false alarm F, which is the probability

[6]For simplicity, we omit normalization in the embedding. Thus, the energy $\|\widetilde{I}\|^2$ of a watermarked work is expected to be higher than the original work. Our proof can be modified (but tedious) when normalization is to be included.

that a randomly chosen $\widetilde{I}$ is declared to be watermarked by a random valid watermark W. That is

$$F = \Pr[\widetilde{I} \cdot W > T] \tag{2.9}$$

where $\widetilde{I}$ is drawn from the distribution of randomly watermarked works $\mathbf{X_r}$, and W is uniformly chosen from $\mathcal{W}$ the set of valid watermarks.

The false alarm F is small. To see that, consider any given $W \in \mathcal{W}$ and $\widetilde{I}$ randomly chosen from distribution $\mathbf{X_r}$, it is not difficult to show that the distribution $(\widetilde{I} \cdot W)$ is a zero-mean normal distribution with standard derivation δ where δ can be analytically derived. If $T = C_0\delta$ where $C_0 > 0$ is some positive constant, then the probability that a random $\widetilde{I}$ satisfies $(\widetilde{I} \cdot W > T)$ is less than $\exp(-C_0^2/2)$. Using the parameters in (2.8), $\delta < 2\sqrt{n}$. Since $T = n/200$, it is many times larger than the standard derivation δ.

For each $W_i \in \mathcal{W}$, where $1 \le i \le |\mathcal{W}|$, let F_i be the probability that $\widetilde{I} \cdot W_i > T$ for random $\widetilde{I}$ from $\mathbf{X_r}$. By the argument above, F_i is exponentially small with respect to n. More precisely, given the parameters in (2.8) and random $\widetilde{I}$ from $\mathbf{X_r}$,

$$F_i = \Pr[\widetilde{I} \cdot W_i > T] = \exp(-d_i n) \tag{2.10}$$

for some positive constant d_i. Therefore,

$$F = \sum_{i=1}^{|\mathcal{W}|} F_i \Pr[W = W_i] \le \exp(-C_1 n) \tag{2.11}$$

where C_1 is the maximum d_i in (2.10), which is a positive constant.

By choosing $k = 1/100$, the distortion introduced during embedding is 1% of the original work. We could also choose k to be a slow decreasing function, for e.g. $k = 1/\sqrt{\log n}$, so that the ratio of the distortion over the energy of the work tends to 0 as n increases. Our proof still holds for this set of parameters.

Similarly, the scheme is very robust. Since the expected inner product of a watermarked image and the watermark is $E[(I + kW) \cdot W] = kn$, a noise of large energy is required to pull the inner product below the threshold $T = kn/2$. In this case, for noise with energy n (i.e. same as the original image), the watermark can still be detected in the corrupted work with high probability.

Watermark Generation (Parameter m)

A watermark is generated using a CSPRNG $f : \{0, 1\}^m \to \{-1, 1\}^n$ where $m \leq n$. Thus, it takes a small seed of m bits and produces a watermark. Note that this CSPRNG can be easily translated from the Blum CSPRNG by mapping the output 0 to -1, and 1 unchanged. Let $\mathcal{W}$ to be the range of the function f, and it is actually the set of valid watermarks. Clearly, $|\mathcal{W}| \leq 2^m$.

Intuitively, for better security, we should have large m so that given a valid watermark, it is computationally difficult for an attacker to find the key K, such that $f(K) = W$. However, in some applications and our proof, we need the number of valid watermark to be small, so that it is computationally difficult for an attacker to find a valid watermark. On the other hand, if m is too small, an attacker can look for a suitable valid watermark using brute-force search.

In our construction, we choose $m = \sqrt{n}$, thus $|\mathcal{W}| = 2^{\sqrt{n}}$. As a result, it is computationally infeasible to do a brute-force search in the set of valid watermarks. At the same time, consider a randomly watermarked work $\widetilde{\mathcal{I}_n}$

drawn from distribution $\mathbf{X_r}$, which is of length n. With the parameters as in (2.8), the probability that $\widetilde{\mathcal{I}_n}$ contains any valid watermark $W \in \mathcal{W}$ is very small. Let us denote this probability $V(n)$ as a function of n, that is,

$$V(n) = \Pr[\exists W \in \mathcal{W}, \quad \widetilde{\mathcal{I}_n} \cdot W > T] \tag{2.12}$$

where $\widetilde{\mathcal{I}_n}$ is drawn from $\mathbf{X_r}$. Recall from Section 2.3.5 that the probability F_i that a randomly watermarked work can be declared as watermarked by a given valid watermark $W_i \in \mathcal{W}$ is exponentially small with respect to n. In particular, $F_i \leq \exp(-C_1 n)$ for some positive constant C_1 and for all $1 \leq i \leq |\mathcal{W}|$. Therefore,

$$\begin{aligned} V(n) &= 1 - \prod_{i=1}^{|\mathcal{W}|}(1 - F_i) \leq 1 - (1 - \exp(-C_1 n))^{2^m} \\ &< 2^m \exp(-C_1 n) < \exp(-C_1 n + \sqrt{n}) \end{aligned} \tag{2.13}$$

where C_1 is some positive constant. Note that $V(n)$ is a negligible function of n.

2.3.6 Proof of Security

Now, we are ready to prove that the proposed protocol is secure. We assume that the function f is a CSPRNG. Suppose that there is a successful attacker B as defined in Definition 1, we want to extend it to a statistical test $\mathcal{T}$ that has an advantage in distinguishing sequences produced by f from that by a truly random source. Since f is a CSPRNG, this leads to a contradiction, and thus such a B is impossible.

Given an input $W \in \{-1, 1\}^n$, the following steps are carried out by $\mathcal{T}$:

1. Randomly pick an original work I.

2. Compute $\widetilde{I} = I + kW$. That is, embed W into I.

3. Pass $\widetilde{I}$ to B and obtain an output.

4. If the output of B is a pair $(\widehat{W}, \widehat{K})$, such that $\widehat{W} = f(\widehat{K})$, then $\mathcal{T}$ declares that W is generated by f by outputting a 0. Otherwise B outputs a $\perp$, then $\mathcal{T}$ declares that W comes from a random source by outputting a 1.

We want to calculate the expected output of $\mathcal{T}$ for the following 2 cases. If the difference of the expected outputs of these 2 cases is non-negligible, then by the definitions in Section 2.3.4, f is not a CSPRNG, thus leads to a contradiction.

Case 1: W is from a random source. Suppose W is from a random source, then the probability that there exists a valid watermark $\widehat{W} \in \mathcal{W}$ in $\widetilde{I}$ is exactly the probability $V(n)$ in (2.13), which is negligible with respect to n as we have shown in Section 2.3.5. Hence, we know that $\mathcal{T}$ will almost always output a 1 to correctly declare that it is from the random source, except in the unlikely event $\mathcal{E}$ where $\widetilde{I}$ happens to contain a valid watermark. Clearly $\mathcal{E}$ happens with negligible probability $V(n)$. We observe that, when $\mathcal{E}$ happens, $\mathcal{T}$ may output a 0 with a probability that is not negligible (since B is a successful attacker). We consider the obvious worst case (best case for the attacker) that, $\mathcal{T}$ always output 0 when $\mathcal{E}$ happens. In this case, the fraction of 0's output by $\mathcal{T}$ is $V(n)$, which is still negligible. Therefore, let $E_1(\mathcal{T})$ be the expected output of $\mathcal{T}$, we have

$$E_1(\mathcal{T}) > 1 - V(n). \tag{2.14}$$

Case 2: W is from the CSPRNG f. Suppose W is generated by f, then W is a valid watermark. Since B is a successful attacker, by definition B is able to find a valid watermark $\widehat{W}$ that is already embedded in $\widetilde{I}$ with a probability that is not negligible. More specifically, for any positive integer n_0,

$$\Pr[B(\widetilde{I}) = \texttt{PASS}] > 1/p(n)$$

for some positive polynomial $p(\cdot)$ and for some $n > n_0$. Hence, the probability that $\mathcal{T}$ decides that W is from the CSPRNG f is more than $1/p(n)$. Hence, let $E_2(\mathcal{T})$ be the expected output of $\mathcal{T}$ in this case, and we have

$$E_2(\mathcal{T}) < \left(1 - \frac{1}{p(n)}\right). \tag{2.15}$$

Consider the difference between (2.14) and (2.15). Since $V(n)$ is negligible but $1/p(n)$ is not, the difference cannot be negligible because the sum of two negligible functions is still negligible. Hence, the difference between $E_1(\mathcal{T})$ and $E_2(\mathcal{T})$ is not negligible. Thus $\mathcal{T}$ has an advantage in distinguishing the truly random source from the the output of f, therefore f by definition is not a CSPRNG, which is a contradiction. As a result, such a successful attacker B does not exist.

2.3.7 Remarks

Choice of m. In our construction we require the parameter m to be small. However, it seems that even if it is large, say $m = n/2$, the protocol is still secure. Thus it would be interesting to find an alternative proof that handles large m.

Underlying watermarking scheme. For simplicity in the proof, we use a simple watermarking scheme, and "discretized" watermark $W \in \{-1,1\}^n$. The draw back is that the performance of false alarm, robustness and distortion would be far from optimal. Recent results in communication theory offer schemes that can achieve much higher performance. Thus, we can have much lower false alarm, with other requirement fixed. On the other hand, it is also not clear whether we can make these schemes secure against inversion attacks. This is because in these schemes, the watermark is usually derived from the original in an insecure manner. It is interesting to investigate this issue. Furthermore, our proof requires valid watermarks to be "sparsely populated" in $\{-1,1\}^n$. On the other hand, schemes with high performance usually require the watermarks to be densely populated, so as to reduce the distortion. Therefore, it is interesting to know if our proof can be extended.

Proving ownership. As mentioned earlier, to prove the ownership of a work $\widetilde{I}$, Alice has to show that she knows a pair $(K_{\mathtt{A}}, W_{\mathtt{A}})$, such that $W_{\mathtt{A}}$ is correctly generated from $K_{\mathtt{A}}$ and is detectable in $\widetilde{I}$. However, directly revealing such a pair in the proof might leak out information that leads to successful attacks. One alternative is to use zero-knowledge interactive proofs to prove the relationship between $K_{\mathtt{A}}$ and $W_{\mathtt{A}}$ without revealing the actual values. We note that it is straight forward to apply known zero-knowledge interactive proofs efficiently in our scheme. This is an advantage of our construction over schemes that involves hash functions (such as [20]), which are difficult to prove using known zero-knowledge interactive proofs.

Generation of watermarks. In Craver et al. [20], Alice computes a secure hash of the original I, and uses the hash value $h(I)$ to generate the watermark $W_{\mathtt{A}}$, which is then embedded into I. It is commonly believed that we need to generate the watermark from the original in a one-way manner to achieve non-invertibility since the attacker would be forced to break the underlying one-way function.

Interestingly, our scheme does not use the original I to derive the key $K_{\mathtt{A}}$, nor the watermark $W_{\mathtt{A}}$. Hence the watermark is statistically independent from the original. Although we can view the hash value $h(I)$ as the secret key $K_{\mathtt{A}}$ in our setting, our results show that it is not necessary to enforce a relationship between the watermark and the original work.

Chapter 3

Secure Authentication Using Helper Information

3.1 Secure Biometric Authentication

Many biometric data are noisy in the sense that small noises are introduced during acquisition and processing. Hence, two biometric samples that are different but close to each other, are considered to belong to the same identity. This poses technical challenges in applying classical cryptographic operations on them. Recently, new generic techniques such as fuzzy commitment [34], helper data [45] and secure sketch [23] are introduced to handle noisy data. These techniques attempt to remove the noise with the aid of some additional public data P. Given original data X, P is constructed and made public. During reconstruction, from Y, which is close to X, together with P, the original X can be determined. Here we call such data P the *helper information*.

An important requirement for helper information is that the amount of information about X revealed by publishing P should be limited. Dodis et al. [23] propose a notion of *entropy loss* to measure the security of the scheme, which provides a convenient way to bound the entropy loss for **any** distribution of X (Section 3.1.2 gives the details). Such worst case analysis is important in practice because typically, the actual distribution of the biometric data is not known. It is also desirable, although of a secondary concern, to avoid using randomness during the construction of P, and to make it short. Helper information that can be constructed deterministically can achieve some limited form of *reusability* [9].

Not surprisingly, the design of helper information is very much dependent on the definition of closeness. Helper information for the following two main types of data have been proposed: (1) The data are from a vector space, and two sequences are close to each other if their distance (e.g., Hamming distance) is less than a threshold. (2) The data X and Y are subsets of a universe $\mathcal{U}$, where $|X| = |Y| = s$, and they are close if the set difference $s - |X \cap Y| \leq t$, for some threshold t.

We observe that in many applications, a combination of the above is required. For example, a fingerprint template is typically represented as a set of minutiae points in a discrete 2-dimensional space, or even 3-dimensional if the less reliable orientation attribute is included [28]. Under noise, each points may be slightly perturbed, and a small number of points may be replaced.

We study helper information for such *point-sets*. A point-set X is a set of s points from a discrete d-dimensional domain $[0, N-1]^d$. Under permissible *white noise*, for each point $\langle x_1, .., x_d \rangle \in X$, each x_i, $1 \leq i \leq d$, may be perturbed by at most δ. In addition, at most t points in X may be replaced by the *replacement noise*. In other words, two sets X and Y are close to each other, if we can find a subset $X' \subset X$, $|X'| \geq s - t$, such that for each $x \in X'$, there is a unique $y \in Y$ that satisfies $\|x - y\|_\infty \leq \delta$, where $\| \cdot \|_\infty$ is the infinity norm. We assume that a point-set X is always *well-separated*, that is, for any $x, x' \in X$, the distant $\|x - x'\|_\infty \geq 3\delta$. This assumption is reasonable in practice. For example, in a fingerprint template, two minutiae points cannot be too close to each other, otherwise they will be considered as false minutiae and should be corrected [36].

Clancy et al. [14] give the following construction of helper information for a point-set that consists of two parts. Given the original point-set X, a set R of random points are generated in a way that no two points in $(X \cup R)$ are close to each other. We call the set $\mathcal{C} = (X \cup R)$ the *codebook*, and each point in $\mathcal{C}$ a *codeword*. The codewords are indexed according to some pre-defined order, and the description of the codebook forms the first part of the helper information. Next, using known techniques for set difference, helper information for the indices of the points in X (with respect to the ordered codebook) are constructed. This serves as the second part of the helper information. During reconstruction, for each point $y \in Y$, it is matched with the closest point $x \in \mathcal{C}$. If Y is close to X, at least $s - t$ points in X will be matched. By using the helper information for set difference, the replaced points (at most t of them) can be recovered.

Clancy et al. propose generating R iteratively. A point that is not close to the already generated points and X is uniformly chosen. The process is repeated until it is impossible to add more points. Such process is essentially the online parking problem which has intrinsic statistical properties [32]. However, it is not easy to analyze the above stochastic process and bound the entropy loss. For instance, it is not clear whether it would be better use a fixed number of iterations and hence obtain a fixed number of random points, or as suggested, generate maximum possible number of points. In addition, it is not easy to avoid using randomness during construction.

Here, we follow the approach of dividing the helper information into two parts. We focus on the first part (for white noise), and investigate different ways of generating the random R. For the second part, we employ known techniques for set different. Let $\mathcal{L}_{SD}(s,t,n)$ be the entropy loss of the helper information for set difference, where $n = |\mathcal{C}|$ is the size of codebook. There are schemes such that $\mathcal{L}_{SD}(s,t,n)$ is in $O(t \log n)$ (e.g., those in [23]). When $t = 0$, the total entropy loss is the same as that of the helper information for the white noise.

We first give a generic method of generating R, and an upper bound $\mathcal{L}_H$ for the entropy loss of $\mathcal{C} = (X \cup R)$. We show that $\mathcal{L}_H < s(d \log(4\delta + 1) + \log e)$, where e is the base of natural logarithm and $\log e \approx 1.443$. The overall entropy loss is at most $\mathcal{L}_H + \mathcal{L}_{SD}(s,t,N^d/(4\delta+1)^d)$. The bound is quite tight in the sense that there is a distribution of X such that the entropy loss of $\mathcal{C}$ is at least $\mathcal{L}_H - \epsilon$ where ϵ is a positive constant that is at most 3. By pre-rounding and carefully partitioning the domain $[0, N-1]$ into cells, we can improve the entropy loss in $d = 1$ to at most $s(2 + \log \delta) + \mathcal{L}_{SD}(s,t,N/(3\delta))$. We further apply the technique of partitioning to some special cases in two dimensions ($d = 2$) and obtain some improvements. Such techniques probably can be extended to $d = 2$ in general, and to higher dimensions. In addition, we give a method to reduce the size of the helper information, and avoid using randomness during the construction. We also give another method in one dimension to demonstrate that, using the size of R as the security measure would be misleading.

3.1.1 Related Works

Recently, a few new cryptographic primitives for noisy data are proposed. Fuzzy commitment scheme [34] is one of the earliest formal approaches to error tolerance. The fuzzy commitment scheme uses an error correcting code to handle Hamming distance. The notions of *secure sketch* and *fuzzy extractor* are introduced in [23], which gives constructions for Hamming distance, set difference, and edit distance. Under their framework, a reliable key is extracted from noisy data by reconstructing the original data with a given sketch (i.e., the helper information), and then applying a normal extractor (such as pair-wise independent hash functions) on the data. The issue of *reusability* of the helper information is addressed in [9]. It is shown that a helper information scheme that is provably secure may be insecure when multiple constructions of the helper information of the same biometric data are obtained.

The set difference metric is first considered in [33], which gives a *fuzzy vault* scheme. Later, [23] proposed three constructions. The entropy loss by all these schemes are roughly the same. They differ in the sizes of the helper information, decoding efficiency and also the degree of ease in practical implementation. The BCH-based scheme [23] has small helper information and achieves "sublinear" (with respect to the size of the universe) decoding by careful reworking of the standard BCH decoding algorithm. All these schemes cannot handle multi-sets. The set reconciliation protocol presented in [37] is designed for two parties to jointly discover the union of their data, with as little communication cost as possible. Although the problem settings are different, the techniques in handling set difference is similar and can be employed to obtain helper information.

A *fuzzy fingerprint vault* scheme is proposed in [14], where a fuzzy vault scheme is used in secure fingerprint verification using a smart card, and brute force attackers are considered.

Another line of research yields the constructions of *approximate message authentication codes* ([29, 6, 47, 22]), which can authenticate images that are corrupted by certain levels of noises, which are common to images (such as white noise and compression). There are also attempts in refining the extraction of biometric features so that the features are invariant to permissible noises [50]. Unfortunately, the reliability of such systems is not high.

3.1.2 Preliminaries

Entropy and entropy loss. We follow the definitions of entropy in [23]. They propose to examine the *average min-entropy* of X given P, which gives the minimum length of an almost uniform secret key that can be extracted even if P is made public.

Let $\mathbf{H}_\infty(A)$ be the min-entropy of random variable A, i.e., $\mathbf{H}_\infty(A) = -\log(\max_a \Pr[A = a])$. For two random variables A and B, the average min-entropy of A given B is defined as $\widetilde{\mathbf{H}}_\infty(A \mid B) = -\log(\mathbb{E}_{b\leftarrow B}[2^{-\mathbf{H}_\infty(A|B=b)}])$.

The entropy loss of X given P is defined as $\mathcal{L} = \mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X|P)$. When X is clear from the context, we just call it the entropy loss of P. This definition is useful in the analysis of entropy loss, since for any ℓ-bit string B, we have $\widetilde{\mathbf{H}}_\infty(A \mid B) \geq \mathbf{H}_\infty(A) - \ell$. For any helper information scheme, let R be the randomness invested in constructing the helper information, it can be shown that when R can be recovered from X and P,

$$\mathcal{L} = \mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X \mid P) \leq |P| - \mathbf{H}_\infty(R). \tag{3.1}$$

This gives a general method to find an upper bound of $\mathcal{L}$ that is independent of X, hence it applies to any distribution of X.

Helper information. Let $\mathcal{M}$ be a set with a *closeness* relation $\mathtt{C} \subseteq \mathcal{M} \times \mathcal{M}$. When $(X, Y) \in \mathtt{C}$, we say the Y is close to X, or (X, Y) is a close pair. Similar to the definitions in [23], we have

Definition 7 *A helper information scheme is a tuple* $(\mathcal{M}, \mathtt{C}, \mathsf{Enc}, \mathsf{Dec})$, *where* $\mathsf{Enc} : \mathcal{M} \to \{0,1\}^*$ *is an encoder and* $\mathsf{Dec} : \mathcal{M} \times \{0,1\}^* \to \mathcal{M}$ *is a decoder such that for all* $X, Y \in \mathcal{M}$, $\mathsf{Dec}(Y, \mathsf{Enc}(X)) = X$ *if* $(X, Y) \in \mathtt{C}$. *The string* $P = \mathsf{Enc}(X)$ *is to be made public and we call it the helper information. We say that a helper information scheme is* $\mathcal{L}$*-secure if for all random variable* X *over* $\mathcal{M}$, *the entropy loss of* P *is at most* $\mathcal{L}$. *That is,* $\mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X \mid \mathsf{Enc}(X)) \leq \mathcal{L}$.

Closeness relations. For any two points x and y, we define the closeness $\mathtt{C}_\delta$, where $(x, y) \in \mathtt{C}_\delta$ if $\|x - y\|_\infty \leq \delta$. We further define the closeness $\mathtt{PS}_{\delta,s,t}$ for two point-sets.

Definition 8 *For any* $X = \{x_1, \ldots, x_s\}$ *and* $Y = \{y_1, \ldots, y_s\}$, *which are subsets of some universe* $\mathcal{U}$, *we say that* $(X, Y) \in \mathtt{PS}_{\delta,s,t}$ *if there exists a 1-1 correspondence* f *on* $\{1, \ldots, s\}$ *such that* $|\{i \mid (x_{f(i)}, y_i) \in \mathtt{C}_\delta\}| \geq s - t$.

In other words, two point-sets are close if the maximum number of matched points (in infinity norm sense) is greater than some threshold.

A Lower Bound of The Entropy Loss

Here we give a lower bound $\mathcal{L}_0$ of the entropy loss. That is, for any helper information scheme $(\mathcal{P}([0, N-1]^d), \mathtt{PS}_{\delta,s,t}, \mathsf{Enc}, \mathsf{Dec})$, there exists a distribution of X such that the entropy loss of $P = \mathsf{Enc}(X)$ is at least $\mathcal{L}_0$.

For any distribution of X, let $\mathcal{X}_b$ to be the set of all possible original given $P = b$. We observe that

$$\max_a \Pr[X = a \mid P = b] \geq \frac{1}{|\mathcal{X}_b|}. \tag{3.2}$$

Substitute it into the definition, we have

$$\widetilde{\mathbf{H}}_\infty(X|P) \leq \max_{b, \Pr[P=b]\neq 0} \log |\mathcal{X}_b|. \tag{3.3}$$

Now, note that during reconstruction we require that $\mathsf{Dec}(\mathsf{Enc}(X), Y) = X$ for all $(X, Y) \in \mathtt{PS}_{\delta,s,t}$, by counting, the right hand side is bounded by

$$\log m - \left(ds \log \delta + \log \begin{pmatrix} s \\ t \end{pmatrix} + \log \begin{pmatrix} (\lfloor N/2\delta \rfloor)^d - 2^d s \\ t \end{pmatrix} \right) \tag{3.4}$$

in d-dimensional spaces, where m is the total number of well-separated sets that are of size s in $\mathcal{M}$.

Now, considering X that is uniformly distributed over all well-separated sets of size s in $[0, N-1]^d$, the min-entropy of X is $\mathbf{H}_\infty(X) = \log m$, and the entropy loss can be bounded by

$$\mathcal{L}_0 = \mathbf{H}_\infty(X) - \widetilde{\mathbf{H}}_\infty(X|P) \geq ds \log \delta + \log \begin{pmatrix} s \\ t \end{pmatrix} + \log \begin{pmatrix} (\lfloor N/2\delta \rfloor)^d - 2^d s \\ t \end{pmatrix} \tag{3.5}$$

in d-dimensional space.

Furthermore, we can see that $\mathcal{L}_0$ is in $ds \log(2\delta+1) + \Omega(d \log(N/(4\delta+2)))$, when t is a positive constant, and $s < N/(4\delta + 2)$. The second term is due to the replacement noise, and the first term is due to the white noise, which we are more interested in.

3.1.3 The Basic Construction

Recall that the helper information consists of two parts $P_H P_S$, where P_H is the helper information that removes the white noise. During encoding, a large number of points R is generated to form the codebook $\mathcal{C} = (X \cup R)$, and P_H is its description. During decoding, the points in Y are matched with the nearest codewords in $\mathcal{C}$, so that white noise can be removed. The helper P_S for set difference is constructed using known schemes on $\mathcal{C}$ to correct the replacement noise. We also assume that X is well-separated.

Here we focus on the construction of P_H. We will first give our basic construction in one dimension ($d = 1$), and then show that it can be extended to higher dimensions.

Construction of P_H in One Dimension ($d = 1$)

For any point $x \in [0, N-1]$, call the set $S_1(x) = \{x+1, x+2, \ldots, x+2\delta\}$ the *half-sphere* of x.

Given $X = \{x_1, \ldots, x_s\}$, we construct as below the helper information P_H to be a sequence $\langle h_0, h_1, \ldots, h_{N-1} \rangle$, where each $h_i \in [0, p_1 - 1]$, and p_1 is a parameter that is chosen to be $2\delta + 1$ for optimal performance.

1. For each $x \in X$, set $h_x = 0$, and for each $a \in S_1(x)$, h_a is uniformly chosen at random from $\{1, \ldots, p_1 - 1\}$.

2. For each h_i that has not been set in step 1, uniformly choose its value from $\{0, \ldots, p_1 - 1\}$.

From P_H, the codebook $\mathcal{C} = (X \cup R)$ can be determined. For each point $x \in [0, N-1]$, $x \in \mathcal{C}$ if and only if $h_x = 0$ and $h_a \neq 0$ for all $a \in S_1(x)$. We then construct the second part P_S on $\mathcal{C}$.

During decoding, given Y, each point $y \in Y$ is matched with its nearest point in $\mathcal{C}$. Suppose y is a noisy version of an $x \in X$, i.e. $|y - x| \le \delta$, it is easy to to verify that x is its closest point in $\mathcal{C}$. Hence, P_H can correct the white noise. Lemma 9 gives the entropy loss, and Lemma 10 shows that the bound is quite tight.

LEMMA 9 *The entropy loss of X given P_H is at most*

$$s\left(\log(2\delta+1) + 2\delta\log(1+\frac{1}{2\delta})\right)$$

which is no greater than $s\left(\log(2\delta+1) + \log e\right)$*, where e is the base of natural logarithm.*

Proof: Since the randomness invested in constructing P_H can be recovered from X and P_H, we can apply (3.1) as in Section 3.1.2. In particular, we look at the difference between the size of P_H, which is $N \log p_1$, and the randomness invested in constructing P_H. For any h_i in P_H, if it is not set in Step 1 of the above construction, $|h_i| = \log p_1$, which equals the invested randomness, and hence it does not contribute to the difference. For each h_x such that $x \in X$, it is set to 0, which contributes $\log p_1$ to the difference. For each h_a such that $a \in S_1(x)$ for some $x \in X$, we use $\log(p_1 - 1)$ bits of randomness, hence the difference introduced is $\log \frac{p_1}{p_1 - 1}$.

Therefore, the total difference (hence the entropy loss) is at most

$$s\left(\log p_1 + 2\delta \log \frac{p_1}{p_1 - 1}\right).$$

When $p_1 = 2\delta + 1$, we have

$$\mathcal{L}_H \le s\left(\log(2\delta+1) + 2\delta\log(1+\frac{1}{2\delta})\right).$$

Since $(1+\frac{1}{2\delta})^{2\delta}$ approaches e from below when δ approaches infinity, we have the above claimed bound. $\square$

LEMMA 10 *There exists a distribution of X, where the entropy loss of X given P_H is at least $s(\log(2\delta+1)+2\delta\log(1+\frac{1}{2\delta}))-\epsilon$ for some positive constant ϵ.*

Proof: Let $\Delta = 2\delta+1$. Consider the distribution $X=\{x_1, x_1+2\Delta, \cdots, x_1+2(s-1)\Delta\}$, where x_1 is uniformly chosen from a set $A=\{a_1,\cdots,a_\lambda\}$ of λ points. Hence, $\mathbf{H}_\infty(X)=\log\lambda$. Given P_H, a point $a_i\in A$ is a candidate for x_1 if and only if $h_{a_i}=0$ and $h_b\neq 0$ for all $b\in S_1(a_i)$. For any $a_i\neq x_1$, the probability that a_i is a candidate for x_1 is at most $\frac{1}{\Delta^s}(1-\frac{1}{\Delta})^{(\Delta-1)s}$. Let C be the number of candidates of x_1 for a given P_H, then $\mathbb{E}[C]\leq 1+\frac{\lambda-1}{\Delta^s}(1-\frac{1}{\Delta})^{(\Delta-1)s}\leq 1+\frac{\lambda}{\Delta^s}(1-\frac{1}{\Delta})^{(\Delta-1)s}$. Now choose $\lambda=2^{s(\log\Delta+(\Delta-1)\log(1+\frac{1}{\Delta}))}$, we have $\mathbb{E}[C]\leq 2$. By Markov's Inequality, we have $\Pr[C\leq 4]\geq 1-\mathbb{E}[C]/4\geq 1/2$. We note that

$$\begin{aligned}&\mathbb{E}_{b\leftarrow P_H}\left[2^{-\mathbf{H}_\infty(X|P_H=b)}\right]\\ =&\mathbb{E}_{b\leftarrow P_H}\left[\max_a \Pr[X=a|P_H=b]\right]\\ \geq&\frac{1}{4}\Pr[C\leq 4]\geq\frac{1}{8}.\end{aligned}\tag{3.6}$$

Therefore, the left-over entropy $\widetilde{\mathbf{H}}_\infty(X|P)\leq -\log\frac{1}{8}=3$. Considering that $\mathbf{H}_\infty(X)=\log\lambda=s\left(\log(2\delta+1)+2\delta\log(1+\frac{1}{2\delta})\right)$, and let $\epsilon=3$, we have the claimed bound. $\square$

Extension to Higher Dimensions

Let us first define a total order for the points in $[0, N-1]^d$. We define a total order $\langle x_1, x_2, \ldots, x_d \rangle \succ \langle x'_1, x'_2, \ldots, x'_d \rangle$ if and only if there exists an i such that $x_i > x'_i$ and $x_j = x'_j$ for all $1 \leq j < i$. We define the half-sphere $S_d(x) = \{y \mid 0 < \|y - x\|_\infty \leq 2\delta \text{ and } y \succ x\}$.

The helper information P_H is a set of N^d symbols. For each $h_y \in P_H$, we have $y \in [0, N-1]^d$ and $h_y \in \{0, \ldots, p_d - 1\}$ for some parameter p_d that is to be chosen later. We construct P_H as below.

1. For each $x \in X$, set $h_x = 0$. For every $a \in S_d(x)$, uniformly choose h_a at random from $\{1, \ldots, p_d - 1\}$.

2. For each h_y that is not set in step 1, choose its value uniformly at random from $\{0, \ldots, p_d - 1\}$.

From P_H we can determine the codebook $\mathcal{C}$ as follows. A point $x \in [0, N-1]^d$ is in $\mathcal{C}$ if and only if $h_x = 0$ and for every $a \in S_d(x)$, we have $h_a \neq 0$. We can then construct the second part P_S of the helper information for set difference. Suppose y is a noisy version of an $x \in X$, that is, $\|y - x\|_\infty \leq \delta$, it is not difficult to verify that its closest point in $\mathcal{C}$ is x.

In fact, this construction is essentially the same as the construction for $d = 1$, except that the size of $S_d(x)$ is larger when $d > 1$. By simple counting we have

$$|S_d(x)| = \frac{(4\delta + 1)^d - 1}{2}. \tag{3.7}$$

Similar to the one-dimensional case, we choose $p_d = |S_d(x)| + 1$, and we have the

THEOREM 11 *The entropy loss of X given P_H is at most*

$$s\left(\log p_d + (p_d - 1)\log(1 + \frac{1}{p_d - 1})\right) \leq s\left(d\log(4\delta+1) + \log e\right)$$

in d-dimensions, where $p_d = \frac{(4\delta+1)^d+1}{2}$, and e is the base of natural logarithm.

Similarly, the above bound is tight. That is, there is a distribution of X such that the entropy loss is at least $s\left(\log p_d + (p_d - 1)\log(1 + \frac{1}{p_d-1})\right) - \epsilon$ for some positive constant ϵ. Taking into consideration the entropy loss of helper information for set difference, we have

COROLLARY 12 *The entropy loss of X given $P_H P_S$ is at most*

$$s\left(d\log(4\delta+1) + \log e\right) + \mathcal{L}_{SD}\left(s, t, \frac{N^d}{(2\delta+1)^d}\right)$$

in d-dimensions.

3.1.4 Improved Schemes

The generic construction in Section 3.1.3 can indeed be further improved in terms of entropy loss. We employ two techniques. The first is *pre-rounding.* That is, each point in X and Y is rounded prior to both encoding and decoding. We observe that, the effect of the white noise is reduced on the rounded points. The second technique is *partitioning*, where we carefully partition the domain into cells, and require that there is at most one codeword in each cell. Both techniques are useful in reducing the randomness required in constructing P_H.

Improvement in One Dimension ($d = 1$)

First, we give an improvement for $\delta = 1$ using partitioning, and we observe that this scheme can be extended to any $\delta > 1$ by pre-rounding.

We partition the domain $[0, N-1]$ into cells of size 3, such that each cell contains 3 consecutive points of the form $3k, 3k+1, 3k+2$. Now we construct P_H as a binary sequence of size $\lceil N/3 \rceil$, in which we assign one bit for each cell. Firstly, for each point $x \in X$, let $i = \lfloor x/3 \rfloor$, and $r = x \mod 3$. Considering the set $\{h_{i-1}, h_i, h_{i+1}\}$, we set the values for two of them according to Table 3.1(a), where a '-' indicates that the value is not set. For all h_i that is not set in the previous step, we uniformly choose it from $\{0, 1\}$ at random. For any i, we find the codeword in the cell $\{3i, 3i+1, 3i+2\}$ using the value of h_i and h_{i+1}, according to Table 3.1(b). This is also illustrated

	h_{i-1}	h_i	h_{i+1}
$r = 0$	0	0	-
$r = 1$	-	1	0
$r = 2$	-	1	1

(a)

	$h_{i+1} = 0$	$h_{i+1} = 1$
$h_i = 0$	$3i$	$3i$
$h_i = 1$	$3i+1$	$3i+2$

(b)

Table 3.1: Improved Scheme for $d = 1$.

in Figure 3.1.

During decoding, each point $y \in Y$ is rounded as the following. If there is a codeword $w \in \mathcal{C}$ such that it is within the same cell as y and $|w - y| \leq 1$, then y is rounded to w, otherwise it is rounded to the nearest codeword in $\mathcal{C}$.

Similar to the results we have in Section 3.1.3, it is not difficult to show that for each $x \in X$ we need specify at most 2 bits in the sequence $\langle h_0, \ldots, h_{N-1} \rangle$, which introduces 2 bits of entropy loss. Hence, the entropy loss due to P_H constructed here is at most $2s$. Therefore, we have the

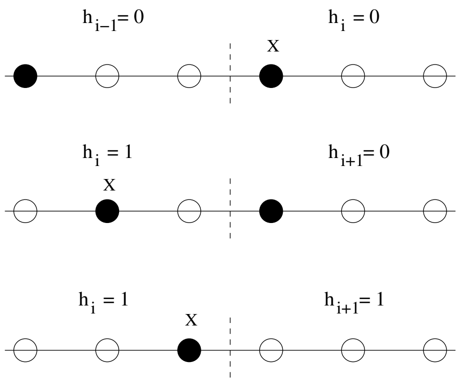

Figure 3.1: Improvement in one dimension.

Lemma 13 *The entropy loss for the above scheme when $\delta = 1$ is at most $2s + \mathcal{L}_{SD}(s,t,N/3)$.*

To extend this scheme to any δ, we employ pre-rounding. In particular, let $\widetilde{\mathcal{U}} = [0, \lceil N/\delta \rceil - 1]$, for each point $x \in [0, N-1]$, we round it to $k \in \widetilde{\mathcal{U}}$ if and only if k is the largest value such that $k\delta \leq x$. Note that x under noise can only be rounded to one of the values in $\{k-1, k, k+1\}$. Let $\widetilde{X} \subset \widetilde{\mathcal{U}}$ be the rounded points. Next, we apply the above scheme on $\widetilde{X}$ in universe $\widetilde{\mathcal{U}}$. Furthermore, the entropy loss due to the pre-rounding is at most $\log \delta$ for each point. Hence, we have the

Theorem 14 *The entropy loss for the above scheme is at most*

$$(2 + \log \delta)s + \mathcal{L}_{SD}\left(s, t, N/(3\delta)\right).$$

Improvement for $d = 2$ and $\delta = 1$

For $\delta = 1$ in two dimensions, with a parameter $\alpha \in [0,4]$, we partition the space such that every 5 points of the form $\{(x, 5k+\alpha), (x, 5k+\alpha+1), (x, 5k+\alpha+2), (x, 5k+\alpha+3), (x, 5k+\alpha+4)\}$ for some non-negative integer k, are grouped into a cell (Figure 3.2). Each cell is assigned a number $q \in [0, p_2 - 1]$ for some parameter $p_2 \geq 6$. If $q \leq 4$, then $(x, 5k+q)$ is a codeword, otherwise there is no codeword in this cell.

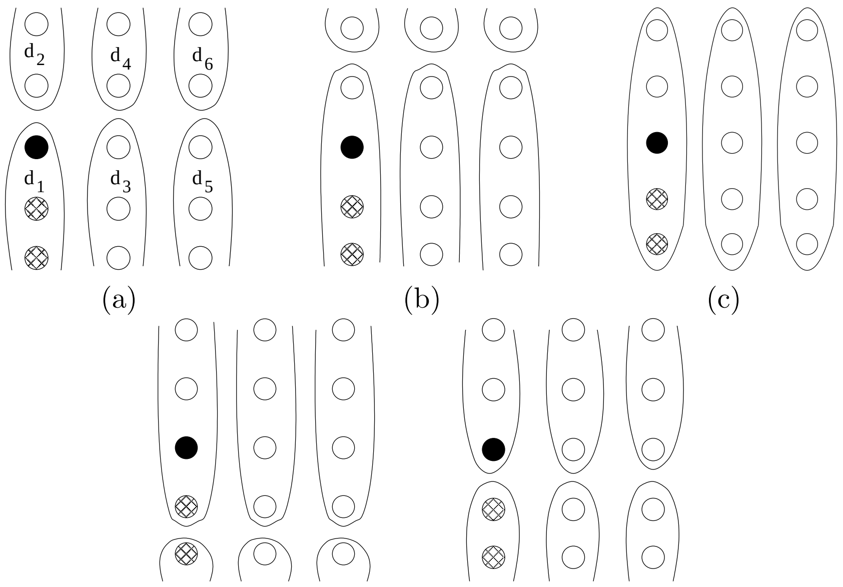

Figure 3.2: Cells of size 5. The black point is a data point.

All five possible scenarios for a data point x are illustrated in Figure 3.2. In each of these scenarios, the black point is a data point in X, and by our construction we require that all the white points are not codewords, and for the shadowed points, we do not care.

Now we count the entropy loss for the scenario in Figure 3.2(a). By the same principle as in Section 3.1.3, all the white points in the figure cannot be a codeword. Hence, for cell labelled d_1, there is only 1 choice for the value of the corresponding q, for d_3 and d_5, there are $p_2 - 3$ choices, and for d_2, d_4, and d_6, there are $p_2 - 2$ choices. Hence the entropy loss for this point is $\log p_2 + 2\log(p_2/(p_2 - 3)) + 3\log(p_2/(p_2 - 2))$.

Now we choose $p_2 = 14$, and the entropy loss for all five scenarios are as shown in Table 3.1.4. Next, we choose a value for α, such that scenario (e)

(a)	$\log p_2 + 2\log(p_2/(p_2-3)) + 3\log(p_2/(p_2-2))$	< 5.1704
(b)	$\log p_2 + 2\log(p_2/(p_2-4)) + 3\log(p_2/(p_2-1))$	< 5.0990
(c)	$\log p_2 + 2\log(p_2/(p_2-5))$	< 5.0823
(d)	$\log p_2 + 2\log(p_2/(p_2-4)) + 2\log(p_2/(p_2-1))$	< 4.9921
(e)	$\log p_2 + 2\log(p_2/(p_2-3)) + 2\log(p_2/(p_2-2))$	< 4.9480

Table 3.2: Entropy loss in five scenarios.

happens most often. By this choice of α, it is not difficult to see that the worst case is that each of the scenarios (a), (b), (c) and (e) happens 1/4 of the times. In this case, we have

$$\mathcal{L}_H \leq 5.0750s \tag{3.8}$$

whereas in the basic construction in Section 3.1.3, the bound is at least $5.0861s$ for $\delta = 1$.

Although the improvement is small, this construction suggests that the basic construction can be further improved by partitioning, and this technique is likely to be extensible to higher dimensions.

A Scheme in Two Dimensions ($d = 2$) for `0-1` Noise

In 2-dimensions, we have a better scheme for a special type of white noise, under which each point in $(x_1, x_2) \in X$ is perturbed to one of the four possible points in $\{(x_1, x_2), (x_1 + 1, x_2), (x_1, x_2 + 1), (x_1 + 1, x_2 + 1)\}$. We call this noise the `0-1` noise.

Note that each point $x \in X$ is of one of the four types $(2k, 2j), (2k + 1, 2j), (2k, 2j + 1), (2k + 1, 2j + 1)$, for some k and j. Assume that the most common type of these points is $(2k+\alpha, 2j+\beta)$, where $\alpha, \beta \in \{0, 1\}$. Now we group together in a cell every four points of the form $(2k + \alpha, 2j + \beta), (2k + \alpha + 1, 2j + \beta), (2k + \alpha, 2j + \beta + 1), (2k + \alpha + 1, 2j + \beta + 1)$, for some k

and j (Figure 3.3). We call the point $(2k+\alpha, 2j+\beta)$ the *base point* of the cell. The values of α and β form part of P_H, and introduce at most 2 bits of entropy loss. In the following discussions, without loss of generality, we assume that $\alpha = \beta = 0$. We assume that each cell is given a unique index in $[0, N^2/4-1]$. For instance, for the cell containing point (x_1, x_2), we can define its index as $\texttt{ind}(x_1, x_2) = \lfloor \frac{x_2}{2} \rfloor \frac{N}{2\delta} + \lfloor \frac{x_1}{2} \rfloor$, assuming N is divisible by 2δ. Let $\texttt{ind}(x)$ be the index of the cell where x is in.

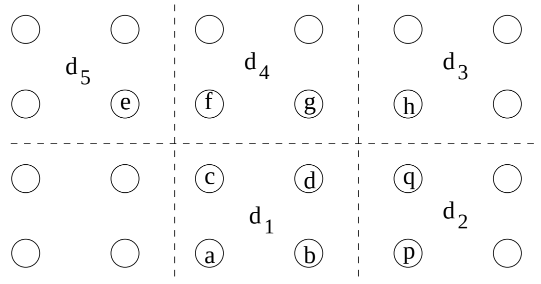

Figure 3.3: Cells in 2-D. A point $x \in X$ is in cell d_1.

We now construct a binary sequence P_H as the following. For each $x = (x_1, x_2) \in X$, the point $a = (2\lfloor x_1/2 \rfloor, 2\lfloor x_2/2 \rfloor)$ is the base point of the cell that x is in. Let $d_1 = \texttt{ind}(x)$, $d_2 = \texttt{ind}(x+(2,0))$, $d_3 = \texttt{ind}(x+(2,2))$, $d_4 = \texttt{ind}(x+(0,2))$, and $d_5 = \texttt{ind}(x+(-2,2))$, as illustrated in Figure 3.3.

- Set $h_{2d_1} = x_1 \mod 2$, and set $h_{2d_1+1} = x_2 \mod 2$.
- If $x = a$ is the base point. No other action is required.
- If $x = b$, set $h_{2d_2} = 1$, and $h_{2d_2+1} = r_1$.
- If $x = c$, set $h_{2d_4} = r_2$, $h_{2d_4+1} = 1$, $h_{2d_5} = r_3$, and $h_{2d_5+1} = r_4$.
- If $x = d$, set $h_{2d_2} = r_5$, $h_{2d_2+1} = r_6$, $h_{2d_4} = r_7$, $h_{2d_4+1} = 1$, $h_{2d_3} = r_8$, and $h_{2d_3+1} = r_9$.

Where r_1, r_2 and r_7 are uniformly chosen from $\{0,1\}$, r_3r_4, r_5r_6 and r_8r_9 are uniformly chosen from the three combinations $\{00,01,11\}$, $\{00,10,11\}$ and $\{10,01,11\}$ respectively.

During decoding, we can find the codeword in a cell with index d and base point a as $a+(h_{2d},h_{2d+1})$. Given a corrupted data point $y=(y_1,y_2)\in Y$, Consider the three points $w_1=(y_1-1,y_2)$, $w_2=(y_1,y_2-1)$, and $w_3=(y_1-1,y_2-1)$ in sequence. If one of them is within the same cell as y, then y is rounded to it. Otherwise y is rounded to the first codeword in those three. In case none of these points is a codeword, we round y to any codeword in $\mathcal{C}$.

Since x is most likely to be the base point a, it is not difficult to show that the entropy loss of P_H is at most $(8/3+\log(4/3))s+2\approx 3.08s+2$. Hence, we have the

THEOREM 15 *The entropy loss for the above scheme is at most* $(\frac{8}{3}+\log\frac{4}{3})s+2+\mathcal{L}_{SD}(s,t,\frac{N^2}{4})\approx 3.08s+2+\mathcal{L}_{SD}(s,t,\frac{N^2}{4})$.

Note that in the basic construction when $d=2$ and $\delta=1$, the entropy loss is $5.09s$.

3.1.5 Short Description of P_H

In the basic constructions (Section 3.1.3), we can view P_H as a random sequence of length $N^d\log p_d$ with two types of constraints: Type 0 constraint is of the form $(k,0)$, which requires that $h_k=0$, and type 1 constraint is of the form $(k,1)$ which requires that $h_k\neq 0$. The main idea is as follows: Find the seed of some pseudo-random generator, such that the generated sequence satisfies all the type 0 and 1 constraints, and use the seed as the helper information. In this section, we give two methods. The first method has efficient decoding and encoding algorithms, but still requires randomness. The second method eliminates all randomness but there is no known efficient encoder.

Using High Degree Polynomials

Let $n = N^d$, and assign each $x \in [0, N-1]^d$ a unique index $\mathtt{ind}(x)$ in $[0, n-1]$. Given a constraint set $S = \{(k_1, r_1), \ldots, (k_m, r_m)\}$, we construct a polynomial $f(x)$ of degree at most $m-1$ in $\mathbb{Z}_n^*$ as the following.

1. Uniformly choose $d_1, \ldots, d_m \in \mathbb{Z}_n^*$ at random such that for $1 \leq i \leq m$, if $r_i = 0$, then $d_i \equiv 0 \mod p_d$, otherwise $d_i \not\equiv 0 \mod p_d$.
2. Find the polynomial f of degree at most $m-1$ such that $f(\mathtt{ind}(k_i)) \equiv d_i \mod n$ for $1 \leq i \leq m$.

The m coefficients of f is published as the helper information. During decoding, each h_k in P_H can be recovered by computing $h_k = (f(\mathtt{ind}(k)) \mod n) \mod p_d$. Since for each point x we can have at most $|S_d(x)| + 1$ constraints, The polynomial f can be represented using $\frac{ds((4\delta+1)^d+1)}{2} \log N$ bits.

When p_d divides n, the entropy loss of this helper information is the same as the basic construction.

Using Almost k-Wise Independence

We observe that the helper information can be further shortened and made deterministic by employing an *almost k-wise independent* sample space [4].

A sample space on n-bit strings is k-wise independent if, the probability distribution, induced on every k bit locations in a randomly chosen string from the sample space, is uniform. Alon et al [4] considered almost k-wise independence with small sample size, and gave several constructions.

DEFINITION 16 (ALMOST k-WISE INDEPENDENCE [4]) *Let S_n be a sample space and $X = x_1 \cdots x_n$ be chosen uniformly from S_n. S_n is almost k-wise independent with ϵ statistical difference if, for any k positions $i_1 < i_2 < \ldots < i_k$, and any k-bit string α, we have*

$$\sum_{\alpha \in \{0,1\}^k} |\Pr[x_{i_1} x_{i_2} \ldots x_{i_k} = \alpha] - 2^{-k}| \leq \epsilon. \tag{3.9}$$

If we choose $\epsilon = 2^{-k}$, the probability $\Pr[x_{i_1} x_{i_2} \ldots x_{i_k} = \alpha]$ in (3.9) is non-zero. To see this, let $X' = x_{i_1} x_{i_2} \ldots x_{i_k}$, then $\Pr[X' = \alpha] = 0$, and there exists some $\beta \neq \alpha$ such that $\Pr[X' = \beta] > 2^{-k}$, since otherwise $\sum_{\alpha \in \{0,1\}^k} \Pr[X' = \alpha] < 1$. Thus $|\Pr[X' = \alpha] - 2^{-k}| + |\Pr[X' = \beta] - 2^{-k}| > 2^{-k}$, which is a contradiction. Hence, we can always find such X given any $i_1, \ldots, i_k$ and any α. Furthermore, the number of bits required to describe the sample is $(2 + o(1))(\log\log n + 3k/2 + \log k)$ which is in $O(k + \log\log n)$. Note that the construction of such a k-wise independent space does not require any randomness.

We observe that this construction can be employed to make the helper information shorter. For instance, for $d = 1$ and $\delta = 1$ in our basic construction, we can construct such a sample space with $k = 3s$ and $n = N$, and use the first sample that satisfies the constraints. The size of the helper information is in $o(s + \log\log N)$, which is also an upper bound for the entropy loss. It is possible to extend it to any d and δ, but we do not have a good upper bound of the entropy loss.

3.1.6 Entropy Loss of a Random Placement Method

Intuitively, it seems that it is better to have the codebook $\mathcal{C} = (X \cup R)$ as large as possible, since then a brute-force attacker will need to try the maximum number of times to guess X. In this section we give a seemingly natural random placement method to construct P_H with a large R in one dimension, and we show that the entropy loss is high for certain distributions of X.

1. Let $r_0 = -\delta$, $i = 1$, and $\Delta = 2\delta + 1$.
2. If there is an $x \in X$ s.t. $x - r_{i-1} \in [2\delta, 4\delta]$ then let $r_i = x$.
3. If there is an $x \in X$ s.t. $x - r_{i-1} \in [4\delta + 1, 6\delta]$, uniformly choose r_i from $[r_{i-1} + 2\delta, x - \Delta]$. Otherwise, uniformly choose r_i from $[r_{i-1} + 2\delta, r_{i-1} + 4\delta]$.
4. Increase i by 1, and repeat from Step 2 until $i = \lceil N/(2\delta + 1) \rceil + 1$.
5. Output $P_H = \{r_1, \ldots, r_{\lceil N/(2\delta+1) \rceil}\}$.

In other words, we randomly choose the smallest random point first, then the second smallest, and so on, until no further point can be added. The codebook is P_H, where r_i's are ignored if they are greater than $N - 1$. The extra padding is only for the ease of the proof.

Consider $X = \{x_1, x_1 + 2\Delta, \ldots, x_1 + 2(s-1)\Delta\}$, where x_1 is uniformly distributed. It can be shown that the entropy loss of X given P_H is at least $2s \log \Delta - \epsilon$ for some small positive constant ϵ. Comparing with other constructions in this section, this method reveals the most information, even though it produces the largest number of codewords.

3.1.7 Helper Information for Set Difference

It is observed that in previous works, the schemes for set difference cannot handle multi-sets, i.e., sets that may contain duplicated elements. Furthermore, it seems to be difficult to have a scheme with efficient encoder and decoder and small helper information.

Here we give a scheme that can handle the case where X is a multi-set. The size of the helper information is at most $2t(1 + \log N)$. In addition, there exists a simple and yet efficient decoding algorithm – we just need to solve a linear system with $2t$ equations and unknowns and find the roots of two degree t polynomials.

To handle a special case, we assume that X does not contain any element in $\{0, 1, \ldots, 2t-1\}$, and will discuss how to remove this assumption later at the end of this section. Our construction is similar to the set reconciliation protocol in [37], but the problem settings are different.

In the following algorithms, we assume that N is prime and all computations are done in $\mathbb{Z}_N^*$.

The encoder Enc_s. Given $X = \{x_1, \ldots, x_s\}$, the encoder does the following.

1. Construct a monic polynomial $p(x) = \prod_{i=1}^{s}(x - x_i)$ of degree s.
2. Publish $P = \langle p(0), p(1), \ldots, p(2t-1)\rangle$.

The decoder Dec_s. Given $P = \langle p(0), p(1), \ldots, p(2t-1)\rangle$ and $Y = \{y_1, \ldots, y_s\}$, the decoder follows the steps below.

1. Construct a polynomial $q(x) = \prod_{i=1}^{s}(x - y_i)$ of degree s.
2. Compute $q(0), q(1), \ldots, q(2t-1)$.

3. Let $p'(x) = x^t + \sum_{j=0}^{t-1} a_j x^j$ and $q'(x) = x^t + \sum_{j=0}^{t-1} b_j x^j$ be monic polynomials of degree t. Construct the following system of linear equations with the a_j's and b_j's as unknowns.

$$q(i)p'(i) = p(i)q'(i), \quad \text{for } 0 \le i \le 2t-1 \tag{3.10}$$

4. Find one solution for the above linear system. Since there are $2t$ equations and $2t$ unknowns, such a solution always exists.

5. Solve for the roots of the polynomials $p'(x)$ and $q'(x)$. Let them be X' and Y' respectively.

6. Output $\widetilde{X} = (Y \cup X') \setminus Y'$.

The correctness of this scheme is straight forward. When there is exactly t replacement errors, we can view $p'(x)$ as the "missed" polynomial whose roots are in $X' = X \setminus Y$. Similarly, $q'(x)$ is the "wrong" polynomial, whose roots are in $Y' = Y \setminus X$. Since the roots of $p(x)$ and $q(x)$ are in X and Y respectively, we have $q(x)p'(x) = p(x)q'(x)$. This interpretation motivates the equation (3.10).

When there are less than t replacement errors, there will be many degree t monic polynomials $p'(x)$ and $q'(x)$ that satisfy $q(x)p'(x) = p(x)q'(x)$. For any such $p'(x)$ and $q'(x)$, they share some common roots, which could be some arbitrary multi-set Z. That is, $X' = (X \setminus Y) \cup Z$, and $Y' = (Y \setminus X) \cup Z$. In Step 6, this extra Z will be eliminated.

When $X \cap \{0, \ldots, 2t-1\} \neq \emptyset$, some equations in (3.10) would degenerate, which makes the rank of the linear system less than $2t$. In this case, it is not clear how to find the correct polynomial in the solution space. Hence we require that $X \cap \{0, \ldots, 2t-1\} = \emptyset$.

Note that in the above we do not require the elements of X and Y to be distinct, so this scheme can handle multi-sets. Furthermore, since the size of each $p(i)$ for $1 \leq i \leq 2t$ is $(\log N)$, the size of P is $2t(\log N)$. Therefore, we have the

Lemma 17 *When $X \subset \mathbb{Z}_N^*$ and $X \cap \{0, \ldots, 2t-1\} = \emptyset$, the entropy loss due to $\mathsf{Enc}_s(X)$ is at most $2t \log N$.*

Removing the assumption on X and Y. The assumption that X cannot contain any element from $\{0, \ldots, 2t-1\}$ can be easily relaxed. We can find the smallest prime M such that $M - N \geq 2t$, and then apply the scheme on $\mathbb{Z}_M^*$. But instead of publishing $p(0), \ldots, p(2t-1)$, we publish $p(M-1), \ldots, p(M-2t)$. In this way, the size of the helper information is $2t \log M$. In practice, this is not a problem since the size of the universe may not be prime, and we will need to choose a larger finite field anyway. For t that is not too large (say, $t \leq N/4$), we can always find at least one prime in $[n + 2t, 2n]$, using the bounds in [43]. Hence, we have the

Lemma 18 *When $t \leq N/4$, the entropy loss due to $\mathsf{Enc}_s(X)$ is at most $2t(1 + \log N)$.*

3.1.8 Future Work

Having obtained some preliminary results in handling point-sets, it is interesting to investigate further the following issues in secure biometric authentication.

Better understanding of the problem: After having a helper information scheme for point-set difference, it would be meaningful to go deeper and gain better understanding of the problem. Currently our solution is to put a large number of “fake” data points to confuse the attacker, while allowing a legitimate user to recover the original data points with some data that are close to the original. However, the following questions remain: Is this only the abstract view of our particular solution, or does it represent the fundamental characteristics of the problem? Does this interpretation leads to the optimal solution to the original problem? Under our setting, there is still a gap between the lower bound that we can prove, and the entropy loss we actually achieve. It would be interesting to investigate how to fill in this gap.

Towards a working system: Our initial motivation is secure fingerprint authentication by making use of helper information. Beside the results we have already obtained, there are further problems to be solved before we can achieve this goal.

For example, our preliminary scheme assumes that the data points are well separated in space. It is not clear what the best way is to handle points that are close to each other. Also, a minutiae point is often represented in 3-dimensional space, where a orientation attribute is included. In this case, it is necessary to investigate how to make use of this third attribute.

Furthermore, the actual fingerprint verification system will also depend on the performance the minutiae point extraction algorithm. It might not be straight forward to incorporate our scheme. Our scheme also requires that the fingerprints to be pre-aligned in a consistent way, which may be difficult with the current techniques.

Other biometrics: Besides fingerprints, there are other biometrics that are possible candidates for authentications. For example, hand geometry, voice, retina, iris pattern, face, etc. Each of these biometrics has different characteristics such that we will not be able to adapt the existing schemes easily. For example, some state-of-the-art facial recognition techniques extract feature points in 3-D space, where each point may have its own semantics. It would be interesting to investigate how the ideas of helper information can be extended to handle such feature points and/or other biometrics.

Other distance metrics: Under our setting, we considered the combination of white noise and replacement noise on the point-sets, which correspond to infinity-norm and set difference, respectively, in terms of distance metric.

It is also interesting to consider other distance metrics or different combinations of them. For example, suppose an authentication system is designed in such a way that a user needs to enter five passwords to be authenticated, each password is considered as correct if and only if at most two symbols are not the same as the original password, and the user is authenticated if four out of the five passwords are correct. In this example, we could consider the combination of set difference and Hamming distance.

Chapter 4

Conclusions

In this thesis, we study the security aspect of digital image watermarking and biometric authentication, where the attackers are smart.

For digital watermarking schemes, we consider the inversion attacks, where the attackers try to create ambiguities on the ownership of a work. We show that it is possible to construct a non-invertible watermarking scheme by giving a scheme and prove its security using well accepted methods in cryptography. We also consider group watermark detection, where the watermark detection routine is designated to some proxies. Under our setting, each individual is not trusted, but we show that a certain level of trust can be established on the group as a whole, when the majority is honest. Next, we consider oracle attacks, or sensitivity attacks, on a class of spread-spectrum based watermarking schemes on binary sequences. We relate this type attacks to the Twenty Questions Games, and give both a lower bound and a matching upper bound of the effort required by the attackers.

In this thesis we also study the problem of biometric authentication by making use of helper information. In particular, we examine the problem of secure authentication of fingerprints that are represented by minutia points in a 2-dimensional space. We observe that the problem is reduced to the construction of helper information with respect a combination of white noise and replacement noise on point-sets. We give a general construction with provably small entropy loss, and give some improvements for certain special cases.

We further extend the idea of helper information in image authentication, where the images are represented by a vector of coefficients. We construct helper information from these coefficients and use it as the authentication tag. We consider the robustness, sensitivity, and the size of the authentication tag, and show how to obtain optimal parameters under different constraints.

Appendix A

A.1 List of Publications

- Qiming Li, Ee-Chien Chang and Mun Choon Chan. *On the Effectiveness of DDoS Attacks on Statistical Filtering.* In IEEE INFOCOM, 2005.
- Qiming Li and Ee-Chien Chang. *On the Possibility of Non-Invertible Watermarking Schemes.* Information Hiding Workshop, volume 3200 of LNCS, pages 13-24, 2004. Springer Verlag.
- Qiming Li and Ee-Chien Chang, *Public Watermark Detection Using Multiple Proxies and Secret Sharing.* International Workshop on Digital Watermarking, volume 2939 of LNCS, pages 558–569, 2003. Springer Verlag.
- Qiming Li and Ee-Chien Chang, *Security of Public Watermarking Schemes for Binary Sequences.* Information Hiding Workshop, volume 2578 of LNCS, 119-128, 2002. Springer Verlag.

A.2 Papers under Review

- Secure Sketch for Point-Sets

Bibliography

[1] A. Adelsbach, S. Katzenbeisser, and A. Sadeghi. On the insecurity of non-invertible watermarking schemes for dispute resolving. In *2nd International Workshop on Digital Watermarking*, 2003.

[2] A. Adelsbach, S. Katzenbeisser, and H. Veith. Watermarking schemes provably secure against copy and ambiguity attacks. *ACM Workshop on Digital Rights Management*, pages 111–119, 2003.

[3] A. Adelsbach and A. Sadeghi. Zero-knowledge watermark detection and proof of ownership. *4th Int. Workshop on Info. Hiding*, LNCS 2137:273–288, 2000.

[4] N. Alon, O. Goldreich, J. Håstad, and R. Peralta. Simple constructions of almost k-wise independent random variables. In *Proc. of the 31st FOCS*, pages 544–553, 1990.

[5] A. Ambainis, S.A. Bloch, and D.L. Schweizer. Playing twenty questions with a procrastinator. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 844–845. ACM Press, 1999.

[6] G.R. Arce, L. Xie, and R.F. Graveman. Approximate image authentication codes. In *Proc. 4th Annual Fedlab Symp. on Advanced Telecommunications/Information Distribution*, 2000.

[7] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *STOC*, pages 1–10, 1988.

[8] L. Blum, M. Blum, and M. Shub. A simple secure unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15:364–383, 1986.

[9] X. Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM conference on Computer and Communications Security*, pages 82–91. ACM Press, 2004.

[10] E.C. Chang and M. Orchard. Geometric properties of watermarking schemes. In *ICIP*, volume 3, pages 714–717, 2000.

[11] B. Chen and G.W. Wornell. Achievable performance of digital watermarking systems. *IEEE Int. Conf. on Multimedia Computing & Systems*, 1:13–18, 1999.

[12] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *FOCS*, pages 383–395, 1985.

[13] J. Chou, S.S. Pradhan, and K. Ramchandran. On the duality between distributed source coding and data hiding. *33rd Asilomar conference on Signals, System and Computers*, pages 1503–1507, 1999.

[14] T.C. Clancy, N. Kiyavash, and D.J. Lin. Secure smartcard-based fingerprint authentication. In *ACM Workshop on Biometric Methods and Applications*, 2003.

[15] M. Costa. Writing on dirty paper. *IEEE Trans. on Information Theory*, 29(3):439–441, 1983.

[16] I.J. Cox and J-.P. Linmartz. Public watermarks and resistance to tampering. *IEEE Int. Conf. on Image Processing*, 3(0_3–0_6), 1997.

[17] I.J. Cox, M.L. Miller, and J.A. Bloom. *Digital Watermarking.* Morgan Kaufmann, 2002.

[18] S. Craver. Zero knowledge watermark detection. *3rd Intl. Workshop on Information Hiding*, LNCS 1768:101–116, 2000.

[19] S. Craver and S. Katzenbeisser. Copyright protection protocols based on asymmetric watermarking. In *CMS'01*, pages 159–170, 2001.

[20] S. Craver, N. Memon, B.L. Yeo, and M.M. Yeung. Resolving rightful ownerships with invisible watermarking techniques: Limitations, attacks, and implications. *IEEE Journal on Selected Areas in Communications*, 16(4):573–586, 1998.

[21] A. Dhagat, P. Gács, and P. Winkler. On playing "twenty questions" with a liar. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 16–22. ACM Press, 1992.

[22] G. DiCrescenzo, R. Graveman, G. Arce, and R. Ge. A formal security analysis of approximate message authentication codes. In *Proc. CTA Comm. and Networks*, 2003.

[23] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Eurocrypt'04*, volume 3027 of *LNCS*, pages 523–540. Springer-Verlag, 2004.

[24] J.J. Eggers, J.K. Su, and B. Girod. Public key watermarking by eigenvectors of linear transforms. In *European Signal Processing Conference*, 2000.

[25] J.J. Eggers, J.K Su, and B. Girod. Asymmetric watermarking schemes. *Sicherheit in Mediendaten*, September 2000.

[26] P. Feldman. A practical scheme for non-interaive verifiable secret sharing. In *FOCS*, pages 427–437, 1987.

[27] T. Furon and P. Duhamel. An asymmetric public detection watermarking technique. *3rd Intl. Workshop on Information Hiding*, LNCS 1768:88–100, 2000.

[28] M.D. Garris and R.M. McCabe. Fingerprint minutiae from latent and matching tenprint images. *NIST Special Database 27*, 2000.

[29] R. Graveman and K. Fu. Approximate message authentication codes. In *Proc. 3rd Annual Fedlab Symp. on Advanced Telecommunications/Information Distribution*, 1999.

[30] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. *CRYPTO'95*, LNCS 963:339–352, 1995.

[31] M. Hirt, U. Maurer, and B. Przydatek. Efficient secure multi-party computation. In *ASIACRYPT'00*, volume LNCS 1976, pages 143–161, 2000.

[32] E.G. Coffman Jr., L. Flatto, and P. Jelenković. Interval packing: the vacant interval distribution. *The Annals of Applied Probability*, 10(1):240–257, 2000.

[33] A. Juels and M. Sudan. A fuzzy vault scheme. In *IEEE Intl. Symp. on Information Theory*, 2002.

[34] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proc. ACM Conf. on Computer and Communications Security*, pages 28–36, 1999.

[35] Q. Li and E.-C. Chang. Security of public watermarking schemes for binary sequences. In *Information Hiding Workshop*, volume 2578 of *LNCS*, pages 119–128. Springer Verlag, 2002.

[36] D. Maltoni, D. Maio, A.K. Jain, and S. Prabhakar. *Handbook of Fingerprint Recognition.* Springer-Verlag, 2003.

[37] Y. Minsky, A. Trachtenberg, and R. Zippel. Set reconciliation with nearly optimal communications complexity. In *ISIT*, 2001.

[38] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Attacks on copyright marking systems. *2nd Intl. Workshop on Information Hiding*, LNCS 1525:219–239, 1998.

[39] P.A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach.* The MIT Press, 2000.

[40] L. Qiao and K. Nahrstedt. Non-invertible watermarking methods for MPEG encoded audio. In *Proceedings of the SPIE 3675, Security and Watermarking of Multimedia Contents*, pages 194–202, 1998.

[41] L. Qiao and K. Nahrstedt. Watermarking schemes and protocols for protecting rightful ownerships and customer's rights. *Journal of Visual Communication and Image Representation*, 9(3):194–210, 1998.

[42] M. Ramkumar and A. Akansu. Image watermarks and counterfeit attacks: Some problems and solutions. In *Symposium on Content Security and Data Hiding in Digital Media*, pages 102–112, 1999.

[43] J. B. Rosser and L. Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois J. Math*, 6:64–97, 1962.

[44] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[45] P. Tuyls and J. Goseling. Capacity and examples of template-protectiong biometric authentication systems. *Biometric Authentication Workshop*, pages 158–170, 2004.

[46] S. Ulam. *Adventures of a mathematician.* Scribner and Sons, 1976.

[47] L. Xie, G.R. Arce, and R.F. Graveman. Approximate image message authentication codes. In *IEEE Trans. on Multimedia*, pages 242–252, 2001.

[48] A. Yao. Probabilistic computations: Toward a unified measure of complexity. *18th IEEE Symposium on Foundations of COmputer Science*, pages 222–227, 1977.

[49] A. Yao. Theory and application of trapdoor functions. *23rd IEEE Symposium on Foundation of Computer Science*, pages 80–91, 1982.

[50] W. Zhang, Y.-J. Chang, and T. Chen. Optimal thresholding for key generation based on biometrics. *Int. Conf. on Image Processing*, 2004.