A Survey on Traitor Tracing Schemes

by

Jason Qiang Chen

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2000

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

The University of Waterloo requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

When intellectual properties are distributed over a broadcast network, the content is usually encrypted in a way such that only authorized users who have a certain set of keys, can decrypt the content. Some authorized users may be willing to disclose their keys in constructing a pirate decoder which allows illegitimate users to access the content. It is desirable to determine the source of the keys in a pirate decoder, once one is captured. Traitor tracing schemes were introduced to help solve this problem. A traitor tracing scheme usually consists of: a scheme to generate and distribute each user's personal key, a cryptosystem used to protect session keys that are used to encrypt/decrypt the actual content, and a tracing algorithm to determine one source of the keys in a pirate decoder. In this thesis, we survey the traitor tracing schemes that have been suggested. We group the schemes into two groups: *symmetric* in which the session key is encrypted and decrypted using the same key and *asymmetric* schemes in which the session key is encrypted and decrypted using different keys. We also explore the possibility of a truly public scheme in which the data supplier knows the encryption keys only. A uniform analysis is presented on the efficiency of these schemes using a set of performance parameters.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Consider an application which provides data that should be available to authorized users only. The number of authorized users is big enough so that broadcasting the data is much more efficient than establishing a secure channel between the data provider and each authorized user. The data could obviously be protected from unauthorized access by encryption. And the data supplier could provide the decryption keys to the authorized users only, and broadcast the encrypted ciphertext. However this does not prevent one or more authorized users from retransmitting the plaintext they have obtained by decrypting the received ciphertext, or simply disclosing their personal keys to some unauthorized users. In this event, unauthorized users have access to data that they are not entitled to.

We call this unauthorized access *piracy*. The *traitors* are the groups of authorized users who allow unauthorized users to obtain the data, either by retransmitting the plaintext, or disclosing their personal decryption keys. The unauthorized users who obtained the data are called *pirate users*.

Traitor tracing schemes or *traceability schemes* are cryptographic techniques

preventing traitors from distributing their personal keys to enable pirates decrypting the data. In such a scheme, each authorized user is given a decoder which contains the user's personal decryption key. A symmetric encryption algorithm (such as DES, AES) is used to encrypt the data using a randomly generated session key. The data distributer encrypts the session key in a way such that only an authorized user's decoder is able to decrypt the session key and hence recover the data. Suppose a group of traitors contribute their personal keys to build a pirate decoder which can also decrypt the ciphertext. The scheme should discover the keys in the pirate decoder and determine one or more traitors who have helped build the pirate decoder by contributing their personal keys. The following is a scheme which can trace the traitor if there is only one traitor.

Example 1.0.1 *There are n users:$\{u_1, u_2, \cdots, u_n\}$, and $2\log n$ keys:*

$$\alpha = \{k_{1,0}, k_{1,1}, k_{2,0}, k_{2,1}, \cdots, k_{\log n,0}, k_{\log n,1}\}.$$

We assume n is a power of 2. The personal key for user u_i is the set of $m = \log n$ keys:

$$\{k_{1,b_{i,1}}, k_{2,b_{i,2}}, \cdots, k_{\log n,b_{i,\log n}}\},$$

where $b_{i,j}$ is the j-th bit in the binary representation of i. Suppose we use DES to encrypt the content M using session key s. Let $\mathcal{B}_m = DES_s(M)$ denote the ciphertext. We choose $s_1, s_2, \cdots, s_{\log n}$, such that $\bigoplus_{i=1}^{\log n} s_i = s$. Then s_i is encrypted using keys $k_{i,0}, k_{i,1}$. Let

$$\mathcal{B}_e = \|_{i=1}^{\log n} \left(DES_{k_{i,0}}(s_i) \| DES_{k_{i,1}}(s_i) \right),$$

where $\|$ denotes concatenation. Both of $\mathcal{B}_m$ and $\mathcal{B}_e$ are broadcasted. Notice that each user u_i has keys $k_{j,b_{i,j}}$, $1 \le j \le \log n$. u_i can decrypt all s_j's and hence obtain

s. Suppose a user u_c reveals his personal key:

$$\{k_{1,b_{c,1}}, k_{2,b_{c,2}}, \cdots, k_{\log n,b_{c,\log n}}\}.$$

Since each user's personal key is unique, we can easily find the identity of the user from the disclosed key.

Traitor tracing schemes do not address treachery where the traitors retransmit the plaintext they obtained. This is in contrast to normal *fingerprinting schemes*, which address the situation where traitors redistribute the pictures, text, or programs they obtain. This is achieved by embedding an unique *fingerprint* to each copy of data distributed, so that the data distributor can identify the original receiver of a redistributed copy. In principle in traitor tracing schemes, the keys used to decrypt the encrypted the session key are fingerprinted instead of the data itself. Fingerprinting schemes are not suitable to broadcasting, since the copy received by each user must differ from the copies received by others. In traitor tracing schemes, only one copy of the content is broadcasted.

In some applications, it might be valid to assume that redistributing the plaintext is too costly. A typical application would be pay-TV, where it is obviously too expensive and too risky to operate a pirate broadcast system. We are also going to assume that the session key is changed frequently such that broadcasting the session key is also infeasible and risky.

One trivial solution is to encrypt the data separately under different personal keys and broadcast all ciphertexts encrypted under every key. This means that the amount of the data broadcasted is at least n times the amount of the original data, where n is number of authorized users. Such a solution is clearly not very efficient.

The intent of this thesis is to survey the traitor tracing schemes have been suggested and perform a uniform analysis of the efficiency of these schemes. We are

going to measure the efficiency of each scheme in terms of the following performance parameters:

1. storage and computational requirements at the data distributor/broadcastor side,
2. storage and computational requirements at the user side,
3. bandwidth requirement,
4. computational requirements by the tracing algorithm, i.e., given a pirate decoder, the computational cost to determine one or more traitors.

We begin in Chapter 2 with a list of definitions. In Chapter 3 we describe symmetric schemes in which the session key is encrypted using a symmetric key system. We present asymmetric schemes in which the session key is encrypted and decrypted using different keys but the data supplier knows both keys in Chapter 4; and investigate the possibility of constructing truely public key schemes in which the data supplier knows only the encrytion key but not the decryption key in Chapter 5. A couple of other schemes are discussed in Chapter 6. And finally Chaper 7 gives a conclusion.

Chapter 2

Definitions

Traitor tracing scheme was first introduced by Chor, Fiat and Naor [6] in 1994. Since then, other traitor tracing schemes were suggested by B. Pfitzmann in [13], Stinson and Wei in [15], Dwork, Lotspiech, and Naor in [7], Anderson and Manifavas in [2], Kurosawa and Desmedt in [10], Boneh and Franklin in [3], Fiat and Tassa in [8]. A typical traitor tracing scheme consists of three components:

1. *key generation/distribution scheme*: used by the data supplier to generate and distribute users' personal keys. The data supplier has a *master-key* α that defines a mapping $P_\alpha : U \mapsto K$, where U is the set of possible users and K is the set of all possible personal keys. In Example 1.0.1,

$$P_\alpha(u_i) = \{k_{1,b_{i,1}}, k_{2,b_{i,2}}, \cdots, k_{\log n, b_{i,\log n}}\},$$

   where $b_{i,j}$ is the j-th bit in the binary representation of i.

2. *encryption/decryption scheme*: an encryption scheme E_α is used by the data supplier to encrypt the session key before broadcasting, and a decryption

scheme D_β is used by each authorized user (i.e. its decoder) to decrypt the session key, where $\beta = P_\alpha(u_i)$. Obviously, for any session key s, $s = D_\beta(E_\alpha(s))$. And the session key is used to encrypt data content using a off-the-shelf symmetric encryption scheme such as DES.

3. *tracing algorithm*: used upon confiscation of a pirate decoder, to determine the identity of one or more traitors. We are not going to assume the content of a pirate decoder can always be viewed by the tracing algorithm. We prefer that the tracing algorithm is only able to access any pirate decoder as a black box and perform the tracing based on the decoder's respond on different input ciphertexts.

We usually require the off-the-shelf encryption scheme E to be a block cipher. The data supplier divides the content into sessions whose size is a multiple of a block size accepted by E. For each content session M, a typical traitor tracing scheme will output two blocks. A *ciphertext block* $\mathcal{B}_c$ is the result of encrypting M by the encryption scheme E using some key s randomly chosen from the key space of E: $\mathcal{B}_c = E_s(M)$. We call s the *session key*. A second block is called an *enabling block*, because it contains data that enables each authorized user to obtain the session key s, and hence decrypt the corresponding ciphertext block (see Figure 2.1. In Example 1.0.1, the enabling block is

$$\mathcal{B}_e = \|_{i=1}^{\log n} \left(DES_{k_{i,0}}(s_i) \| DES_{k_{i,1}}(s_i) \right).$$

An obvious and preliminary requirement from traitor tracing schemes is that the underlying encryption scheme used to encrypt session keys must be computationally secure. That is, an adversary which has no information on any personal keys should not be able to decrypt the session key. (It is assumed that the off-the-shelf is

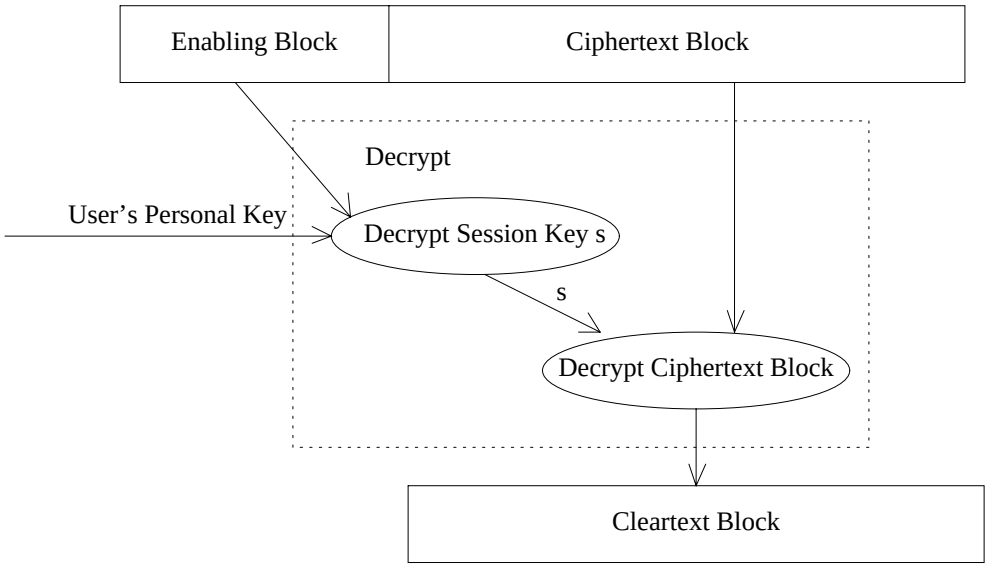


Figure 2.1: The Decryption of a Ciphertext Block

computationally secure.) Therefore, if there is a pirate decoder which decrypts the ciphertext with non-negligible probability, it must contain certain information on some personal keys. It is assumed that these personal keys belong to the traitors who helped construct the pirate decoder. The tracing algorithm should be able to determine at least one of the traitors.

Suppose that a coalition of k users collude to create a pirate decoder. We would like to determine at least one member of the coalition. Intuitively, a traitor tracing scheme is *fully resilient* if it can identify (with high certainty) at least one member of the coalition, given that the underlying encryption scheme is not broken.

Definition 2.0.1 *Suppose the underlying encryption scheme is computationally secure. A scheme is fully (p, k)-resilient if for every pirate decoder constructed by a coalition of at most k traitors and decrypts ciphertext correctly with a probability greater than the probability of breaking the underlying encryption scheme, the trac-*

ing algorithm can trace at least one traitor of the coalition with probability at least $1-p$. *If the scheme further achieves* $p=0$, *it is called a fully* k*-resilient.*

Example 1.0.1 is a fully 1-resilient traitor tracing scheme.

There are many applications for which the pirate decoder must decrypt with probability close to 1. For example, if a TV broadcast is partitioned into m segments and these m segments are encrypted independently, then a decoder which decrypts only 90 percent of the segments is probably not very useful or attractive. In this circumstance, we can concentrate on tracing the pirate decoders that can decrypt with probability greater than a certain threshold.

Definition 2.0.2 *Suppose the underlying encryption scheme is computationally secure. A scheme is called q-threshold (p,k)-resilient scheme if for every pirate decoder constructed by a coalition of at most k traitors and decrypts ciphertext correctly with a probability greater than q, the tracing algorithm can trace at least one member of the coalition with probability at least* $1-p$.

We can further distinguish between two types of schemes:

Definition 2.0.3 *A scheme is called an open scheme if it treats circumstances where the key generation/distribution scheme and decryption schemes used by all users are in the public domain, whereas the master key is the only information that is kept secret.*

Definition 2.0.4 *A scheme is called a secret scheme if the actual key generation/distribution scheme as well as the keys are kept secret.*

Clearly, the adversary's task is harder with a secret scheme compared to an open scheme. In general, secret schemes are more efficient than open schemes that achieve the same level of security and traceability. However, we do not encourage to base the security and traceability of a scheme on the premise that the adversary does not know the key generation/distribution scheme. We would prefer a scheme which achieves the desired security and traceability under Kerckhoff's principle, that is the adversary knows the complete scheme except for the key.

Chapter 3

Symmetric Schemes

Chor, Fiat and Naor first introduced traitor tracing schemes in [6]. With Pinkas in 1998, they further suggested threshold traitor tracing schemes [5]. Six different types of traitor tracing schemes were mentioned: one level open scheme, two level open scheme, one level secret scheme, two level secret scheme, one level threshold scheme, and two level threshold scheme. We call these schemes Chor-Fiat-Naor-Pinkas (CFNP) schemes. Example 1.0.1 is a one level open CFNP scheme. The existence of CFNP schemes were proved using probabilistic method. But the proofs are not constructive. Stinson and Wei provided several constructions for a variant of the one level open scheme in [14].

In this chapter we are going to present the six CFNP schemes, as well as the SW schemes suggested by Stinson and Wei. All the schemes in this chapter are symmetric in the sense that the session keys are encrypted and decrypted using same set of keys.

3.1 Overview

Throughout the rest of this thesis we are going to use the following notations. U is the set of authorized users, and $n = \mid U \mid$. k is the upper bound on the number of traitors. T is a set of traitors such that $\mid T \mid \leq k$.

The master key α is actually a set of keys randomly chosen from the key space of the off-the-shelf symmetric encryption scheme E. We call this set of keys *base keys*. In Example 1.0.1,

$$\alpha = \{k_{1,0}, k_{1,1}, k_{2,0}, k_{2,1}, \cdots, k_{\log n,0}, k_{\log n,1}\}.$$

A session key s is divided into several shares using secret sharing schemes, so that to contruct s, a certain subset of shares must be obtained. The shares are encrypted using E.

CFNP schemes employ a set of hash functions to assign each user u a subset of the base keys α. We call this subset of keys *personal key* of u, and denote it $P(u)$. The assignment must satisfy two properties. First, it must ensure each user is able to decrypt enough shares to construct the session key s. In Example 1.0.1, the key distribution must ensure a user has at least one key in $K_j = \{k_{j,0}, k_{j,1}\}$, $1 \leq j \leq \log n$, so that the user can decrypt s_j. Furthermore, the assignment must guarantee that any set F of keys, taken from a set of users T, $\mid T \mid \leq k$, $(F \subseteq P(T) = \cup_{v \in T} P(v))$, has the following property: if F enables the decryption of session key s, then there does not exist an innocent user $u \in U \setminus T$, such that $\mid F \cap P(u) \mid \geq \mid F \cap P(v) \mid$, for all $v \in T$. This property will enable us to identify at least one member of T, as there exists at least one $v \in T$, such that $\mid F \cap P(v) \mid \geq \mid F \cap P(u) \mid$, for all $u \in U$. Then v is called an *exposed user*.

Assume E is compuationally secure. To decrypt the cipher block, an adversary

must know the session key s. In order to construct s, an adversary has to obtain a subset of secret shares that satisfes a certain property. Since each share is encrypted by E and E is computationally secure, we can assume that the adversary must possess a subset of base keys with a certain property. This property plays an important role in tracing the identity of the owners of the keys the adversary has. In Example 1.0.1, the adversary has to obtain all the shares $s_1, s_2, \cdots, s_{\log n}$. He must have at least one key from each of $K_i, 1 \leq i \leq \log n$, since each share s_i is encrypted by E using keys in K_i.

In the rest of this section, we are going to prove an important result which is used frequently in the following sections where we present the six CFNP schemes.

First we state a standard theorem from [1] (Theorem A.12, page 237).

Theorem 3.1.1 (Chernoff Bound:) *Assume:*

$$
\begin{aligned}
& p_1, p_2, \cdots, p_n \in [0,1], \\
& p = \frac{p_1 + p_2 + \cdots + p_n}{n}, \\
& X_1, X_2, \cdots, X_n \textit{ independent random variables with} \\
& \quad Pr[X_i = 1 - p_i] = p_i \\
& \quad Pr[X_i = -p_i] = 1 - p_i \\
& X = X_1 + X_2 + \cdots + X_n
\end{aligned}
$$

Then, for all $\beta \geq 1$, we have

$$Pr[X \geq (\beta - 1)pn] < (\frac{e^{\beta - 1}}{\beta^{\beta}})^{pn}.$$

□

We are going to prove the following corollary of the Chernoff Bound.

Corollary 3.1.2 *With the same assumptions as in the above theorem except,*

$$Pr[X_i = 1] = p_i$$
$$Pr[X_i = 0] = 1 - p_i$$

Then, for all $\beta \geq 1$, we have

$$Pr[X \geq \beta pn] < (\frac{e^{\beta-1}}{\beta^\beta})^{pn}.$$

Proof: Define random variables, $Y_1, Y_2, \cdots, Y_n$,

$$Y_i = X_i - p_i, 1 \leq i \leq n,$$

Then,

$$Pr[Y_i = 1 - p_i] = Pr[X_i = 1] = p_i$$
$$Pr[Y_i = -p_i] = Pr[X_i = 0] = 1 - p_i$$

So, by Chernoff Bound we have,

$$Pr[Y \geq (\beta - 1)pn] < (\frac{e^{\beta-1}}{\beta^\beta})^{pn}.$$

But, $X = Y + \sum_{i=1}^{n} p_i = Y + pn$. Therefore,

$$\begin{aligned} Pr[X \geq \beta pn] &= Pr[Y + pn \geq \beta pn] \\ &= Pr[Y \geq (\beta - 1)pn] \\ &< (\frac{e^{\beta-1}}{\beta^\beta})^{pn}. \end{aligned}$$

□

3.2 One Level Open Scheme

Key distribution: The master key α in the one level open scheme introduced here contains $2k^2l$ base keys randomly chosen from the key space of the underlying encryption scheme E, where l is a parameter we will explain later. The keys are partitioned into l buckets, such that each bucket contains $2k^2$ keys. Let $B_1, B_2, \cdots, B_l$ denote the buckets and let $k_{i,j}, 1 \leq i \leq l, 1 \leq j \leq 2k^2$ denote the jth key in B_i. In Example 1.0.1, there are $\log n$ buckets and each bucket has two keys: $B_i = \{k_{i,0}, k_{i,1}\}$.

The scheme employs l hash functions $h_1, h_2, \cdots, h_l$. Each hash function

$$h_i : U = \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, 2k^2\}, 1 \leq i \leq l$$

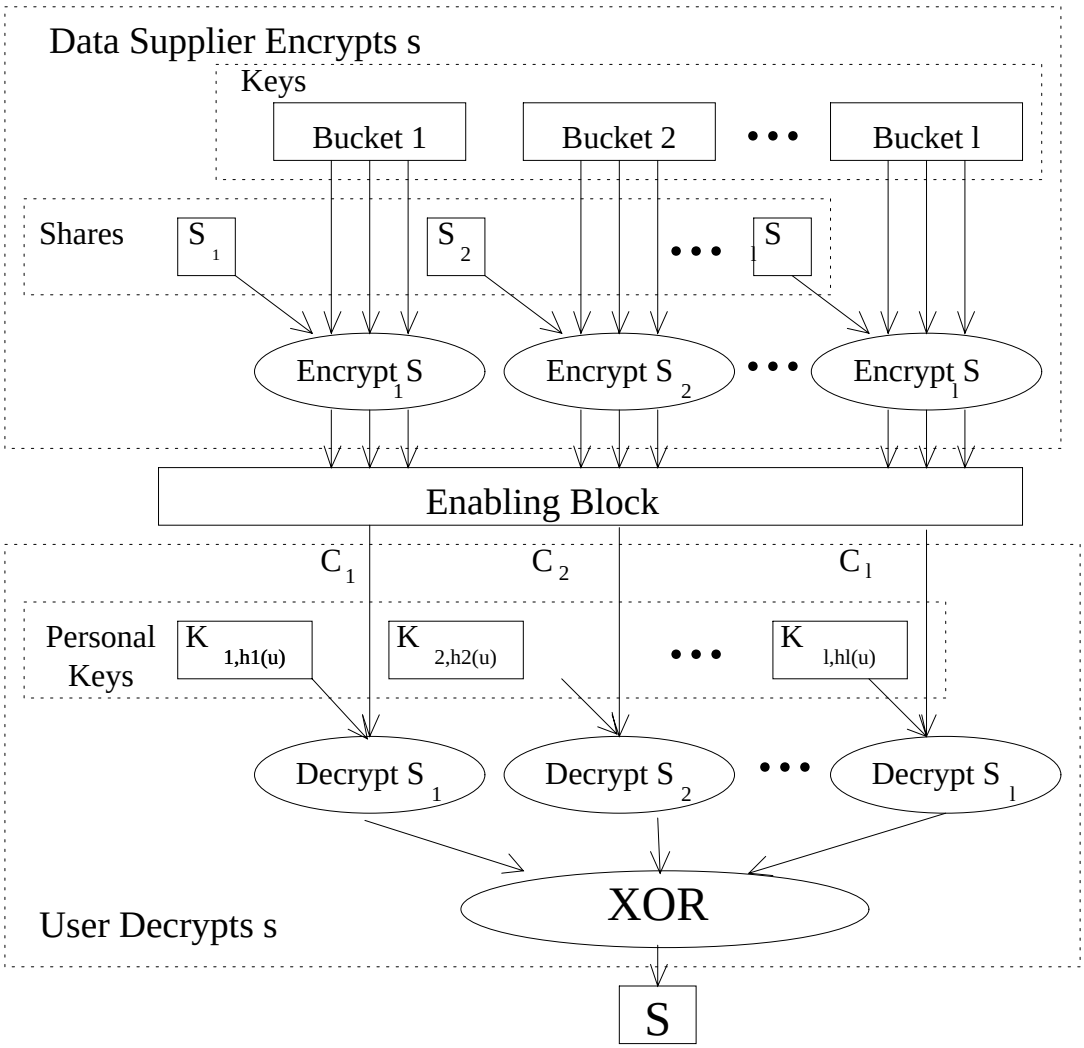
assigns one key from bucket B_i to each user. Thus, a user $u \in U$ gets personal key:

$$P(u) = \{k_{1,h_1(u)}, k_{2,h_2(u)}, \cdots, k_{l,h_l(u)}\}.$$

Encryption: For each plaintext session M, a key s is randomly chosen in the key space of E. s is divided into l shares, $s_1, s_2, \cdots, s_l$, such that $s = \bigoplus_{i=1}^{l} s_l$. Then s_i is encrypted using E with all keys in bucket i, $1 \leq i \leq l$. The concatenation of all these encrypted shares form the enabling block. So, $\mathcal{B}_e = \|_{i=1}^{l}(\|_{j=1}^{2k^2} E_{k_{i,j}}(s_i))$. The cipher block is simply $\mathcal{B}_c = E_s(M)$.

Since any authorized user $u \in U$ has one key in each bucket, u can decrypt all of $s_1, s_2, \ldots, s_l$, and hence reconstruct s and decrypt the plaintext M. And if an adversary wants to obtain s, he must know the values of all secret shares. Thus we can assume that if a decoder $\mathcal{D}$ can decrypt M with a non-negligible probability, then $\mathcal{D}$ must possess at least one key from each bucket.

Tracing: We assume the pirate decoder has at least one key from each bucket. Upon confiscation of a pirate decoder, we would like to expose at least one key from



$$C_i = E_{k_{i,h_i(u)}}$$

Figure 3.1: The Encryption and Decryption of a Session Key in a One Level CFNP Open Scheme

each bucket by experimenting with the decoder without taking the decoder apart, i.e., treating the decoder as a blackbox.

Figure 3.2 gives the tracing algorithm presented in [5] which identifies one of the traitors who helped build the pirate decoder by contributing their personal key. The algorithm takes a pirate decoder $\mathcal{D}$ as input and outputs one of the traitors. Lines 1 to 7 determine a subset F' of the keys in $\mathcal{D}$, such that F' has exactly one key in each bucket. Lines 8 to 11 identify one user u who has the maximum number of personal keys in common with F', i.e., $\mid F' \cap P(u) \mid \geq \mid F' \cap P(v) \mid$, for all $v \in U$.

Notice that experiment $E_{i,j}$ has the same effect as removing keys $k_{i,1}, k_{i,2}, \cdots, k_{i,j}$ from K_i, where K_i is the set of keys used to encrypt s_i. Suppose $\mathcal{D}$ uses key k_{i,c_i} in bucket i to recover the share s_i. Then $\mathcal{D}$ would fail E_{i,c_i} since key k_{i,c_i} is no longer valid in E_{i,c_i}. But $\mathcal{D}$ would successfully decode $E_{i,c_{i-1}}$.

Now what if $\mathcal{D}$ has more than one key in a bucket? Let F_i denote the keys $\mathcal{D}$ has in bucket i, $\mid F_i \mid \geq 2$. For each session, $\mathcal{D}$ randomly chooses one key from F_i to recover share s_i. The algorithm would fail if we performed each experiment $E_{i,j}$ once. For example, suppose $F_i = \{k_{i,1}, k_{i,3}\}$. $\mathcal{D}$ uses $k_{i,3}$ in $E_{i,1}$, so it decrypts successfully. But, $\mathcal{D}$ also uses $k_{i,1}$ in $E_{i,2}$. Since key $k_{i,1}$ is not longer valid in $E_{i,2}$, $\mathcal{D}$ will fail, and the algorithm would mistakenly conclude $\mathcal{D}$ has key $k_{i,2}$. What [5] suggested is to repeat $E_{i,j}$ a sufficient number of times and take the fraction of times $\mathcal{D}$ decrypts successfully. Suppose $F_i = \{k_{i,a_1}, k_{i,a_2}, \cdots, k_{i,a_{k'}}\}$,where $1 \leq a_1 < a_2 < \cdots < a_{k'} \leq 2k^2, k' \leq k$. Suppose the probability that key k_{i,a_j} is chosen by $\mathcal{D}$ to recover s_i is $Pr(j)$. Then the probability that $\mathcal{D}$ decrypts experiment $E_{i,j}$ successfully is $\sum_{1 \leq l \leq k', a_l > j} Pr(j)$. Let $f_{i,j}$ be the fraction of times $\mathcal{D}$ decrypts correctly on experiment $E_{i,j}$. Then $f_{i,j} = f_{i,j-1}$ if $k_{i,j} \notin F_i$ and $f_{i,j} < f_{i,j-1}$ if $k_{i,j} \in F_i$.

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. Let $E_{i,j}$ be the experiment: prepare a normal encryption session, but choose j random keys to replace $k_{i,1}, k_{i,2}, \cdots, k_{i,j}$.
2. for $1 \leq i \leq l$ do
3. for $0 \leq j \leq 2k^2$ do
4. repeat experiment $E_{i,j}$ r times,
5. let $f_{i,j}$ be the fraction of times $\mathcal{D}$ decrypts correctly on experiment $E_{i,j}$.
6. if $f_{i,j} - f_{i,c_{i-1}} > f_{i,0}/2k^2$
7. $c_i = j$, i++, goto 3
8. for each $u \in U$ do
9. for $1 \leq i \leq l$ do
10. crt_u++ if $h_i(u) = c_i$
11. return u such that $crt_u = MAX_{u \in U} crt_u$

Figure 3.2: Tracing Algorithm for One Level Open Scheme

However, if $\mathcal{D}$ has more than one key in a bucket, it can detect if it is being inquired by a tracing algorithm, by using all keys in F_i to decrypt share s_i. In an ordinary decoding operation, all values obtained should be the same. But in the tracing algorithm, if some of F_i are removed with respect to experiment $E_{i,j}$, the values of s_i obtained by using different keys would differ. If this happens, $\mathcal{D}$ knows it is being inquired by the tracing algorithm, and can intentionally decrypt incorrectly to fool the algorithm. Thus the suggested algorithm does not work well if $\mathcal{D}$ has more than one key in a bucket.

Here, we assume $\mathcal{D}$ has exactly one key in each bucket and the tracing algorithm detects the set of keys $F' = \{k_{1,c_1}, k_{2,c_2}, \cdots, k_{l,c_l}\}$ in $\mathcal{D}$. We are going to mark a user $u \in U$ once for each of his personal keys in F' (line 10). Let crt_u denote the number of times u gets marked. Then $crt_u = \mid F' \cap P(u) \mid$. We claim that the user u with the largest value crt_u is a traitor. Notice that there are at most k traitors, and there are l keys in F'. Thus there is one traitor $t \in T$ such that $crt_t \geq l/k$. Clearly a user u will not be declared as a traitor unless $crt_u \geq l/k$.

Notice the algorithm always declares one user to be a traitor. We are going to show that there exists a choice of $h_1, h_2, \cdots, h_l$ such that the tracing algorithm won't mistakenly identify an innocent user as a traitor. We say a coalition T can *frame* an innocent user u (i.e., $u \notin T$) if T can build a pirate decoder $\mathcal{D}$ such that the tracing algorithm with $\mathcal{D}$ as the input will claim u as a traitor.

Lemma 3.2.1 *There exists a choice of $h_1, h_2, \cdots, h_l$, such that no coalition of k traitors can frame an innocent user not in the coalition.*

Proof: Let $h_1, h_2, \cdots, h_l$ be randomly chosen. We show that the probability that there exists a coalition of k traitors, and a user $u \in U \setminus T$, such that T can frame

u is less than 1. And hence, there must exist a choice of $h_1, h_2, \cdots, h_l$ such that no coalition of k traitors can frame any innocent user.

Consider a specific user u, and a specific coalition $T \subseteq U$ of k traitors, $u \notin T$. Suppose T builds a pirate decoder $\mathcal{D}$ such that $\mathcal{D}$ can decrypt with a non-negligible probability. $\mathcal{D}$ is inputted to the tracing algorithm. Let $A_{u,T}$ denote the event that the tracing algorithm will identify u as a traitor, i.e., T can frame u. Let $A'_{u,T}$ denote the event that at the termination of the tracing algorithm, we have $crt_u \geq l/k$. Clearly, $Pr(A_{u,T}) \leq Pr(A'_{u,T})$, since there exists a traitor t with $crt_t \geq l/k$.

As the hash functions are chosen at random, the values $h_i(u)$ are uniformly distributed in $\{1, 2, \cdots, 2k^2\}$, and so the key $k_{i,h_i(u)}$ is uniformly distributed in bucket B_i. Let $F = \cup_{t \in T} P(t)$, i.e., F is the set of personal keys belonging to the traitors in T. Let $F_i = F \cap B_i$, i.e., F_i is the set of personal keys in bucket i belong to the traitors in T. Then $\mid F_i \mid \leq k$, and the probability that $k_{i,h_i(u)}$ is in F_i is at most $\frac{k}{2k^2} = \frac{1}{2k}$, since there are $2k^2$ keys in B_i. Note that, since this is a open scheme, the traitors might know which key in B_i is assigned to u. Also they can build a decoder that the key $k_{i,h_i(u)}$ is exposed by the tracing algorithm, if $k_{i,h_i(u)} \in F_i$, so that u would get more marks.

Let X_i be a zero-one random variable, where

$$X_i = \begin{cases} 1 & \text{if there exists } t \in T \text{ such that } h_i(t) = h_i(u) \\ 0 & otherwise. \end{cases} \tag{3.1}$$

The expected value of X_i, $E(X_i) \leq \frac{1}{2k}$, and the expected value of $\sum_{i=1}^{l} X_i$, $E(\sum_{i=1}^{l} X_i) \leq \frac{l}{2k}$. Notice that $\sum_{i=1}^{l} X_i$ is not smaller than the number of marks user u gets, since $F' \subseteq F$.

Now, apply Corollary 3.1.2, substituting $p = \frac{1}{2k}$, $r = l$, and $\beta = 2$. We have,

$$\begin{aligned} Pr[\frac{1}{l}\sum_{j=1}^{l} X_j \geq 2\frac{1}{2k}] &< (\frac{e^{2-1}}{2^2})^{\frac{l}{2k}} \\ \Rightarrow Pr[\frac{1}{l}\sum_{j=1}^{l} X_j \geq \frac{1}{k}] &< (\frac{e}{4})^{\frac{l}{2k}} \\ &= (\frac{1}{4/e})^{\frac{1}{2k}} \\ &< (\frac{1}{\sqrt{2}})^{\frac{1}{2k}} \\ &= (2^{-\frac{l}{4k}}) \end{aligned}$$

The probability that T is able to frame u is $Pr(A_{u,T}) \leq Pr(A'_{u,T}) < 2^{-\frac{l}{4k}}$. There are at most $n \cdot \begin{pmatrix} n \\ k \end{pmatrix}$ possible choise of a coalition of size k and an innocent user. Let A denote the event that there exists a coalition of size k which can frame an innocent user. Then

$$\begin{aligned} Pr(A) &= \sum_{u \in U} \left(\sum_{T \subseteq U, |T|=k, u \notin T} Pr(A_{u,T}) \right) \\ &< n \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{-\frac{l}{4k}}. \end{aligned}$$

We would like to choose a value for l, so that the above probability is less than 1.

Suppose we take $l \geq 4k(k+1)\log n$. Then we have

$$\begin{aligned} Pr(A) &= n \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{-\frac{l}{4k}} \\ &< n \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{-(k+1)\log n} \end{aligned}$$

$$\begin{aligned} &\leq n \cdot n^k \cdot (2^{\log n})^{-(k+1)} \\ &= n^{k+1} \cdot n^{-(k+1)} \\ &= 1. \end{aligned}$$

Therefore there exists a choice of l hash functions $h_1, h_2, \cdots, h_l$, such that for any pirate decoder built by any coalition T of size k, no innocent user will be incriminated by the tracing algorithm. □

By using the hash functions $h_1, h_2, \cdots, h_l$ with the above property, we have a fully k-resilient traitor tracing scheme.

Theorem 3.2.2 *There is an open fully k-resilient scheme, where the data supplier's master key consists of* $8k^3(k+1)\log n$ *base keys, and a user's personal key consists of* $4k(k+1)\log n$ *keys.* □

Notice that the proof of Lemma 3.2.1 is not constructive. Thus Theorem 3.2.2 shows only the existence of a open fully k-resilient traceability schemes. Although it does provide us with a randomized method for constructing the scheme that works with high probability, it does not suggest an explicit construction for a deterministic scheme. However, the desired property of a given construction can be verified efficiently. The idea is to examine all pairs of distinct users u,v, and check the number of hash functions $h_i, 1 \leq i \leq l$ in the given construction such that $h_i(u) = h_i(v)$.

Theorem 3.2.3 *Given hash functions* $h_i : \{1, 2, \cdots, n\} \mapsto \{1, 2, \cdots, d\}, 1 \leq i \leq l$, *if for every pair of distinct users* u, v, *the number of* h_i*'s such that* $h_i(u) = h_i(v)$ *is less than* l/k^2, *i.e.,* $\mid h_i, 1 \leq i \leq l : h_i(u) = h_i(v) \mid < l/k^2$, *then we have a k-resilient traceability scheme, with n users, ld base keys, where each user has l keys.*

Proof: Let T be a coalition of at most k traitors. Recall that there exists a traitor $t \in T$ such that t gets marked at least $\lceil l/k \rceil$ times by the tracing algorithm. We only have to show that there is no innocent user $u \in U \setminus T$ such that u gets marked at least $\lceil l/k \rceil$ times.

Notice, for any $t \in T$, $\mid h_i, 1 \leq i \leq l : h_i(u) = h_i(t) \mid < l/k^2$ Then the number of h_i's such that there *exists* a $t \in T$ such that $h_i(u) = h_i(t)$ is less than $l/k^2 \cdot k = l/k \leq \lceil l/k \rceil$. Thus, no innocent user will ever be marked at least $\lceil l/k \rceil$ times by the tracing algorithm. So, we have a k resilient traceability scheme. □

If we represent a user u's personal key as a codeword of length l, over an alphabet of size d, where d is the size of each bucket, i.e., $P(u) = (h_1(u), h_2(u), \cdots, h_l(u))$, we can rephase the requirement as follows.

Corollary 3.2.4 *If there is a code C with n codewords, with length l, over an alphabet of size d, such that the Hamming distance of C is more than $l - l/k^2$, then there is a k-resilient traceability scheme with n users, ld base keys, and each user has l keys.*

Proof: Since C has distance more than $l - l/k^2$, every two codewords must have less than l/k^2 entries in common. So, for every pair of distinct users u, v, we have $\mid h_i, 1 \leq i \leq l : h_i(u) = h_i(v) \mid < l/k^2$. □

The property stated in the above theorem is stronger than the property required by the one level scheme. The following example illustrates why.

Example 3.2.5 *Consider a scheme with 12 base keys, and 9 users, in which each user gets 4 keys. There are 4 buckets, each bucket has key 0, 1, and 2. The columns of the following matrix $A = [a_{ij}]$ represent the hash functions, h_1, h_2, h_3, h_4, each*

row represents the keys assigned to a user, $a_{ij} = h_j(i)$, i.e., the key in bucket j assigned to user i.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 2 & 1 & 1 & 0 \\ 2 & 2 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 2 \\ 1 & 2 & 2 & 0 \\ 2 & 0 & 2 & 2 \\ 0 & 2 & 1 & 2 \\ 0 & 1 & 2 & 1 \end{pmatrix}$$

We claim this is a fully 2-traceability scheme.

Let a, b be two codewords of length n over an alphabet Q. Define the set of descendants $D(a,b)$ by

$$D(a,b) := \{x \in Q^n | x_i \in \{a_i, b_i\}, 1 \leq i \leq n\}$$

Let $r_i, 1 \leq i \leq 9$, denote the ith row of A. Suppose a, b are two traitors. Then $D(r_a, r_b)$ is set of all possible key assignments a and b can construct in order to decrypt contents. Let r' denote the four keys found in a pirate decoder constructed by a, b. Then $r' \in D(r_a, r_b)$. And notice $d(r', r_a) + d(r', r_b) = d(r_a, r_b)$, where $d(x,y)$ denotes the hamming distance between $x, y \in Q^n$.

Now, notice that any two rows in A have hamming distance 3. So there exists $i \in \{a, b\}$, such that $d(r', r_i) \leq 1$. WLOG, say $d(r', r_a) \leq 1$, i.e., a will be marked at least 3 times by the tracing algorithm. Now consider $d(r', r_j), 1 \leq j \leq 9, j \notin \{a, b\}$. Suppose exists a j such that $d(r', r_j) \leq 1$, then $d(r_a, r_j) \leq d(r', r_j) + d(r', r_a) \leq$

$1 + 1 = 2$. *This contradicts the fact that any two rows in A have distance 3. Thus we conclude $d(r', r_j) \geq 2, 1 \leq j \leq 9, j \notin \{a, b\}$, i.e., no innocent user will be marked more than twice by the tracing algorithm. Thus we have a 2-resilient tracing scheme.*

In this scheme, we have $l = 4$, $k = 2$, and $l/k^2 = 1$. But any 2 rows have 1 entry in common. So $\mid \{h_i, 1 \leq i \leq 4 : h_i(1) = h_i(2)\} \mid = 1 \not< 1$. □

In the above example, rows of A form a ternary hamming code of length 4. Using the similar argument, we can prove the following theorem.

Theorem 3.2.6 *If there exists a equidistant code C of length l, over an alphabet Q, $\mid Q \mid = q$, with an odd distance, then there is a 2-traceability scheme, with $\mid C \mid$ users, lq base keys, each user has l keys.* □

We can also provide explicit constructions from transversal designs.

Definition 3.2.7 *A t-transversal design t-TD(l,m) consists a collection $\mathcal{B}$ of m-subsets (blocks) of a set $\mathcal{X}$ of lm elements (points), and a collection of l disjoint m-subsets (groups) which partition $\mathcal{X}$ such that,*

1. *each block contains exactly one element from each group,*
2. *any t elements from different groups occur in exactly one block.* □

Then the following corollary of Theorem 3.2.3 is straight forward.

Corollary 3.2.8 *If there is a t-TD(l,m) with $n = m^t$ blocks, then we have a k-traceability scheme with n users, lm base keys, and each user has l keys, where $k = \lfloor \sqrt{(l-1)/(t-1)} \rfloor$.*

Proof: Let $\mathcal{X}$ be the set of lm base keys, partition the lm keys into l groups of m keys each as in the t-TD(l,m) transversal design. Each user is assigned the keys in a unique block. Thus there can be n users and each user gets l keys. And any two users would have at most $t-1$ keys in common. But

$$\frac{l}{k^2} \geq \frac{l}{(l-1)/(t-1)} = \frac{l(t-1)}{l-1} > t-1.$$

Thus any two users have less than $\frac{l}{k^2}$ keys in common. By Theorem 3.2.3, we have a k-traceability scheme. □

$t-TD(l,m)$ designs can be constructed from designs known as *orthogonal arrays*.

Definition 3.2.9 *An orthogonal array $OA(t,l,s)$ is a $l \times s^t$ array, with entries from a set of $s \geq 2$ symbols, such that in any t rows, every $t \times 1$ column vector appears exactly once.* □

Theorem 3.2.10 *If there is a $OA(t,l,m)$, then there exists a t-TD(l,m) with m^t blocks.*

Proof: Suppose there is an $OA(t,l,m)$ on elements $0,1,\cdots,m-1$. Add $(i-1)m$ to each element in the ith row of the OA, so the elements in the ith row are in $\{(i-1)m,(i-1)m+1,(i-1)m+2,\cdots,im-1\}$. Let $\mathcal{X}$= $\{0,1,2,\cdots,lm-1\}$. Let $G_i = \{(i-1)m,(i-1)m+1,(i-1)m+2,\cdots.im-1\}, 1 \leq u \leq l$. Clearly $G_i, 1 \leq u \leq l$ partition $\mathcal{X}$ into l groups of size m. Let each column of the OA be a block. Then each block has exactly one element from each $G_i, 1 \leq u \leq l$. Any t elements from different groups $G_{i_1}, G_{i_2}, \cdots, G_{i_t}$ occur in one block, since in the rows $i_1, i_2, \cdots, i_t$, any t-tuple occurs in exactly one column in the OA. Therefore, we have a t-TD(l,m) with m^t blocks. □

The following corollary follows directly from Theorem 3.2.8 and Corollary 3.2.7.

Corollary 3.2.11 *Suppose there is an $OA(t, l, m)$, then there exists a k-traceability scheme, with m^t users, lm base keys, and each user gets l keys, where*

$$k = \lfloor \sqrt{(l-1)/(t-1)} \rfloor.$$

□

It is known that there exists an $OA(t, q+1, q)$ (Reed-Solomon Code), for all prime power q and $t < q$. Thus by the above corollary, there is a $\lfloor \sqrt{q/(t-1)} \rfloor$ -traceability scheme with q^t users, each user has $q+1$ keys, and a total of q^2+q base keys. And we have $k = \lfloor \sqrt{q/(t-1)} \rfloor$, $n = q^t$, $l = q+1$, $v = q^2+q$. Notice $l \approx tk^2$, where $t = \log_q n$. So we have $l \approx \left(\log_q n\right) k^2$, and $v \approx \left(\log_q^2 n\right) k^4$. Thus this scheme is almost as efficient as the scheme whose existence was shown in Theorem 3.2.2, as long as, given k and n we can find a prime power q and $t < q$ such that $k \approx \lfloor \sqrt{q/(t-1)} \rfloor$, and $n \approx q^t$.

The following example is an OA(2,5,4) and hence a key distribution scheme for a 2-traceability scheme with 16 users, in which each user has 5 keys and there are a total of 20 base keys.

Example 3.2.12

$$OA = \begin{pmatrix} 0 & 3 & 2 & 1 & 3 & 0 & 1 & 2 & 2 & 1 & 0 & 3 & 1 & 2 & 3 & 0 \\ 3 & 0 & 1 & 2 & 2 & 1 & 0 & 3 & 0 & 3 & 2 & 1 & 1 & 2 & 3 & 0 \\ 2 & 1 & 0 & 3 & 0 & 3 & 2 & 1 & 3 & 0 & 1 & 2 & 1 & 2 & 3 & 0 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 3 & 3 & 3 & 3 \end{pmatrix}$$

There are 5 buckets, and each bucket has 4 keys: $\{0, 1, 2, 3\}$. User 1 (column 1) gets key 0 in bucket 1, key 3 in bucket 2, and so on $\cdots$. □

The following two lemmas provide some existence result for OAs. Their explicit constructions are also known.

Lemma 3.2.13 *If q is a prime power and $t < q$, then there exists an $OA(t, q+1, q)$.* □

Lemma 3.2.14 *If q is a prime power and $t < q$, $l \leq q$, then there exists an $OA(t, l, q)$.* □

Example 3.2.15 *Let us consider a key assignment scheme such that we can trace any coalition of 4 traitors. By Lemma 3.2.11, there exists an $OA(2, 17, 16)$, $(q = 2^4)$, and by Lemma 3.2.12, there exists an $OA(2, 17, 17)$, $(q = 17)$. Using the first OA as our key assignment scheme allows us to construct a 4-traitor tracing scheme, with $16 \times 17 = 272$ base keys, $16^2 = 256$ users, and each user has 17 personal keys. Using the second OA, we obtain another 4-traitor tracing scheme, with $17^2 = 289$ bases keys, $17^2 = 289$ users, and each user has 17 personal keys. The second scheme allows 33 more users at the cost of 17 more base keys.* □

Now, let us turn our attention to the efficiency parameters. The data provider has $8k^3(k+1)\log n$ base keys. He has to perform the same number of encrypt operations of E to encrypt secret shares $s_1, s_2, \cdots, s_l$, and an additional encryption using E to encrypt the message session M using session key $s = \bigoplus_{i=1}^{l} s_i$.

A user has $4k(k+1)\log n$ encryption keys and has to perform the same number of decryption operations of E to obtain shares $s_1, s_2, \cdots, s_l$, and a decryption operation of E. Each enabling block contains the result of encrypting the secret shares using $8k^3(k+1)\log n$ keys.

Now, consider the tracing algorithm. The first nested for loops (lines 2 to 7) perform the decoding at most $2k^2l = 8k^3(k+1)\log n$ times. Assume the decoder performs the same amount of work as a normal user. The first nested for loops performs $32k^4(k+1)^2\log^2 n + 8k^3(k+1)\log n$ decryption operation of E. The second nested for loops (lines 8 -11) has complexity $O(nl) = O(nk^2\log n)$. Then the tracing algorithm has time complexity $O((k^6\log^2 n)C + nk^2\log n)$, where C is the complexity of the decryption operation of the underlying scheme E.

3.3 Two Level Open Scheme

The two level traceability scheme presented here can be thought of as iterating the one level scheme from the previous section.

Key Distribution: There are l buckets. In each of these buckets, instead of base keys, there are $b = \lceil ek \rceil$ one level key distribution schemes. We call these sub-schemes, and denote the jth scheme in the ith bucket as $S_{i,j}$. Each sub-scheme has d baskets, each basket has $4\log^2 k$ keys. We denote the yth key in the xth basket in sub-scheme $S_{i,j}$ by $k_{i,j,x,y}$. Each user gets one sub-scheme from each bucket, and as in the one level tracing scheme, each user gets one key from each basket in each of sub-scheme he is assigned to; Thus each user gets ld base keys. l and d are to be determined later. Notice that there are 2 levels here: each upper level bucket contains a set of sub-schemes in which there are a set of baskets that contain a set of base keys.

We are going to choose two sets of hash functions. The first set of l first level functions, $h_1, h_2, \cdots, h_l$, each maps $\{1, 2, \cdots, n\}$ to $\{1, 2, ..., \lceil ek \rceil\}$. They are used to assign one sub-scheme in each bucket to each user. A user $u \in U$, gets sub-schemes: $\{S_{1,h_1(u)}, S_{2,h_2(u)}, \cdots, S_{l,h_l(u)}\}$.

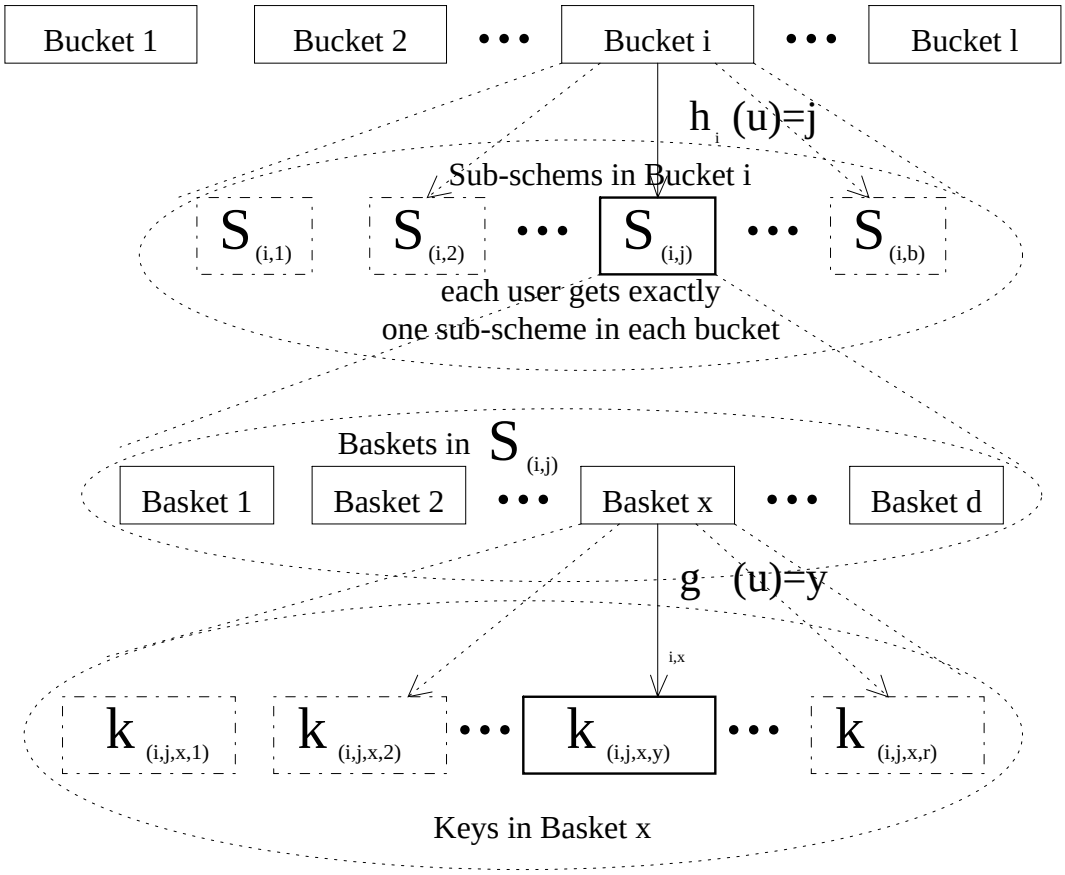


Figure 3.3: Base Keys in Two Level Open CFNP Scheme

For each bucket i, and each $j \in \{1, 2, \cdots, d\}$, there is a second level hash function $g_{i,j}$, which maps $\{1, 2, \cdots, n\}$ to $\{1, 2, \cdots, 4\log^2 k\}$. The function $g_{i,j}$ is used to map each user u to a key in the jth basket of sub-scheme $S_{i,h_i(u)}$.

Therefore the data supplier key has $l \cdot d \cdot \lceil ek \rceil \cdot 4\log^2 k$ base keys. And each user's personal key is a set of $l \cdot d$ base keys:

$$\begin{aligned} P(u) \quad = \quad & \{k_{1,h_1(u),1,g_{1,1}(u)}, k_{1,h_1(u),2,g_{1,2}(u)}, \cdots, k_{1,h_1(u),d,g_{1,d}(u)}, \\ & k_{2,h_2(u),1,g_{2,1}(u)}, k_{2,h_2(u),2,g_{2,2}(u)}, \cdots, k_{2,h_2(u),d,g_{2,d}(u)}, \\ & \vdots, \\ & k_{l,h_l(u),1,g_{l,1}(u)}, k_{l,h_l(u),2,g_{l,2}(u)}, \cdots, k_{l,h_l(u),d,g_{l,d}(u)}\}. \end{aligned}$$

Encryption: The data supplier chooses a key s in the key space of E. s gets divided into l shares, $s_1, s_2, \cdots, s_l$, such that $\bigoplus_{i=1}^{l} s_i = s$. Sub-schemes in bucket i are used to encrypt and decrypt share s_i. s_i is further divided into $d \cdot \lceil ek \rceil$ shares, $s_{i,j,c}, 1 \le j \le \lceil ek \rceil, 1 \le c \le d$, such that

$$\begin{aligned} s_i \quad = \quad & \bigoplus(s_{i,1,1}, s_{i,1,2}, \cdots, s_{i,1,d}) \\ & \bigoplus(s_{i,2,1}, s_{i,2,2}, \cdots, s_{i,2,d}) \\ & \vdots \\ & \bigoplus(s_{i,\lceil ek \rceil,1}, s_{i,\lceil ek \rceil,2}, \cdots, s_{i,\lceil ek \rceil,d}). \end{aligned}$$

$s_{i,j,c}, 1 \le j \le \lceil ek \rceil, 1 \le c \le d$, is encrypted with all keys in the cth basket of sub-scheme $S_{i,j}$. The concatenation of all these encrypted shares form the enabling block, so

$$\mathcal{B}_e = \|_{i=1}^{l}(\|_{j=1}^{\lceil ek \rceil}(\|_{x=1}^{d}(\|_{y=1}^{4\log^2 k} E_{k_{i,j,x,y}}(s_{i,j,x})))).$$

The cipher block is simply $\mathcal{B}_c = E_s(M)$.

Notice, for each i, $1 \leq i \leq l$, a user $u \in U$ has d keys:

$$k_{i,h_i(u),1,g_{i,1}(u)}, k_{i,h_i(u),2,g_{i,2}(u)}, \cdots, k_{i,h_i(u),d,g_{i,d}(u)}.$$

These keys allow u to decrypt the shares:

$$s_{i,h_i(u),1}, s_{i,h_i(u),2}, \cdots, s_{i,h_i(u),2}.$$

And hence to construct:

$$s_i = \bigoplus(s_{i,h_i(u),1}, s_{i,h_i(u),2}, \cdots, s_{i,h_i(u),2})$$

Therefore, each authorized user can reconstruct s and decrypt the cipher block to obtain the plaintext M.

Tracing: Suppose there is a pirate decoder $\mathcal{D}$. To find the key s, one needs to find all the shares among, $s_1, s_2, \cdots, s_l$. Therefore, for every bucket i there should be at least one sub-scheme $S_{i,j}$ that allows $\mathcal{D}$ to construct s_i. Suppose S_{i,c_i} is such a sub-scheme. But to construct s_i in sub-scheme S_{i,c_i}, one needs to find all values of $s_{i,c_i,1}, s_{i,c_i,2}, \cdots, s_{i,c_i,d}$. Thus $\mathcal{D}$ needs at least one key in each of d baskets in sub-scheme S_{i,c_i}. Therefore, it is valid to assume that for any i, $1 \leq i \leq l$, there exists at least one sub-scheme S_{i,c_i}, such that $\mathcal{D}$ has at least one key in each of d baskets of S_{i,c_i}.

Let $M_{0,0}^{i,j}$ be an enabling block in which the encryption with the keys of all the sub-scheme of bucket i, except $S_{i,j}$ are replaced with random data. So, $S_{i,j}$ becomes the only sub-scheme used to encrypt s_i. Let $M_{x,y}^{i,j}$ be an enabling block built from $M_{0,0}^{i,j}$ by replacing with random data the encryption with all the first y keys in the xth basket in subscheme $S_{i,j}$. This has the same effect as removing

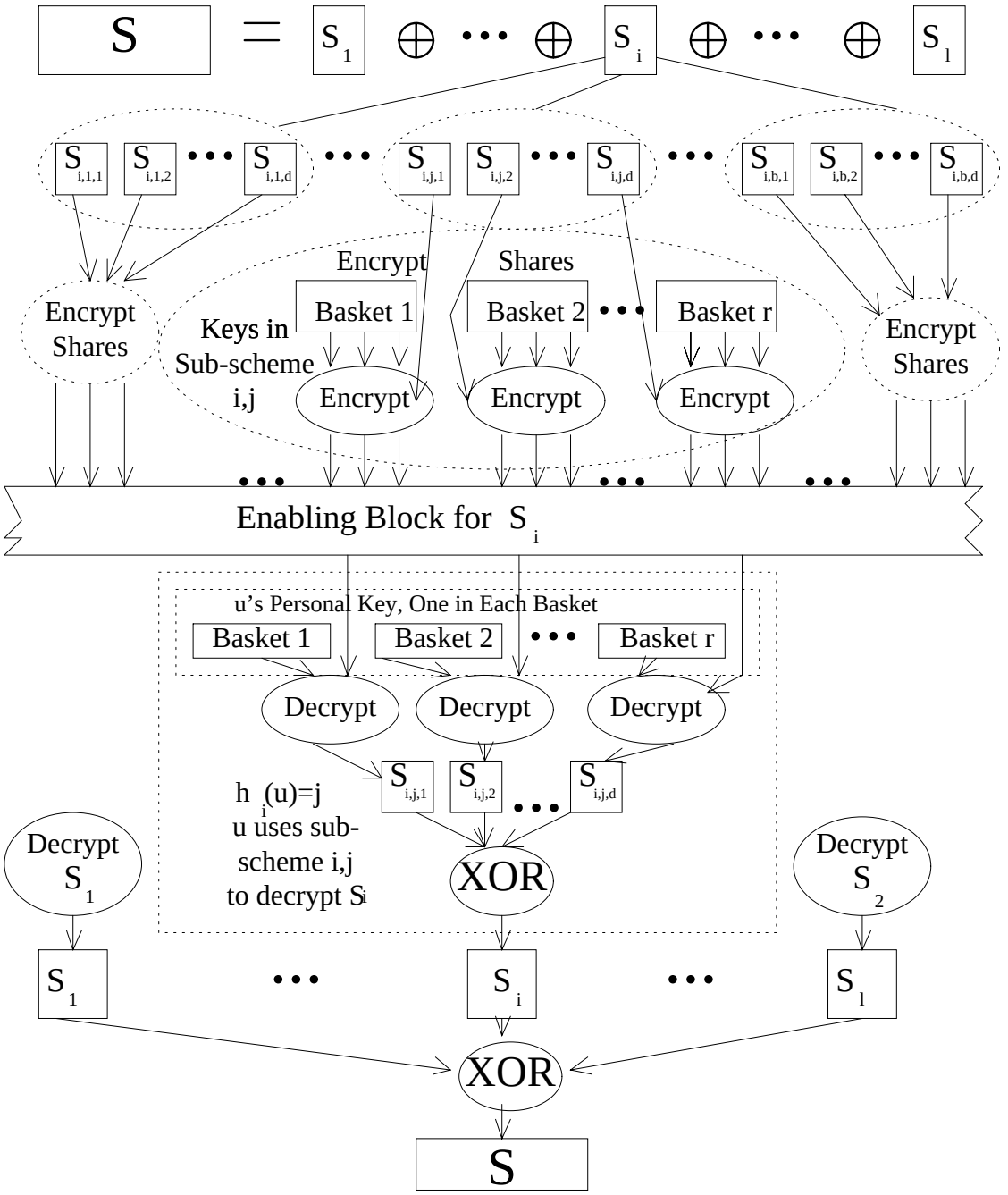


Figure 3.4: The Encryption and Decryption of a Session Key in a Two Level CFNP Open Scheme

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. for $1 \leq i \leq l$, $1 \leq j \leq \lceil ek \rceil$ do
2. if $\mathcal{D}$ decrypts experiment $E_{0,0}^{i,j}$ with non-negligible probability then
3. $c_i = j$
4. for $1 \leq x \leq d$ do
5. for $0 \leq y \leq 4 \log^2 k$ do
6. repeat experiment $E_{x,y}^{i,j}$ r times,
7. let $f_{x,y}^{i,j}$ be the fraction of times $\mathcal{D}$ decrypts $E_{x,y}^{i,j}$ correctly
8. find $t_x^{i,j}$ such that $f_{x,y}^{i,c_i} - f_{x,y}^{i,j-1} > f_{x,y}^{i,c_0}/4 \log^2 k$
9. $t_x^{i,j} = y, x++$, goto 6
10. for each $u \in U$, $1 \leq i \leq l$ do
11. for $1 \leq j \leq d$ do
12. if $h_i(u) = c_i$ and $g_{i,c_i}(u) = t_j^{i,c_i}$
13. $crt_{u,i}++$
14. $scrt_u++$, if $crt_{u,i} \geq \frac{d}{\log k}$
15. return u such that $scrt_u = MAX_{u \in U} scrt_u$

Figure 3.5: Tracing Algorithm for Two Level Open Scheme

keys $k_{i,j,x,1}, k_{i,j,x,y}, \cdots, k_{i,j,x,y}$ from the set of keys used to encrypt $s_{i,j,x}$. Let $E^{i,j}_{x,y}$ be the following experiment: prepare an encryption session, with enabling block $M^{i,j}_{x,y}$. The tracing algorithm is presented in Figure 3.5. Lines 1 to 10 determine a subset F' of the keys in $\mathcal{D}$, and lines 11 to 17 identify one traitor who has contributed his personal keys in building $\mathcal{D}$. This algorithm suffers the same problem as the tracing algorithm for one level open schemes. When the pirate decoder $\mathcal{D}$ has more than one key in one basket, it could detect the operation of the tracing algorithm, by discovering any inconsistency in the enabling block.

Notice that for each i, there is a subscheme S_{i,c_i} that enables $\mathcal{D}$ to obtain share s_i, i.e., $\mathcal{D}$ has at least one key from each basket of sub-scheme S_{i,c_i}. Then $\mathcal{D}$ must be able to decrypt $E^{i,c_i}_{0,0}$, since all the keys in subscheme S_{i,c_i} are still valid in $M^{i,c_i}_{0,0}$. Conversely, if $\mathcal{D}$ can decrypt $E^{i,c_i}_{0,0}$ successfully, $\mathcal{D}$ must have at least one key from each basket of sub-scheme S_{i,c_i}, since the keys in the rest of sub-schemes in bucket i are all invalid. Thus, $\mathcal{D}$ has at least one key in each baskets in subscheme S_{i,c_i} if and only if $\mathcal{D}$ can decrypt experiment $E^{i,c_i}_{0,0}$ successfully. Here we assume $\mathcal{D}$ has only one sub-scheme S_{i,c_i} which allows it to recover share s_i. Otherwise $\mathcal{D}$ can do a comparison of the values of s_i obtained from different sub-schemes and detect whether a tracing algorithm is running.

Lines 4 to 10 are essentially the same as the tracing algorithm in the one level scheme. They are to find one key in each basket which is in $\mathcal{D}$. Let F_i denote the keys belongs to sub-scheme S_{i,c_i} found by the algorithm in $\mathcal{D}$, i.e.,

$$F_i = \{k_{i,c_i,1,t_1^{i,c_i}}, k_{i,c_i,2,t_2^{i,c_i}}, \cdots, k_{i,c_i,d,t_d^{i,c_i}}\}.$$

And let F' be the set of all keys in $\mathcal{D}$ detected by the tracing algorithm, $F' = \cup_{i=1}^{l} F_i$. So $\mid F_i \mid = d$, and $\mid F' \mid = l \cdot d$.

We call a user a *suspect* for s_i if $crt_{u,i} \geq \frac{d}{\log k}$. So a user is a suspect for s_i if u's

personal key in bucket i has at least $\frac{d}{\log k}$ base keys in common with F_i. We declare the user who is a suspect for the largest number of s_i's as a traitor.

We would like to show that there exists a choice of hash functions, $h_i, g_{i,j}, 1 \leq i \leq l, 1 \leq j \leq d$, such that no coalition of at most k traitors can frame an innocent user.

Consider a specific user $u \in U$, and specific coalition T of size k which does not include u. We are going to bound the probability that user u will be a suspect for s_i. We first find a bound on the probability that user u will be hashed to a sub-scheme with more than $\lfloor \log k \rfloor$ traitors in a given bucket.

Lemma 3.3.1 *The probability that more than $\lfloor \log k \rfloor$ traitors are hashed by a randomly chosen hash function $h_i, 1 \leq i \leq l$ to a specific sub-scheme is at most $\left(\frac{1}{\lfloor \log k \rfloor}\right)^{\lfloor \log k \rfloor}$*

Proof: There are a total of $\lceil ek \rceil^k$ ways to hash the k traitors into $\lceil ek \rceil$ sub-schemes. The number of ways that there are at least b traitors hashed to a specific sub-scheme $S_{i,c}$ is at most:

$$\left(\begin{array}{c} k \\ b \end{array} \right) \lceil ek \rceil^{k-b},$$

where $\left(\begin{array}{c} k \\ b \end{array} \right)$ is the number of ways to choose b traitors out of a possible k traitors, and $\lceil ek \rceil^{k-b}$ is the number of ways to hash the remaining $k - b$ traitors into $\lceil ek \rceil$ sub-schemes.

Therefore the probability of more than $\lfloor \log k \rfloor$ traitors are hashed to a specific

sub-scheme is

$$\begin{aligned} p_1 &\leq \frac{\binom{k}{\lfloor \log k \rfloor} \lceil ek \rceil^{k-\lfloor \log k \rfloor}}{\lceil ek \rceil^k} \\ &= \binom{k}{\lfloor \log k \rfloor} \lceil ek \rceil^{-\lfloor \log k \rfloor} \\ &\leq \left(\frac{ek}{\lfloor \log k \rfloor} \right)^{\lfloor \log k \rfloor} (ek)^{\lfloor -\log k \rfloor} \\ &= \left(\frac{1}{\lfloor \log k \rfloor} \right)^{\lfloor \log k \rfloor}. \end{aligned}$$

□

The first level hash function h_i partitions the users into $\lceil ek \rceil$ subsets:

$$h_i^{-1}(j) = \{u \in U | h_i(u) = j\}, 1 \leq j \leq \lceil ek \rceil.$$

Let $h_i(u) = c_i$. Consider the conditional probability space where there are at most $\lfloor \log k \rfloor$ traitors in $h_i^{-1}(c_i)$. We are going to show that the probability that u is a suspect of sub-scheme S_{i,c_i} is at most $\frac{1}{16k}$.

Lemma 3.3.2 *Suppose there are at most $\lfloor \log k \rfloor$ traitors in $h_i^{-1}(c_i)$. The probability that u being marked at least $\frac{d}{\log k}$ times is at most $\frac{1}{16k}$, where $d = \left(\frac{8}{3}\right) \log^2 k$, when $k \geq 16$.*

Proof: Let $P_i(u)$ denote u's keys in sub-scheme S_{i,c_i},

$$P_i(u) = \{k_{i,c_i,1,g_{i,1}(u)}, k_{i,c_i,2,g_{i,2}(u)}, \cdots, k_{i,c_i,d,g_{i,d}(u)}\}$$

Let F_i denote the keys detected by the tracing algorithm in the sub-scheme,

$$F_i = \{k_{i,c,1,t_1^{i,c}}, k_{i,c,2,t_2^{i,c}}, \cdots, k_{i,c,d,t_d^{i,c}}\}.$$

u will not get marked for the jth basket unless there exists a $t \in T \cap h_i^{-1}(c)$ such that $g_{i,j}(u) = g_{i,j}(t) = t_j^{i,c}$.

There are $4\log^2 k$ keys in a basket. At most $\log k$ of them are assigned to some member in $T \cap h_i^{-1}(c)$ which has cardinality at most $\lfloor \log k \rfloor$. So the probability that u is marked with respect to basket j is at most $\frac{\lfloor \log k \rfloor}{4\log^2 k} \leq \frac{1}{4\log k}$. With respect to d baskets in sub-scheme S_{i,c_i}, user u is expected to be marked $\frac{d}{4\log k}$ times.

Let $X_i, 1 \leq i \leq d$ be a zero-one random variable, where

$$X_i = \begin{cases} 1 & \text{if } u \text{ is marked with respect to basket j} \\ 0 & otherwise. \end{cases} \tag{3.2}$$

$\sum_{j=1}^{d} X_j$ is the number of marks u gets. Now, apply Corollary 3.1.2, substituting $p = \frac{1}{4\log k}$, $r = d$, and $\beta = 4$, we have,

$$\begin{aligned}
Pr[\frac{1}{d}\sum_{j=1}^{d} X_j \geq 4\frac{1}{4\log k}] &\leq Pr[\frac{1}{d}\sum_{j=1}^{d} X_j \geq 4p] \\
&< (\frac{e^{4-1}}{4^4})^{\frac{d}{4\log k}} \\
\Rightarrow Pr[\sum_{j=1}^{d} X_j \geq \frac{d}{\log k}] &< (\frac{e^3}{4^4})^{\frac{d}{4\log k}} \\
&= (\frac{e^3}{2^8})^{\frac{d}{4\log k}} \\
&< (\frac{1}{2^3})^{\frac{d}{4\log k}} && e^3 < 2^5 \\
&= 2^{\frac{-3d}{4\log k}} \\
&= 2^{\frac{-8\log^2 k}{4\log k}} && \text{let } d = \left(\tfrac{8}{3}\right)\log^2 k \\
&= 2^{-2\log k} \\
&= \frac{1}{k^2} \\
&\leq \frac{1}{16k} && \text{if } k \geq 16
\end{aligned}$$

Therefore the probability that u gets more than $\frac{d}{\log k}$ is at most $\frac{1}{16k}$, when $k \geq 16$. □

Recall that a user u is a suspect with respect to s_i, if u gets marked more than $\frac{d}{\log k}$ in the subscheme $S_{i,h_i(u)}$. So, the probability that u being marked at least $\frac{d}{\log k}$ times is at most $\frac{1}{16k}$, when $k \geq 16$, if there are at most $\lfloor \log k \rfloor$ traitors in $h_i^{-1}(c_i)$. Now, we are ready to bound the probability that u is suspect to any $s_i, 1 \leq i \leq l$.

Lemma 3.3.3 *The probability that u is suspect to any $s_i, 1 \leq i \leq l$, is at most $\frac{1}{8k}$, if $k \geq 16$.*

Proof: Let A_s denote the event that u is a suspect with respect to s_i. Let A_m denote the event that u is mapped by h_i to a same sub-scheme with no more than $\lfloor \log k \rfloor$ traitors.

Suppose $h_i(u) = c$, then $Pr(\overline{A_m})$ is same as the probability of more than $\lfloor \log k \rfloor$ traitors gets mapped to sub-scheme $S_{i,c}$. By Lemma 3.3.1, we see that

$$\begin{aligned} Pr(\overline{A_m}) &\leq \left(\frac{1}{\lfloor \log k \rfloor}\right)^{\lfloor \log k \rfloor} \\ &\leq \left(\frac{1}{4}\right)^{\lfloor \log k \rfloor} && \text{since } k \geq 16 \\ &= \frac{1}{2^{(\lfloor \log k \rfloor)(2)}} \\ &\leq \frac{1}{k^2} \\ &\leq \frac{1}{16k} && \text{since } k^2 \geq 16k \text{ when } k \geq 16 \end{aligned}$$

By Lemma 3.3.2, $Pr(A_s|A_m) \leq \frac{1}{16k}$. The probability that u is a suspect with respect to s_i is:

$$Pr(A_s) = Pr(A_s, A_m) + Pr(A_s, \overline{A_m})$$

$$\begin{aligned} &= Pr(A_s|A_m)Pr(A_m) + Pr(A_s|\overline{A_m})Pr(\overline{A_m}) \\ &\leq \frac{1}{16k}Pr(A_m) + Pr(A_s|\overline{A_m})\frac{1}{16k} \\ &\leq \frac{1}{16k} + \frac{1}{16k} \\ &= \frac{1}{8k} \end{aligned}$$

□

Let us consider the probability that u is a suspect for at least $\frac{3l}{4k}$ of the buckets.

Lemma 3.3.4 *The probability that u is a suspect for at least $\frac{3l}{4k}$ of the buckets is at most $1 - 2^{\frac{-l}{k}}$.*

Proof: Let $Y_i, 1 \leq i \leq l$ be a zero-one random variable, where

$$Y_i = \begin{cases} 1 & \text{if } u \text{ is a suspect for } s_i \\ 0 & otherwise. \end{cases} \tag{3.3}$$

Now, apply Corollary 3.1.2 again, substituting $p = \frac{1}{8k}$, $r = l$, and $\beta = 6$, we have,

$$\begin{aligned} Pr[\frac{1}{l}\sum_{j=1}^{l} Y_j \geq 6\frac{1}{8k}] &< (\frac{e^{6-1}}{6^6})^{\frac{l}{8k}} \\ \Rightarrow Pr[\sum_{j=1}^{d} Y_j \geq \frac{3l}{4k}] &< (\frac{e^5}{6^6})^{\frac{d}{8k}} \\ &< (\frac{1}{2^8})^{\frac{d}{8k}} \qquad \frac{6^6}{e^5} = 314 > 2^8 \\ &= 2^{\frac{-l}{k}} \end{aligned}$$

So, the probability that u is a suspect for at least $\frac{3l}{4k}$ of the buckets is at most $2^{\frac{-l}{k}}$. □

Call a bucket *bad* if it contains a sub-scheme into which more than $\lfloor \log k \rfloor$ traitors are mapped, and *good* otherwise. Notice that in a good bucket, there are at most $\lfloor \log k \rfloor$ traitors mapped to any sub-scheme in the bucket. Let S_{i,c_i} denote the sub-scheme in bucket i, such that F' has exactly one key from each basket of S_{i,c_i}. Then at least one traitor is marked with respect to at least $\frac{d}{\lfloor \log k \rfloor} \geq \frac{d}{\log k}$ baskets in S_{i,c_i}, and therefore a suspect of s_i. So, there is at least one traitor is declared as a suspect in a good bucket. Let l' denote the number of good buckets. Next we want to show that the probability that $l' < \frac{3l}{4}$ is small.

Lemma 3.3.5 *The probability that there are at least $\frac{l}{4}$ bad buckets is at most $2^{\frac{-l}{5}}$, when $k \geq 48$.*

Proof: By Lemma 3.3.1, the probability that more than $\lfloor \log k \rfloor$ traitors are hashed to a specific sub-scheme is at most $\left(\frac{1}{\lfloor \log k \rfloor}\right)^{\lfloor \log k \rfloor}$. Thus the probability that there exists a sub-scheme into which more than $\lfloor \log k \rfloor$ traitors are hashed is at most

$$\begin{aligned}
\left(\frac{1}{\lfloor \log k \rfloor}\right)^{\lfloor \log k \rfloor} \lceil ek \rceil &\leq \frac{1}{k^2}\lceil ek \rceil && \text{see proof of Lemma 3.3.3} \\
&\leq \frac{1}{48k}\lceil e \rceil k && \text{since } k^2 \geq 48k \text{ when } k \geq 48 \\
&= \frac{1}{16}
\end{aligned}$$

Therefore, the probability that a bucket is bad is at most $\frac{1}{16}$ when $k \geq 48$.

Let $Z_i, 1 \leq i \leq l$ be a zero-one random variable, where

$$Z_i = \begin{cases} 1 & \text{if bucket } i \text{ is bad} \\ 0 & otherwise. \end{cases} \tag{3.4}$$

Now, apply Corollary 3.1.2 again, substituting $p = \frac{1}{16}$, $r = l$, and $\beta = 4$, we have,

$$\begin{aligned}
Pr[\frac{1}{l}\sum_{j=1}^{l} Z_j \geq 4\frac{1}{16}] &< (\frac{e^{4-1}}{4^4})^{\frac{l}{16}} \\
\Rightarrow Pr[\sum_{j=1}^{d} Z_j \geq \frac{l}{4}] &< (\frac{e^3}{4^4})^{\frac{d}{16}} \\
&< (\frac{1}{2^3})^{\frac{l}{16}} \qquad e^3 < 2^5 \\
&= 2^{\frac{-3.67l}{16}} \\
&< 2^{\frac{-l}{5}}
\end{aligned}$$

Thus the probability that there are at least $\frac{l}{4}$ bad buckets is at most $2^{\frac{-l}{5}}$. □

We are ready to prove the main theorem in this section.

Theorem 3.3.6 *There exists a choice of hash functions $h_i, g_{i,j}, 1 \leq i \leq l, 1 \leq j \leq d$, such that no coalition T of size at most k can frame any innocent user $u \notin T$, with $d = \left(\frac{8}{3}\right)\log^2 k$, $l > k^2 \log\left(\frac{en}{k}\right) + k\log(n+1)$, when $k \geq 48$.*

Proof: Suppose there are less than $\frac{l}{4}$ bad buckets, so at least $\frac{3l}{4}$ good buckets. By Lemma 3.3.5 this happens with probability of at least $1 - 2^{\frac{-l}{5}}$. Recall that at least one traitor who is declared a suspect in a good bucket. Since there are at least $\frac{3l}{4}$ good buckets, and there are k traitors, there exists at least one traitor is suspect for at least $\frac{3l}{4k}$ s_i's. Therefore the probability that u is mistakenly identified as a traitor is less than the probability that u is a suspect for at least $\frac{3l}{4k}$ of the s_i's, which is $2^{\frac{-l}{k}}$ by Lemma 2.4.

Therefore the probability that for one of $\begin{pmatrix} n \\ k \end{pmatrix}$ possible coalition of size k, and given that there are at least $\frac{3l}{4}$ good buckets, some innocent user is mistakenly

identified as a traitor is at most $n \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{\frac{-l}{k}}$. And, the probability that for some coalition there are less than $\frac{3l}{4}$ good buckets, is at most $\begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{\frac{-l}{5}}$.

Thus the probability of any innocent user is mistakenly identified as a traitor is at most,

$$p_e = n \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{\frac{-l}{k}} + \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{\frac{-l}{5}} \quad \leq \quad (n+1) \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{\frac{-l}{k}}$$

with $k \geq 5$.

Choose $l > k^2 \log\left(\frac{en}{k}\right) + k \log(n+1)$, then

$$\begin{aligned} p_e &< (n+1) \cdot \begin{pmatrix} n \\ k \end{pmatrix} \cdot 2^{-(k \log\left(\frac{en}{k}\right) + \log(n+1))} \\ &\leq (n+1) \cdot \left(\frac{en}{k}\right)^k \cdot \left(\frac{en}{k}\right)^{-k} \cdot (n+1)^{-1} \\ &= 1 \end{aligned}$$

Therefore, there exists a choice of hash functions $h_i, g_{i,j}, 1 \leq i \leq l, 1 \leq j \leq d$, such that an innocent user is never mistakenly identified as a traitor. □

Now, let us consider the Iefficiency parameters. In the above scheme, there are $l = k^2 \log\left(\frac{en}{k}\right) + k \log(n+1)$ buckets, each bucket contains $b = \lceil ek \rceil$ sub-schemes. And, there are $d = \frac{3}{8} \log^2 k$ baskets in each sub-scheme, each basket contains $4 \log^2 k$ keys. Each enabling block consists of the result of encrypting the secret shares using the same number of keys.

The data provider has

$$\begin{aligned} l \cdot d \cdot \lceil ek \rceil \cdot 4 \log^2 k &= \frac{42}{3} \lceil ek \rceil \log^4 k \left(k^2 \log\left(\frac{en}{k}\right) + k \log(n+1)\right) \\ &= O\left(k^3 \log^4 k \log\left(\frac{n}{k}\right)\right) \end{aligned}$$

keys. The same number of encryptions are needed to encrypt secret shares: $s_{i,j,c}$, $1 \leq i \leq l$, $1 \leq j \leq b$, $1 \leq c \leq d$. And, an additional encrypting operation of E is required to encrypt the message M.

A user has

$$\begin{aligned} l \cdot d &= \frac{8}{3} \log^2 k \left(k^2 \log \left(\frac{en}{k} \right) + k \log(n+1) \right) \\ &= O\left(k^2 \log^2 k \log \left(\frac{n}{k} \right) \right) \end{aligned}$$

encryption keys and has to perform the same number of decryption operations of E to obtain sufficient shares in order to obtain the session key s. One additional decryption operation of E is required to obtain the content.

Now, consider the tracing algorithm. The first part (lines 1 to 10) of the algorithm consists of four nested loops and performs up to

$$\begin{aligned} l\lceil ek \rceil d(4 \log^2 k) &= \frac{42}{3} \lceil ek \rceil \log^4 k \left(k^2 \log \left(\frac{en}{k} \right) + k \log(n+1) \right) \\ &= O\left(k^3 \log^4 k \log \left(\frac{n}{k} \right) \right) \end{aligned}$$

decryptions using decoder $\mathcal{D}$. Assume the decoder performs the same amount of work as a normal user. Then there are up to

$$O\left(k^5 \log^6 k \log^2 \left(\frac{n}{k} \right) + k^3 \log^4 k \log \left(\frac{n}{k} \right) \right)$$

decryption operations of E. The second part (lines 11 to 17) has complexity

$$O(nld) = O\left(nk^2 \log^2 k \log \left(\frac{n}{k} \right) \right).$$

Then the tracing algorithm has time complexity

$$O\left(\left(k^5 \log^6 k \log^2 \left(\frac{n}{k} \right) \right) C + nk^2 \log^2 k \log \left(\frac{n}{k} \right) \right)$$

where C is the complexity of the decryption operation of the underlying scheme E.

3.4 One Level Secret Scheme

Secret schemes can be made more efficient than open schemes since the traitors do not have the knowledge of which keys other user received. In a secret scheme, even if the set of keys of the coalition of traitors includes a large number of the personal keys of an innocent user, the traitors do not know which keys the user has and hence cannot put together a pirate decoder that frames a specific user.

The one level secret scheme is very similar to the one level open scheme, except that there are $4k$ keys in each bucket instead of $2k^2$ in the open scheme. The l hash functions, $h_i, 1 \leq i \leq l$, map $\{1, 2, \cdots, n\}$ to $\{1, 2, \cdots, 4k\}$. The encryption scheme and tracing scheme is same as in the one level open scheme.

Lemma 3.4.1 *There exists a choice of $h_i, 1 \leq i \leq l$, such that the probability that there exists a coalition T of size k traitors that can frame an innocent user $u \in U \setminus T$ is at most p, for any $p \in [0, 1]$.*

Proof: Let $h_i, 1 \leq i \leq l$ be a set of l randomly chosen hash functions that each maps $\{1, 2, \cdots, n\}$ to $\{1, 2, \cdots, 4k\}$. Let u be any user. Let A_u denote the event that u can be framed by a coalition T, $u \notin T, \mid T \mid = k$.

Recall that the tracing algorithm is going to identify l keys, one from each bucket, in a pirate decoder $\mathcal{D}$. Let F' denote the set of keys detected by the tracing algorithm. Then $F' = \{k_{1,c_1}, k_{2,c_2}, \cdots, k_{l,c_l}\}$. And, there is at least one traitor t with $crt_t \geq l/k$. Let A'_u denote the event that u gets marked at least $\frac{l}{k}$ times. Clearly $Pr(A_u) \leq Pr(A'_u)$.

As the hash functions are randomly chosen, the value $h_i(u)$ is uniformly distributed in $\{1, 2, \cdots, 4k\}$. Let T be any coalition of size k. Since T has no knowledge on $h_i(u)$, it has to randomly choose a key $k_{i,c_i} \in F$ to be exposed by the

tracing algorithm. The probability of $h_i(u) = c_i$ is $\frac{1}{4k}$. Notice this probability does not depend on which coalition builts $\mathcal{D}$. So, the probability of u being marked with respect to bucket i is $\frac{1}{4k}$.

Let X_i be a zero-one random variable, where

$$X_i = \begin{cases} 1 & \text{if } u \text{ is marked with respect to bucket } i \\ 0 & otherwise. \end{cases} \tag{3.5}$$

Then $\sum_{j=1}^{l} X_j$ is number of marks u will get.

Apply Corollary 3.1.2, substituting $p = \frac{1}{4k}$, $r = l$, $\beta = 4$,

$$\begin{aligned} Pr[\frac{1}{l}\sum_{j=1}^{l} X_j \geq 4\frac{1}{4k}] &< (\frac{e^{4-1}}{4^4})^{\frac{l}{4k}} \\ \Rightarrow Pr[\sum_{j=1}^{l} X_j \geq \frac{l}{k}] &< (\frac{e^3}{4^4})^{\frac{l}{4k}} \\ &< (\frac{1}{2^3})^{\frac{l}{4k}} \qquad e^3 < 2^5 \\ &= 2^{\frac{-3l}{4k}} \end{aligned}$$

So, the probability that u getting marked at least $\frac{k}{l}$ times is $Pr(A'_u) < 2^{\frac{-3l}{4k}}$. So $Pr(A_u) < 2^{\frac{-3l}{4k}}$. Then the probability that there exists an innocent user being framed by any coalition is $Pr(A) = \sum_{u \in U} Pr(A_u) = n \cdot Pr(A_u) < n \cdot 2^{\frac{-3l}{4k}}$. If we choose $l \geq \frac{4}{3}k\log(n/p)$, then

$$\begin{aligned} Pr(A) &< n \cdot 2^{-\log(n/p)} \\ &= n \cdot \frac{p}{n} \\ &= p \end{aligned}$$

□

Theorem 3.4.2 *There exists a secret (k, p)-resilient scheme, for any $p \in [0, 1]$.*

Proof: By Lemma(3.4.1), the probability of a innocent user being framed is at most p, when $l \geq \frac{4}{3}k\log(n/p)$. Therefore, if we choose hash functions h_i, $1 \leq i \leq l$ randomly, the probability of an innocent user is mistakenly identified as traitor is less than p. Hence we have a fully (p,k)-resilient secret traitor tracing scheme, with $l \geq \frac{4}{3}k\log(n/p)$.

If we choose $l \geq \frac{4}{3}k\log(n)$, then $Pr(A) < 1$. This tells us that there exists a choice of $h_i, 1 \leq i \leq l$ such that no user will be framed by any coalition. That is, there exists a fully k-resilient scheme with $l > \frac{4}{3}k\log(n)$. □

In the above one level fully (p,k)-resilient secret scheme, the data supplier has $\frac{4}{3}k\log(n/p)\cdot 4k = \frac{16}{3}k^2\log(n/p) = O(k^2\log(n/p))$ base keys. He has to perform the same number of encrypting operations to encrypt secret shares $s_1, s_2, \cdots, s_l$, and an additional encrypting operation of E to encrypt the message session M using session key $s = \bigoplus_{i=1}^{l} s_i$.

A user has $\frac{4}{3}k\log(n/p) = O(k\log(n/p))$ encryption keys and has to perform the same number of decryption operations to obtain shares $s_1, s_2, \cdots, s_l$.

Now, consider the tracing algorithm in Figure 3.2. The first nested for loop performs the decryption at most $4kl = \frac{16}{3}k^3\log(n/p)$ times. Assume the decoder performs the same amount of decryptions as a normal user. The first nested for loop performs $\frac{16}{3}k^3\log(n/p)\cdot\left(\frac{4}{3}k\log(n/p)+1\right) = \frac{64}{9}k^4\log(n/p) + \frac{16}{3}k^3\log(n/p)$ decryption operation of E. The second nested for loops has complexity $nl = \frac{4}{3}nk\log(n/p))$. Then the tracing algorithm has time complexity $O(k^4(\log^2(n/p))C + nk\log(n/p))$, where C is the complexity of the decryption operation of the underlying encryption scheme E.

3.5 Two Level Secret Scheme

Here, we are going to present a two-level (p,k)-resilient secret scheme which improves the performance of the one level (p,k)-resilient secret scheme. The difference between this scheme and the open two level scheme is that here it is enough to use only one mapping at the first level.

The scheme uses one random hash function $h : \{1,2,\cdots,n\} \mapsto \{1,2,\cdots,\frac{2ek}{b'}\}$, where $b' = b - \frac{b}{b-1}\ln\left(\frac{ek}{b}\right)$.

Lemma 3.5.1 *For any fixed coalition of k traitors, the probability that b or more traitors are mapped to the same element in $\{1,2,\cdots,\frac{2ek}{b'}\}$ by h is at most $p/2$, if $b = \log(4/p)$.*

Proof: By Lemma 3.3.1, the probability that b or more traitors are mapped to a specific element is $\left(\begin{array}{c} k \\ b \end{array}\right)\left(\frac{2ek}{b'}\right)^{-b}$. Then the probability that there exists an element that b or more traitors are mapped to is

$$\begin{aligned}
\left(\begin{array}{c} k \\ b \end{array}\right)\left(\frac{2ek}{b'}\right)^{-b}\frac{2ek}{b'} &= \left(\begin{array}{c} k \\ b \end{array}\right)\left(\frac{b'}{2ek}\right)^{b-1} \\
&\leq \left(\frac{ek}{b}\right)^{b}\left(\frac{b - \frac{b}{b-1}\ln\left(\frac{ek}{b}\right)}{2ek}\right)^{b-1} \\
&= \frac{ek}{b2^{b-1}}\left(1 - \frac{\ln\left(\frac{ek}{b}\right)}{b-1}\right)^{b-1} \\
&\approx \frac{ek}{b2^{b-1}}e^{-\ln\left(\frac{ek}{b}\right)} \\
&= \frac{1}{2^{(b-1)}} \\
&= \frac{2}{2^{\log(4/p)}} \\
&= p/2
\end{aligned}$$

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. for $1 \leq i \leq r$ do
2. randomly choose a permutation λ on $\{1, 2, \cdots, \frac{2\epsilon k}{b'}\}$.
3. for $1 \leq l \leq \frac{2\epsilon k}{b'}$ do
4. use $\mathcal{D}$ on experiment $E_{\lambda(j)}$
5. let E_c be an experiment which $\mathcal{D}$ decrypts with nonnegligible probability.
6. perform one level tracing algorithm on scheme S_c

Figure 3.6: Tracing Algorithm for Two Level Secret Scheme

□

We are going to construct a secret one level $(p/2, b)$-resilient scheme for each set $h^{-1}(i) = \{u \in U | h(u) = i\}, 1 \leq i \leq \frac{2\epsilon k}{b'}$. Let $S_1, S_2, \cdots, S_{\frac{2\epsilon k}{b'}}$ denote these schemes. Each user u receives his personal key in subscheme $S_{h(u)}$.

Given a message session M, a session key s is randomly chosen. Let $\mathcal{B}_e^i, 1 \leq i \leq \frac{2\epsilon k}{b'}$ denote the enabling block in sub-scheme i with respect to session key s. The concatenation of enabling blocks of all sub-schemes form the enabling block of the two level scheme, $\mathcal{B}_e = \|_{i=1}^{\frac{2\epsilon k}{b'}} \mathcal{B}_e^i$. The cipher block is $\mathcal{B}_c = E_s(M)$.

Any user u has keys from sub-scheme $S_{h(u)}$ and thus can reconstruct s from $\mathcal{B}_e^{h(u)}$. For a decoder $\mathcal{D}$ to decrypt M successfully, $\mathcal{D}$ must have a valid personal key from a sub-scheme, i.e., there must be a sub-scheme S_c such that $\mathcal{D}$ contains at least one key from each bucket in S_c.

Let M_i be an enabling block in which the encryption with the keys of all the sub-schemes except S_i are replaced with random data. Let E_i be the experiment: prepare an encryption session, with enabling block M_i. Figure 3.6 presents the tracing algorithm. It is assumed that there exists a subscheme S_c such that $\mathcal{D}$ contains at least one key from each bucket in S_c. Then $\mathcal{D}$ should encrypt E_c successfully. Clearly, the algorithm will work if $\mathcal{D}$ always use one subscheme in decoding. Since each user is assigned to one subscheme, $\mathcal{D}$ could have access to as many as k subschemes. And $\mathcal{D}$ could choose one subscheme out of a set of S' of $k' \leq k$ available subschemes to use for a session. Note, $\mathcal{D}$ can also detect if a tracing algorithm is running by using several subschemes to decode the content and checking if the result is the same. But this involves more work than decrypting two shares.

Let $S' = \{S_1, S_2, \cdots, S_{k'}\}$, and let Pr_i denote the probability that S_i is chosen by $\mathcal{D}$. Clearly, $\sum_{i=1}^{k'} Pr_i = 1$. For a particular subscheme $S_i \in S'$, the probability that it is not chosen in experiment E_i is $1 - Pr_i$. Then the probability that there does not exist one scheme $S_i \in S'$ that is chosen for experiment E_i, $1 \leq i \leq k'$ is

$$\prod_{i=1}^{k'}(1 - Pr_i) \leq (1 - \frac{1}{k'})^{k'} = \left((1 - \frac{1}{k'})^{-k'}\right)^{-1} \leq e^{-1}$$

If we repeat each experiment r times, the probability is at most e^{-r}. Table 3.1 gives the values of e^{-r} for some r between 1 and 20. If $r = 5$, the probability is at most 0.00674; if $r = 10$, probability is at most 0.0000454. We can choose r so that the algorithm can identify a subscheme used by $\mathcal{D}$ with a high probability. Once such an S_c is found, the one level tracing algorithm can be used to identify a traitor.

Theorem 3.5.2 *The scheme described above is a (p, k)-resilient secret scheme, for any $p \in [0, 1]$.*

r	e^{-r}
1	.3678794412
5	.006737947002
8	.0003354626281
10	.00004539992980
12	$.6144212359 \times 10^5$
15	$.3059023209 \times 10^6$
17	$.4139937724 \times 10^7$
20	$.2061153626 \times 10^8$

Table 3.1: Values of e^{-r}

Proof: We would like to show that the probability of an innocent user being identified as a traitor is less than p. Let A_m denote the event that there is no subscheme into which at least b traitors are mapped together, where $b = \log(4/p)$. By Lemma 3.4.1 $Pr(\overline{A_m}) \leq (p/2)$. Let A_f denote the event that an innocent user is mistakenly identified as a traitor. Since each subscheme is $(p/2, k)$-resilient, the probability of event A_f is at most $p/2$, if event A_m occurs. So, $Pr(A_f|A_m) \leq p/2$. Now we have

$$\begin{aligned} Pr(A_f) &= Pr(A_f, A_m) + Pr(A_f, \overline{A_m}) \\ &= Pr(A_f|A_m)Pr(A_m) + Pr(A_f|\overline{A_m})Pr(\overline{A_m}) \\ &\leq \frac{p}{2}Pr(A_m) + Pr(A_f|\overline{A_m})\frac{p}{2} \\ &\leq \frac{p}{2} + \frac{p}{2} \\ &= p \end{aligned}$$

Therefore we have a (p, k)-resilient scheme. □

In the two level (p,k)-resilient secret scheme, the data supplier has all the keys in the $\frac{2ek}{b'}$ one level $(p/2,b)$-resilient schemes. Each of the one level schemes has $\frac{16}{3}b^2\log\frac{2n}{p}$ keys. Thus, the total number of base keys in the two level scheme is

$$\begin{aligned}
\frac{16}{3}b^2\left(\log\frac{2n}{p}\right)\frac{2ek}{b'} &= \frac{16}{3}b^2\left(\log\frac{2n}{p}\right)\frac{2ek}{b-\frac{b}{b-1}\ln\left(\frac{ek}{b}\right)} \\
&= \frac{32e}{3}kb\left(\log\frac{2n}{p}\right)\frac{1}{1-\frac{1}{b-1}\ln\left(\frac{ek}{b}\right)} \\
&= \frac{32e}{3}kb\left(\log\frac{2n}{p}\right)\frac{b-1}{b-1-\ln\left(\frac{ek}{b}\right)} \\
&= \frac{32e}{3}kb\left(\log\frac{2n}{p}\right)\left(1+\frac{\ln\left(\frac{ek}{b}\right)}{b-1-\ln\left(\frac{ek}{b}\right)}\right)
\end{aligned}$$

The same number of encryption operations are needed to encrypt secret shares, plus one additional encryption operation has to be performed to encrypt the actual content.

Each user u gets the personal key in the one level scheme $S_{h(u)}$. So each user gets $\frac{4}{3}b\log(2n/p)) = O(\log(n/p)\log(1/p))$ base keys, and performs the same amount of decryption operations to decrypt enough secret shares, plus one more to decrypt the content.

The size of the enabling block is the sum of the sizes of the enabling blocks in each of the one level scheme. Each one level scheme has enabling block size $\frac{16}{3}b^2\log(2n/p))$. So the size of the enabling block in the two level scheme is $\frac{16}{3}(b^2\log(n/p)\cdot\frac{2ek}{b'}) = \frac{32e}{3}kb\left(\log\frac{2n}{p}\right)\left(1+\frac{\ln\left(\frac{ek}{b}\right)}{b-1-\ln\left(\frac{ek}{b}\right)}\right)$ which is same as the number of encryption operations the data supplier has to perform to encrypt secret shares of the session key.

The first part (lines 1 to 5) of the tracing algorithm performs $r\frac{2ek}{b'}$ decodings using the decoder $\mathcal{D}$, where $r \leq 20$. Assume $\mathcal{D}$ performs the same amount of work

as a normal user. Then there are

$$\frac{4}{3}rb\frac{2ek}{b'}\log(2n/p)) = \frac{4ekr}{3}\left(\log\frac{2n}{p}\right)\left(1+\frac{\ln\left(\frac{ek}{b}\right)}{b-1-\ln\left(\frac{ek}{b}\right)}\right)+r\frac{2ek}{b'}$$

decryption operation of E. The one level tracing algorithm has complexity

$$O(k^4(\log^2(n/p))C + nk\log(n/p))$$

.

The following theorem states that if k is smaller than $\frac{b}{2p}$, then the total number of keys is less than $\frac{32e}{3}kb(b+1)\left(\log\frac{2n}{p}\right)$.

Theorem 3.5.3 *If* $k \leq \frac{1}{2p}\log\frac{4}{p}$, *then* $\frac{\ln\left(\frac{ek}{b}\right)}{b-1-\ln\left(\frac{ek}{b}\right)} \leq b$.

Proof: Notice that

$$2^{b-3} = 2^{\log(4/p)-3} = \frac{4/p}{8} = \frac{1}{2p}.$$

So, we have

$$\begin{aligned} k &\leq \frac{1}{2p}\log\frac{4}{p} \\ &= 2^{b-3}\cdot b \\ &\leq e^{b-3}\cdot b. \end{aligned}$$

Then,

$$\begin{aligned} k \leq e^{b-3}b &\Rightarrow \frac{k}{b} \leq e^{b-3} \\ &\Rightarrow \left(\frac{k}{b}\right)^{b+1} \leq e^{b^2-2b-3} \end{aligned}$$

$$
\begin{aligned}
&\Rightarrow \quad \left(\frac{ek}{b}\right)^{b+1} \leq e^{b^2-b-2} \\
&\Rightarrow \quad \left(\frac{ek}{b}\right)^{b+1} \leq e^{b^2-b} \\
&\Rightarrow \quad (b+1)\ln\frac{ek}{b} \leq b^2-b \\
&\Rightarrow \quad \ln\frac{ek}{b} \leq b(b-1) - b\ln\frac{ek}{b} \\
&\Rightarrow \quad \frac{\ln\left(\frac{ek}{b}\right)}{b-1-\ln\left(\frac{ek}{b}\right)} \leq b.
\end{aligned}
$$

□

Thus, if $k \leq \frac{1}{2p}\log\frac{4}{p}$, the total number of keys is less than

$$\frac{32e}{3}kb(b+1)\left(\log\frac{2n}{p}\right) = O\left(k\log^2\frac{1}{p}\log\frac{n}{p}\right).$$

And the tracing algorithm has complexity of $O((k\log\frac{n}{p}\log\frac{1}{p})C)$ plus the complexity of the one level tracing algorithm.

3.6 One Level Threshold Scheme

In fully resilient tracing schemes, a decoder $\mathcal{D}$ either can decrypt all the sessions, or cannot decrypt any session at all. For example, in one level schemes, if $\mathcal{D}$ has one key from each bucket, $\mathcal{D}$ can decrypt all sessions successfully; Otherwise $\mathcal{D}$ cannot decrypt any session. And the tracing algorithm can trace one source of the keys in $\mathcal{D}$ if and only if $\mathcal{D}$ can decrypt all the sessions.

In the threshold schemes introduced in this and the next section, we are going to allow $\mathcal{D}$ to decrypt certain subset of but not necessarily all sessions. And the tracing algorithm can trace a source of the keys in $\mathcal{D}$, only if $\mathcal{D}$ can decrypt at least

certain percentage of the sessions. These schemes are useful in those applications where a decoder which does not decode close to 100 percent of message successfully is not very useful. For example, no one will probably buy a pirate TV decoder which can decode only 90 percent of the pictures. For these applications, the fully resilient schemes might be an overkill if we can construct threshold schemes that are more efficient. We are going to consider schemes that trace the source of the keys in a decoder only if the decoder can decrypt with a high success rate, but does not necessarily perform well if the decoder has a low success rate.

In this section we are going to present a one level q-threshold tracing scheme: a scheme traces the source of keys in any decoder which decrypts correctly with a probability at least q. The benefit of using such a threshold scheme is a reduction in the size of enabling blocks, and hence a reduction in redundancy overhead. And user will perform less decryption operations. In fact, the number of share needs to be decrypted could be as few as 1.

The one level threshold scheme has the same key distribution scheme as a one level fully resilient secret scheme. However in the encryption of the message, the session key s is divided into $t \leq l$ shares, s_1, s_2, $\cdots$, s_t, such that $s = \bigoplus_{i=1}^{t} s_i$, instead of l shares in one level secret scheme. t buckets $B' = \{B_{a_1}, B_{a_2}, \cdots, B_{a_t}\}$ are chosen uniformly at random from the l available buckets. s_i is encrypted with all the keys in B_{a_i} using E. And the enabling block also includes the indices of the t chosen buckets. $\mathcal{B}_e = (\|_{i=1}^{t} a_i) \| \left(\|_{i=1}^{t} \left(\|_{j=1}^{4k} E_{k_{a_i,j}}(s_i)\right)\right)$. Any authorized user has one key from each bucket and can always decrypt all $s_i, 1 \leq i \leq t$, and hence reconstruct s. And any decoder that can reconstruct s must have a key in all of the buckets in B'.

The threshold scheme has another parameter w, $0 < w < 1$, such that the tracing algorithm is able to trace the source of the keys in a pirate decoder $\mathcal{D}$ if $\mathcal{D}$

has at least one key from at least wl buckets. t is chosen such that if $\mathcal{D}$ decrypts with probability at least q, then $\mathcal{D}$ must have one key from at least wl buckets.

Let $\overline{B}$ denote the set of buckets in which $\mathcal{D}$ has at least one key. $\mid \overline{B} \mid = d$. Then the probability that a randomly chosen bucket is in $\overline{B}$ is $\frac{d}{l}$. The probability that all t randomly chosen buckets are in $\overline{B}$ is at most $\left(\frac{d}{l}\right)^t$. Given any message session, $\mathcal{D}$ is able to decrypt it if and only if $B' \subseteq \overline{B}$. Since B' are randomly chosen, the probability that $\mathcal{D}$ is able to decrypt the message is at most $\left(\frac{d}{l}\right)^t$. If $\mathcal{D}$ decrypts with success rate at least q, then $\left(\frac{d}{l}\right)^t \geq q$. We can set $t = \log_w q$. Then $q = w^t \leq \left(\frac{d}{l}\right)^t$, which implies $w \leq \frac{d}{l}$, $d \geq wl$, i.e., $\mathcal{D}$ has one key in at least wl buckets. To have $t = 1$, we can set $w = q$.

The tracing algorithm is presented in Figure 3.6. Let $\alpha = \{k_1, k_2, \cdots, k_{4kl}\}$ denote the set of all base keys. Lines 1 to 9 identify at least lw keys in $\mathcal{D}$ where all of these lw keys belong to distinct buckets. Let E_i denote the experiment of using M_i as enabling block in a message block. Since M_0 is a valid enabling block, $\mathcal{D}$ should decrypt E_0 with probability at least q. We built M_i from M_{i-1} by replacing the data encrypted with key $k_{\lambda(i)}$ by random data. So, the knowledge of $k_{\lambda(i)}$ will not help in decrypting E_i. It has the same affect as removing the key $k_{\lambda(i)}$ from its bucket. Now if $\mathcal{D}$ uses this key in decoding, $\mathcal{D}$ will in fact have only $lw - 1$ keys and hence decrypt with a probability less than q. Here again, we have to assume $\mathcal{D}$ has at most one key from each bucket. Otherwise the algorithm will not work and $\mathcal{D}$ can easily detect whether it is being inquired by a tracing algorithm.

So in E_{4kl}, we have removed all the keys. $\mathcal{D}$ would decrypt it with a negligible probability much less than q. Thus there exists a c such that $\mathcal{D}$ decrypts E_{c-1} with probability at least q but decrypts E_c with probability less than q. Thus $k_{\lambda(c)}$ must be in $\mathcal{D}$, since removing $k_{\lambda(c)}$ results in $\mathcal{D}$ having keys from less than lw buckets. And suppose $k_{\lambda(c)}$ is from bucket B_d, then $k_{\lambda(c)}$ must be the last key $\mathcal{D}$ has in B_d.

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. set $F' = \emptyset$ and build a valid enabling block M_0
2. randomly choose a permutation λ of $\{1, 2, \cdots, 4kl\}$
3. for $1 \leq i \leq 4kl$ do
4. build M_i from M_{i-1} by replacing the data encrypted with key $k_{\lambda(i)}$ by random data.
5. if $\mathcal{D}$ decrypts message block with enabling block M_i with probability less than q
6. $F' = F' \cup \{\lambda(i)\}$
7. if no key in F' is from the same bucket as $\lambda(i)$
8. $$q = \left(\begin{array}{c} l \\ lw - 1 \end{array} \right) \Big/ \left(\begin{array}{c} l \\ t \end{array} \right), \; w = q^{1/t}$$
9. Choose $F' \subseteq F'$, such that F' has exactly one key from wl buckets.
10. for each $u \in U$ do
11. for each $c \in F'$ do
12. $crt_u + +$ if $h_i(u) = c$
13. return u such that $crt_u = MAX_{u \in U} crt_u$

Figure 3.7: Tracing Algorithm for One Level Threshold Scheme

After removing $k_{\lambda(c)}$, $\mathcal{D}$ still has a key in at least $lw-1$ buckets, and so it can still decrypt with the probability of $\begin{pmatrix} l \\ lw-1 \end{pmatrix} \Big/ \begin{pmatrix} l \\ t \end{pmatrix}$. So, we modify q and w accordingly and keep on. There should exist another c' such that $\mathcal{D}$ decrypts $E_{c'}$ with probability less than q (modified), but decrypts $E_{c'-1}$ with probability at least q. So $k_{\lambda(c')}$ must also be in $\mathcal{D}$. And we update q and w accordingly and go on until the end of loop. At the end of the loop, F' should have at least wl keys from distinct buckets. Then each user is marked once for each of his personal key in F', and the user with the most number of marks is claimed as a traitor.

Figure 3.8 gives a simpler randomized algorithm. This algorithm also assumes $\mathcal{D}$ has at most one key from each bucket. In each iteration of the repeat loop (lines 2 - 8), one key is exposed. The repeat loop terminates when wl distinct keys are exposed. Suppose D has q keys. Assume one key is randomly exposed at each iteration. So, each of the q keys has a probability of $\frac{1}{q}$ of being exposed at an iteration. Then, after r iterations, the probability of one specific key is not exposed is $(1-\frac{1}{q})^r$. So the probability of there exists a key not exposed is $p \leq q(1-\frac{1}{q})^r$. Choose $r = 20q$, we have $p \leq q(1-\frac{1}{q})^{20q} \leq qe^{-20}$ which is negligibly small. Thus the randomized algorithm is expected to terminate within $20lw$ iterations.

Notice there are at most k traitors, and there are wl keys in F'. So there exists a traitor t with $crt_t \geq \frac{wl}{k}$. We are going to show that there exists a choice of $h_i, 1 \leq i \leq l$, such that the probability of an innocent user u gets at least $\frac{wl}{k}$ marks is small.

Lemma 3.6.1 *There exists a choice of $h_i, 1 \leq i \leq l$, such that the probability that there exists a coalition T of size k can frame an innocent user less than p, for any $p \in [0,1]$.*

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. $F' = \emptyset$
2. repeat
3. build a valid enabling block M_0
4. randomly choose a permutation λ of $\{1, 2, \cdots, 4kl\}$
5. for $1 \leq i \leq 4kl$ do
6. build M_i from M_{i-1} by replacing the data encrypted with key $k_{\lambda(i)}$ by random data.
7. if $\mathcal{D}$ decrypts message block with enabling block M_i with probability less than q
8. $F' = F' \cup \{\lambda(i)\}$
9. until $\mid F' \mid =$ wl
10. for each $u \in U$ do
11. for each $c \in F'$ do
12. $crt_u + +$ if $h_i(u) = c$
13. return u such that $crt_u = MAX_{u \in U} crt_u$

Figure 3.8: Randomized Tracing Algorithm for One Level Threshold Scheme

Proof: Let u be any user. Let A_u denote the event that there is a coalition T can frame u, $\mid T \mid = k, u \in U \setminus T$. Let A'_u denote the event that $crt_u \geq \frac{wl}{k}$. Clearly, $Pr(A_u) \leq Pr(A_u)$, since there exists a traitor t with $crt_t \geq \frac{wl}{k}$. Let $\mathcal{D}$ denote a decoder built by T that decrypts with success rate at least q.

As the hash functions are randomly chosen, the value $h_i(u)$ is uniformly distributed in $\{1, 2, \cdots, 4k\}$. We want to determine the probability that $\mathcal{D}$ exposing $k_{i,h_i(u)}$ as its key. Notice the algorithm exposes wl keys in $\mathcal{D}$. The probability that one of the keys is from bucket i is w. Suppose $\mathcal{D}$ chooses to expose a key from bucket i. Since T has no knowledge on $h_i(u)$, it has to randomly choose a key $k_{i,c_i} \in F$ to be exposed by the tracing algorithm. The probability that $h_i(u) = c_i$ is $\frac{1}{4k}$. The probability that T exposes the key $k_{i,h_i(u)}$ is $\frac{w}{4k}$. Notice that this probability does not depend on which coalition builts $\mathcal{D}$. So, the probability of u being marked with respect to bucket i is $\frac{w}{4k}$.

Let X_i be a zero-one random variable, where

$$X_i = \begin{cases} 1 & \text{if } u \text{ is marked with respect to bucket } i \\ 0 & otherwise. \end{cases} \tag{3.6}$$

Then $\sum_{j=1}^{l} X_j$ is number of marks u will get.

Apply Corollary 3.1.2, substituting $p = \frac{w}{4k}$, $r = l$, $\beta = 4$. Then we have

$$\begin{aligned} Pr[\frac{1}{l}\sum_{j=1}^{l} X_j \geq 4\frac{w}{4k}] &< \left(\frac{e^{4-1}}{4^4}\right)^{\frac{wl}{4k}} \\ \Rightarrow Pr[\sum_{j=1}^{l} X_j \geq \frac{wl}{k}] &< \left(\frac{e^3}{4^4}\right)^{\frac{w}{4k}} \\ &< \left(\frac{1}{2^3}\right)^{\frac{wl}{4k}} \qquad e^3 < 2^5 \\ &= 2^{\frac{-3wl}{4k}} \end{aligned}$$

So, the probability that $crt_u \geq \frac{wl}{k}$ is $P(A'_u) < 2^{\frac{-3wl}{4k}}$, and $P(A_u) < 2^{\frac{-3wl}{4k}}$. Then the probability that there exists an innocent user who can be framed by a coalition of size k is

$$\begin{aligned} P(A) &= \sum_{u \in U} P(A_u) \\ &= n \cdot P(A_u \\ &< n \cdot 2^{\frac{-3wl}{4k}}. \end{aligned}$$

If we choose $l \geq \frac{4k}{3w} \log(n/p)$, then

$$\begin{aligned} P(A) &< n \cdot 2^{-\log(n/p)} \\ &= n \cdot \frac{p}{n} \\ &= p. \end{aligned}$$

□

Theorem 3.6.2 *There exists a q-threshold (k,p)-resilient scheme, for any $p, q \in [0,1]$, with $l \geq \frac{4k}{3w} \log(n/p)$.*

Proof: By the above Lemma, there exists a choice of $h_i, 1 \leq i \leq l$ such that the probability that an innocent user can be framed is at most p for any $p \in [0,1]$

Therefore, if we choose hash functions $h_i, 1 \leq i \leq l$ randomly, the probability of an innocent user is mistakenly identified as a traitor is less than p. Hence we have a fully (p,k)-resilient secret traitor tracing scheme, with $l \geq \frac{4k}{3w} \log(n/p)$.

If we choose $l \geq \frac{4k}{3w} \log(n)$, then $P(A) < 1$. This tells us that there exists a choice of h_i, $1 \leq i \leq l$, such that no user will be framed by any coalition. That is, there exists a fully k-resilient scheme with $l > \frac{4k}{3w} \log(n)$. □

In the above q-threshold (k, p)-resilient scheme, the data supplier has

$$\frac{16k^2}{3w}\log(n/p) = O\left(\frac{k^2}{w}\log(n/p)\right)$$

keys. But he has to perform only $4kt$ encryption operations to encrypt the secret shares, where t is a parameter that can be picked by the data supplier such that $t = \log_w q$. One additional encryption operation is required to encrypt the content. Each user gets

$$\frac{4k}{3w}\log(n/p) = O\left(\frac{k}{w}\log(n/p)\right)$$

keys, but has to perform only $t+1$ decryption operations. Each enabling block has a size of $4kt$. We can set $t = 1$ by setting $w = q$.

The tracing algorithm performs $4kl$ decryptions using $\mathcal{D}$. Assume $\mathcal{D}$ performs a same number of work as an average user in decoding, the algorithm performs a total of

$$4klt = \frac{16tk^2}{3w}\log(n/p) + \frac{16k^2}{3w}\log n$$

decryption operations of E. The rest of the algorithm (lines 10 to 13) has time complexity $O(nlw) = O(kn\log n)$. Thus the algorithm's time complexity is

$$O\left(\left(\frac{tk^2}{w}\log(n/p)\right)C + kn\log(n/p)\right)$$

where C denotes the time complexity of performing a decryption using the underlying symmetric encryption scheme.

3.7 Two Level Threshold Scheme

The two level q-threshold (p, k)-resilient scheme presented in this section is constructed in the same way as the two level secret scheme. The scheme consists

of $\frac{2ek}{b}$ one level threshold schemes. A random hash function $h : \{1, 2, \cdots, n\} \mapsto \{1, 2, \cdots, \frac{2ek}{b}\}$ is needed to map a user to one of the $\frac{2ek}{b}$ subschemes.

Lemma 3.7.1 *If $b = \log\left(\frac{4ek}{p\log(1/p)}\right)$, the probability p' that b or more traitors are mapped to the same subscheme by h is less than $p/2$, where $p \in [\frac{1}{2}, 1]$.*

Proof:

$$\begin{aligned}
p' &= \binom{k}{b}\left(\frac{b}{2ek}\right)^{b-1} && \text{(see the proof of lemma 3.5.1)} \\
&\leq \left(\frac{ek}{b}\right)^{b}\left(\frac{b}{2ek}\right)^{b-1} \\
&= \frac{ek}{b}\frac{1}{2^{b-1}} \\
&= \frac{ek}{b}\frac{2}{\frac{4ek}{p\log(1/p)}} \\
&= \frac{p\log(1/p)}{2b} \\
&= \frac{p}{2}\frac{\log(1/p)}{b} \\
&= \frac{p}{2}\frac{\log(1/p)}{\log\left(\frac{4ek}{p\log(1/p)}\right)} \\
&= \frac{p}{2}\frac{\log(1/p)}{\log(1/p) + 2 + \log\left(\frac{ek}{\log(1/p)}\right)}.
\end{aligned}$$

Since $p \in [\frac{1}{2}, 1]$, we have $0 < log(1/p) \leq 1$, which implies $\log\left(\frac{ek}{\log(1/p)}\right) \geq \log(ek) \geq 0$. Thus,

$$\begin{aligned}
p' &= \frac{p}{2}\frac{\log(1/p)}{\log(1/p) + 2 + \log\left(\frac{ek}{\log(1/p)}\right)} \\
&\leq \frac{p}{2}\frac{\log(1/p)}{\log(1/p) + 2} \\
&\leq \frac{p}{2}.
\end{aligned}$$

□

Theorem 3.7.2 *There exists a two level q-threshold (k,p)-resilient scheme for* $p \in [\frac{1}{2}, 1]$.

Proof: Let $q' = \frac{qb}{2\epsilon k}$. We require each subscheme to be a q'-threshold $(p/2, b)$-resilient scheme. That is, each subscheme can trace the source of the keys in a decoder with a success rate of at least $1 - p/2$, if the decoder can decrypt correctly with probability at least $\frac{qb}{2\epsilon k}$. Notice that if a decoder $\mathcal{D}$ can decrypt none of the subschemes with probability at least $\frac{qb}{2\epsilon k}$, then the probability that $\mathcal{D}$ can decrypt the two level scheme is at less than $\left(\frac{qb}{2\epsilon k}\right) \cdot \left(\frac{2\epsilon k}{b}\right) = q$. Thus if a decoder $\mathcal{D}$ can decrypt the two level scheme with probability at least q, there must exist a subscheme S_c, such that $\mathcal{D}$ can decrypt with probability at least $q' = \frac{qb}{2\epsilon k}$. Since S_c is a q'-threshold $(p/2, b)$-resilient scheme, it can trace a traitor, with the probability of mistakenly identify an innocent user as a traitor being at most $p/2$, if there are less than b traitors mapped to S_i by h.

Let A_m denote the event that there is no sub-scheme to which b or more traitors are mapped simultaneously. Let A_f denote the event that the scheme fails to identify a traitor, i.e., it claims an innocent user is a traitor.

By Lemma 3.7.1, $P(\overline{A_m}) \leq \frac{p}{2}$. Then the probability that the two level scheme mistakenly claims an innocent user to be a traitor is

$$\begin{aligned} P(A_f) &= P(A_f, A_m) + P(A_f, \overline{A_m}) \\ &= P(A_f|A_m)P(A_m) + P(A_f|\overline{A_m})P(\overline{A_m}) \\ &\leq \frac{p}{2}P(A_m) + P(A_f|\overline{A_m})\frac{p}{2} \\ &\leq \frac{p}{2} + \frac{p}{2} \\ &= p \end{aligned}$$

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. for $1 \leq i \leq r$ do
2. randomly choose a permutation λ of $\{1, 2, \cdots, \frac{2\epsilon k}{b}\}$.
3. for $1 \leq l \leq \frac{2\epsilon k}{b}$ do
4. use $\mathcal{D}$ on experiment $E_{\lambda(j)}$
5. let E_c be an experiment in which the probability that $\mathcal{D}$ decrypts correctly is the lowest.
6. perform the one level tracing algorithm on scheme S_c

Figure 3.9: Tracing Algorithm for Two Level Threshold Scheme

Thus the two level scheme is a q-threshold (p, k)-resilient scheme. What remains to be shown is a tracing algorithm to find a subscheme which the pirate decoder can decrypt with probability at least $q' = \frac{qb}{2\epsilon k}$.

Let M_i be an enabling block in which the encryption with the keys of all the sub-schemes except S_i are replaced with random data. Let E_i be the following experiment: prepare an encryption session, with enabling block M_i. Figure 3.9 presents the tracing algorithm.

The tracing algorithm is very similar to the two-level secret scheme. We first need to find the subscheme S_c that $\mathcal{D}$ can decrypt with probability at least $q' = \frac{qb}{2\epsilon k}$. If $\mathcal{D}$ decrypts E_c with lowest success rate among all E_i, $1 \leq i \leq \frac{2\epsilon k}{b}$, then $\mathcal{D}$ must decrypt S_c with the highest success rate among all subschemes, since E_i disables S_i. And $\mathcal{D}$ must decrypt S_c with a success probability at least $q' = \frac{qb}{2\epsilon k}$. Then we

can use the one level tracing algorithm on S_c to find a traitor. □

The data supplier has all the keys from the $\frac{2ek}{b}$ q'-threshold $(p/2, b)$-scheme, where $q' = \frac{qb}{2ek}$. Each sub-scheme has $\frac{16b^2}{3w} \log\left(\frac{2n}{p}\right)$ base keys. So the data supplier has

$$\frac{2ek}{b}\frac{16b^2}{3w}\log\left(\frac{2n}{p}\right) = \frac{32ekb}{w}\log\left(\frac{2n}{p}\right) = O\left(\frac{kb}{w}\log\left(\frac{n}{p}\right)\right)$$

base keys in total. The data supplier performs $4bt + 1$ encryption operations in each subscheme, where $t = \log_{q'} w$, and hence $(4bt+1)\cdot\frac{2ek}{b} = 8ekt + \frac{2ek}{b}$ encryption operations.

Each user gets assigned to a sub-scheme and possesses the personal key in that sub-scheme. So, each user has $\frac{4b}{3w}\log\left(\frac{2n}{p}\right) = O\left(\frac{b}{w}\left(\frac{n}{p}\right)\right)$ keys. A user has to perform only $t + 1$ decryption operations of E.

The tracing algorithm needs to perform $r\frac{2ek}{b}$ decryption using $\mathcal{D}$ in order to determine the sub-scheme which $\mathcal{D}$ is able to decrypt. Assume $\mathcal{D}$ performs the same amount of work as an average user in decryption. Then $\frac{2ertk}{b} + r\frac{2ek}{b}$ decryption operation of E are needed. The complexity of the algorithm is $O(\frac{tk}{b})C$ plus the complexity of the first level tracing algorithm.

3.8 A Variant of CFNP One Level Open Scheme

In this section, we are going to present a variant of CFNP one level schemes. These schemes are suggested by Stinson and Wei in [14] and [15]. Let us call this scheme an SW scheme. Explicit constructions are provided for these schemes.

Key distribution: As in CFNP schemes, the data supplier has a set α of base keys chosen from the key space of a underlying symmetric encryption scheme E. Let $\mid \alpha \mid = v$, $\alpha = \{k_1, k_2, \cdots, k_v\}$. Each user will be assigned l keys. Let $P(u)$ denote the set of keys a user u receives. Note, the scheme does not require the partition of α into buckets. The l keys can be any keys from α, subject to that $P(u)$ will enable u to decrypt the content, and a traitor to be exposed. We will discuss how the assignment is done later in this section.

Encryption: Similar to the CFNP schemes, the contents are divided into sessions which has size as an integral multiple of the block size accepted by E. A session key s is randomly chosen from the key space of E to encrypt one content session. Then a (l, v) threshold secret sharing scheme is employed to produce v shares, such that knowledge of any l of these shares would allow an user to obtain s. Here we use Shamir's (l, v) threshold secret share scheme as an example.

Suppose s is in a field $\mathcal{F}_q$. At first v distinct non-zero values $x_1, x_1, \cdots, x_v \in \mathcal{F}_q$ are chosen. These values are public and can be distributed before the broadcast. The data provider chooses a_1, a_2, $\cdots$, $a_{l-1} \in \mathcal{F}_q$ independently at random. And let

$$a(x) = x(\cdots(x(x + a_{l-1}) + a_{l-2})\cdots + a_1) + s.$$

So $a(x)$ is a polynomial of degree $l-1$ in $\mathcal{F}_q$. The data supplier computes $s_i = a(x_i)$, $1 \leq i \leq v$. Each s_i is then encrypted with key k_i using the underlying encryption scheme E. The enabling block consists all encrypted shares, $\mathcal{B}_e = \|_{i=1}^{v}(E_{k_i}(s_i))$.

If a user u has l keys, k_{i_1}, k_{i_2}, $\cdots$, k_{i_l}, he can decrypt shares s_{i_1}, s_{i_2}, $\cdots$, s_{i_l}. With these l shares he can determine the polynomial $a(x)$ by using Lagrange interpolation. In fact u can compute

$$s = a(0) = \sum_{j=1}^{l} s_{i_j} b_j, \text{ where } b_j = \prod_{1 \le k \le l, k \ne j} \frac{x_{i_j}}{x_{i_k} - x_{i_j}}.$$

Since $x_i, 1 \le i \le v$ are public values available before broadcasting, each user can compute them in advance.

Now, suppose a pirate decoder $\mathcal{D}$ can also decrypts the content. Assume the underlying encryption scheme E is computationally secure. Since an (l, v)-threshold scheme is used, $\mathcal{D}$ must be able to obtain at least l shares of s_i, $1 \le i \le v$ from the enabling block. In order to decrypt l shares, $\mathcal{D}$ must obtain l keys. Thus we can assume that $\mathcal{D}$ must know at least l keys if it can decrypt content successfully.

Tracing algorithm: Let us first assume that a pirate decoder $\mathcal{D}$ has a set of $l' \ge l$ keys, $k_{i_1}, k_{i_2}, \cdots, k_{i_{l'}}$, and always uses them in the construction of s, i.e. $\mathcal{D}$ always computes

$$s = a(0) = \sum_{j=1}^{l'} s_{i_j} b_j, \text{ where } b_j = \prod_{1 \le k \le l', k \ne j} \frac{x_{i_j}}{x_{i_k} - x_{i_j}}.$$

Then if we replace any one share of $s_{i_1}, s_{i_2}, \cdots, s_{i_{l'}}$ with random data. $\mathcal{D}$ will compute an incorrect s and fail to decrypt the content.

The tracing algorithm is presented in Figure 3.10. In each iteration of the first for loop, we remove a share $s_{\lambda(i)}$. If $\mathcal{D}$ decrypts successfully using enabling block M_i, then $\mathcal{D}$ does not use $s_{\lambda(i)}$ in decoding. If $\mathcal{D}$ fails, we know $\mathcal{D}$ has key $k_{\lambda(i)}$, and uses it in decryption. But, then we restore that share so that $\mathcal{D}$ can still decrypt.

Unfortunately, the algorithm will work only when $\mathcal{D}$ always use a certain set of $l' \ge l$ keys in decryption. If D has more than l keys and randomly chooses l of

Input: pirate decoder $\mathcal{D}$

Output: one traitor

1. randomly pick a permutation λ on $\{1, 2, \cdots v\}$
2. let M_0 be a legitimate enabling block
3. for $1 \leq i \leq v$ do
4. construct M_i from M_{i-1} by replacing the encrypted share $s_{\lambda(i)}$ by random data. (this has the affect of removing share $s_{\lambda(i)}$)
5. try $\mathcal{D}$ on M_i
6. if $\mathcal{D}$ fails
7. $M_i = M_{i-1}$ (restoring share $s_{\lambda(i)}$)
8. $F = F \cup \{\lambda(i)\}$
9. for each $u \in U$ do
10. compute $\mid F \cap P(u) \mid$
11. return u with the highest value of $\mid F \cap P(u) \mid$ among U.

Figure 3.10: Tracing Algorithm for One Level Non-Transversal Tracing Scheme

them in each session, we have to modify the algorithm, since if even $\mathcal{D}$ decrypts correctly on enabling block M_i, it does not necessary mean $\mathcal{D}$ does not possess key $k_{\lambda(i)}$, since it could be that $k_{\lambda(i)}$ was not chosen for this session. We need to perform more decryption attempts on each enabling block M_i to make sure that $\mathcal{D}$ does not possess $k_{\lambda(i)}$.

How many times do we have to perform the decryption on a particular enabling block M_i? There are k traitors, $\mathcal{D}$ could have a set F of as many as kl distinct keys, it can randomly choose l keys among F in each session. So for a particular key in the probability it is chosen to decrypt is $\frac{k-1}{k} = 1 - \frac{1}{k}$. If k attempts are performed, the probability that a particular key is not chosen is $\left(1 - \frac{1}{k}\right)^k \leq e^{-1}$. So if rk decryption attempts are performed, the probability is at most e^{-r}. We can see from Table 3.1 that if we choose $r \geq 10$, the probability that a specific key will be chosen at least once after rk decryption attempts is very close to 1. Suppose we perform $10k$ decryption attempts on enabling blocks M_i, and assume $\mathcal{D}$ does not intentionally decrypt incorrectly. If $\mathcal{D}$ does not have key $k_{\lambda(i)}$, then decryption will always succeed. Otherwise there is at least one failure with probability at least 0.9999546. Thus the algorithm will make rkv decryption attempts using $\mathcal{D}$.

However this tracing algorithm suffers a similar drawback as those in CFNP schemes. If $\mathcal{D}$ has more than l keys, it can detect the operation of the tracing algorithm. Suppose $\mathcal{D}$ has $l+1$ keys and hence $l+1$ shares. D can compute s twice using different l shares chosen from the $l+1$ shares it has. If one of the $l+1$ keys is invalidated, then the two s computed would differ. This involves only a bit more work than detecting the tracing algorithm in CFNP schemes.

Now, back to the key assignment. Suppose a coalition T of at most k traitors builds a pirate decoder $\mathcal{D}$. Let F be a set of l keys in $\mathcal{D}$. $F \subseteq \cup_{t \in T} P(t)$. We call a user u *exposed*, if $\mid F \cap P(u) \mid \geq \mid F \cap P(v) \mid$, for all $v \in U$. We require that the

exposed user is a traitor for any coalition of size at most k.

A set system is a pair $(\mathcal{X}, \mathcal{B})$, where $\mathcal{X}$ is a set of elements called *points* and $\mathcal{B}$ is set of subsets of $\mathcal{X}$, the members of which are called *blocks*. A set system can be described by an incidence matrix. Let $(\mathcal{X}, \mathcal{B})$ be a set system, where $\mathcal{X}$= $\{x_1, x_2, \cdots, x_v\}$ and $\mathcal{B}$= $\{B_1, B_2, \cdots, B_b\}$. The incidence matrix of $(\mathcal{X}, \mathcal{B})$ is the $b \times v$ matrix $A = (a_{ij})$ where

$$a_{ij} = \begin{cases} 1 & \text{if } x_j \in B_i \\ 0 & \text{if } x_j \notin B_i \end{cases} .$$

We can think the key assignment scheme as a set system, where $\mathcal{X}$ is the set of base keys and $\mathcal{B}$ is the set of personal keys for each user. To satisfy the requirement that every exposed user must be a traitor, let's consider the concept of a *traceable set system.*

Definition 3.8.1 *A traceable set system is a set system $(\mathcal{X}, \mathcal{B})$, where $|\mathcal{X}|= v$, $|\mathcal{B}|= b$, and every block has size l for some integer l, with the property that for every choice of $k' \leq k$ blocks, $B_1, B_2, \cdots, B_{k'} \in \mathcal{B}$, and any l-subset $F \subseteq \cup_{i=1}^{k'} B_i$, there does not exist a block $B \in \mathcal{B} \backslash \{B_1, B_2, \cdots, B_{k'}\}$ such that $\mid F \cap B_i \mid \leq \mid F \cap B \mid$, for $1 \leq i \leq k'$. We denote such a system by k-(l, b, v)-TSS.* □

Theorem 3.8.2 *If there exits a k-(l, b, v)-TSS, there exists a k-resilient traitor tracing scheme for n users with v base keys, and each user has l keys.*

Proof: Suppose $(\mathcal{X}, \mathcal{B})$ is a k-(l, b, v)-TSS, $\mathcal{B}$= $\{B_1, B_2, \cdots, B_b\}$. We only need to show that if we use $(\mathcal{X}, \mathcal{B})$ as our key assignment scheme, i.e. $P(u) = B_u$, then any exposed user is a traitor.

Let T be a coalition of size at most k, $T = \{i_1, i_2, \cdots, i_{k'}\}, k' \leq k$. Consider any F that $F \subseteq \cup_{t \in T} B_t, \mid F \mid = l$. Since $(\mathcal{X}, \mathcal{B})$ is a k-(l, b, v)-TSS, there is no $B \in \mathcal{B} \backslash \{B_{i_1}, B_{i_2}, \cdots, B_{i_{k'}}\}$, such that

$$\mid F \cap B_{i_j} \mid \leq \mid F \cap B \mid, 1 \leq j \leq k'.$$

Then, there is no $u \in U \setminus T$ such that

$$\mid F \cap P(t) \mid \leq \mid F \cap P(u) \mid, \text{for all } t \in T.$$

So, there must exist a $t' \in T$ such that

$$\mid F \cap P(t') \mid > \mid F \cap P(u) \mid, \text{for all } u \in U \setminus T.$$

Now suppose $\overline{t}$ is an exposed user. Then

$$\mid F \cap P(t) \mid \geq \mid F \cap P(t') \mid > \mid F \cap P(u) \mid, \text{for all } u \in U \setminus T.$$

So $\overline{t} \notin U \setminus T$, and $\overline{t}$ must be a traitor. □.

In the rest of this section we are going to give some constructions of traceable set systems.

Definition 3.8.3 *A t-(v, l, λ) design is a set system $(\mathcal{X}, \mathcal{B})$, where $\mid \mathcal{X} \mid = v$, $\mid B \mid = l$ for every $B \in \mathcal{B}$, and every t subset of $\mathcal{X}$ occurs in exactly λ blocks in $\mathcal{B}$. The number of blocks in $\mathcal{B}$ is* $\begin{pmatrix} v \\ t \end{pmatrix} \Big/ \begin{pmatrix} l \\ t \end{pmatrix}$. □

Theorem 3.8.4 *If there exists a t-$(v, l, 1)$ design, then there exists a*

$$k\text{-}\left(l, \begin{pmatrix} v \\ t \end{pmatrix} \Big/ \begin{pmatrix} l \\ t \end{pmatrix}, v\right)\text{-}TSS,$$

where $k = \lfloor \sqrt{(l-1)/(t-1)} \rfloor$.

Proof: Let $(\mathcal{X}, \mathcal{B})$ be a t-$(v, l, 1)$ design. Let $B_1, B_2, \cdots, B_{k'}, k \leq k$ be k' distinct blocks. Let $B \in \mathcal{B} \backslash \{B_{i_1}, B_{i_2}, \cdots, B_{i_{k'}}\}$. Let $F \subseteq \cup_{i=1}^{k'} B_i, \mid F \mid = l$. Notice there exists a $B_i, 1 \leq i \leq k'$ such that

$$\begin{aligned} \mid F \cap B_i \mid &\geq \lceil \frac{l}{k} \rceil \\ &\geq \frac{l}{\lfloor \sqrt{(l-1)/(t-1)} \rfloor} \\ &\geq \sqrt{\frac{(t-1)l^2}{l-1}} \\ &> \sqrt{(t-1)(l-1)} \end{aligned}$$

Since every t-tuple occurs in exactly one block, any 2 blocks have at most $t-1$ points in common. So $\mid B \cap B_j \mid \leq t-1, 1 \leq j \leq k'$. Then

$$\begin{aligned} \mid B \cap F \mid &\leq (t-1)k \\ &= (t-1)\lfloor \sqrt{(l-1)/(t-1)} \rfloor \\ &\leq \sqrt{(t-1)^2(l-1)/(t-1)} \\ &= \sqrt{(t-1)(l-1)} \\ &< \mid F \cap B_i \mid \end{aligned}$$

Therefore $(\mathcal{X}, \mathcal{B})$ is a k-$\left(l, \begin{pmatrix} v \\ t \end{pmatrix} / \begin{pmatrix} l \\ t \end{pmatrix}, v\right)$-TSS, □

When $t = 2$, a t-(v, l, λ) design is called a *Balanced Incomplete Block Design (BIBD)*. A $(q^2+q+1, q+1, 1)$-BIBD is known to exits for every prime power q, and is called a *projective plane* of order q. There are $\frac{(q^2+q+1)(q^2+q)}{(q+1)q} = q^2 + q + 1$ blocks in a projective plane of order q. Thus we have a $\sqrt{q}-(q+1, q^2+q+1, q^2+q+1)$-TSS for any prime power q. And hence we have a $\sqrt{q}$-traitor tracing scheme with $q^2 + q + 1$ base keys, $q^2 + q + 1$ users and each user has $q + 1$ personal keys, for every prime power q.

Example 3.8.5 *Here we present a* 3-$(10, 91, 91)$-*TSS. The set of points are in* $\mathcal{Z}_{91}$. *Block* $B_i, 0 \leq i \leq 90$ *contains points,*

$$B_i = \{0+i, 1+i, 6+i, 10+i, 23+i, 26+i, 34+i, 41+i, 53+i, 55+i\},$$

where all arithmetic is done in $\mathcal{Z}_{91}$. *This set system is a projective plane of order 9.*

In fact $D' = \{0, 1, 6, 10, 23, 26, 34, 41, 53, 55\}$ *is a* $(91, 10, 1)$ *cyclic difference set. A* (v, l, λ) *cyclic difference set is a set* $D = \{d_1, d_2, \cdots, d_l\}$ *such that each non-zero element* $d \in \mathcal{Z}_v$ *can be expressed in the form* $d = d_i - d_j$, *in precisely* λ *ways. It can be easily verified that each nonzero element in* $\mathcal{Z}_{91}$ *can be expressed as a difference of two elements in* D' *in precisely one way.*

It is known that if $D = \{d_1, d_2, \cdots, d_l\}$ *is a cyclic* (v, l, λ) *difference set, then* $D, D+1, \cdots, D+(v-1)$ *are the blocks of symmetric* (v, l, λ)-*BIBD, where* $D + a = \{d_1 + a, d_2 + a, \cdots, d_l + a\}$.

Thus

$$B_i = \{0+i, 1+i, 6+i, 10+i, 23+i, 26+i, 34+i, 41+i, 53+i, 55+i\},$$

$0 \leq i \leq 90$, *form a* $(91, 10, 1)$-*BIBD.* □

Example 3.8.6

$$D' = \{0, 1, 20, 30, 35, 107, 125, 131, 153, 157, 174, 210, 219, 222, 233, 235, 266\}$$

is a cyclic (273, 17,1) difference set. Then $D', D'+1, \cdots, D'+272$ *is a* $(273, 17, 1)$-*BIBD, i.e. a projective plane of order 16. Thus we have a* 4-$(17, 273, 273)$-*TSS.* □

And, a $(q^2, q, 1)$-BIBD is called an *affine plane* or order q, and also exists for every prime power q. There are $\frac{q^2(q^2-1)}{q(q-1)} = q^2 + q$ blocks in an affine plane of order q. So, an affine plane of order q is also a $\sqrt{q-1} - (q, q^2, q^2 + q)$-TSS.

An affine plane of order q can be obtained from a projective plane of order q by removing the elements in a block B' from all other blocks. A block has $q + 1$ points, so there are q^2 points left. Since no other block contains all $q + 1$ points being removed, there are $q^2 + q$ blocks left. There are $(q + 1)(q^2)$ pairs with one point in B' and the other not in B'. There are $q^2 + q$ blocks, each block can only have at most one point in common with B', so has at most q pairs with one point in B' nd the other not in B'. But $q(q^2 + q) = (q + 1)(q^2)$ which is the total number of such pairs. So each block must have exactly q pairs and hence exactly one point in B'. Thus after removing the points in B', each remaining block has q points each. And every pair of remaining points occur in exactly one block, since it does occur in B'. So, we have a $(q^2, q, 1)$-BIBD, an affine plane of order q.

Example 3.8.7

$$D' = \{0, 1, 3, 30, 37, 50, 55, 76, 98, 117, 129, 133, 157, 189, 199, 222, 293, 299\}$$

is a cyclic $(307, 18, 1)$ *difference set. Then* $D', D' + 1, \cdots, D' + 306$ *is a* $(307, 17, 1)$*-BIBD, i.e. a projective plane of order 17. So, if we remove all elements in any one block we get a* $(289, 17, 1)$*-BIBD, an affine plane of order 17. Thus we have a* 4-$(17, 289, 306)$*-TSS. Compared to the previous example, there are 16 more base keys, and 33 more users, with same number of personal keys per user.* □

Definition 3.8.8 *A* $t-(v, l, \lambda)$ *packing design is a set system* $(\mathcal{X}, \mathcal{B})$*, where* $|\mathcal{X}| = v$*,* $| B | = l$ *for every* $B \in \mathcal{B}$*, and every* t *subset of* $\mathcal{X}$ *occurs in at most* λ *blocks in* $\mathcal{B}$*.* □

Theorem 3.8.9 *If there exists a t-$(v, l, 1)$ packing design with b blocks, then there exists a k-(l, bv)-TSS, where $k = \lfloor \sqrt{(l-1)/(t-1)} \rfloor$.*

Proof: Same as the proof for Theorem 3.8.2. □

It is not known if t-$(v, l, 1)$ design exist if $v > l > t \geq 6$. However there are infinite classes of packing designs with number of blocks close to $\begin{pmatrix} v \\ t \end{pmatrix} / \begin{pmatrix} l \\ t \end{pmatrix}$. They can be obtained from orthogonal arrays.

Theorem 3.8.10 *If there exists an $OA(t, l, m)$, then there exists a t-$(lm, l, 1)$ packing design which has s^t blocks.*

Proof: If there is an $OA(t, l, m)$ with entries from the set $\{0, 1, \cdots, s-1\}$. Define $(\mathcal{X} = \{(x, y) : 0 \leq x \leq l-1, 0 \leq y \leq s-1\}$. For every column $(y_0, y_1, \cdots, y_{l-1})$ in the OA, define a block $B = \{(0, y_0), (1, y_1), \cdots, (l-1, y_{l-1})\}$. Let $\mathcal{B}$ be the set of s^t blocks constructed.

Suppose $B_i, B_j \in \mathcal{B}$ have a t-tuple in common. WLOG say $(0, y_0)$, $(1, y_1)$, $\cdots$, (t, y_t). Consider column i and j of the OA. The first t entries of these 2 columns are the same. This contradicts the fact that any $t \times 1$ column vector appears exactly once in the first t rows.

Therefore, any t-tuple occurs in at most one block in $\mathcal{B}$, and $(\mathcal{X}, \mathcal{B})$ is a t-$(lm, l, 1)$ packing design. □

Recall that we also used an orthogonal array to construct a CNFP open one level scheme in section 3.2. In fact the same orthogonal array can also used to construct a traceability scheme introduced in this section. We know that an $OA(t, q+1, q)$ exists for all prime powers q and $t < q$. Thus by the above theorem, there is a

$\lfloor\sqrt{q/(t-1)}\rfloor$ -traceability scheme with $n = q^t$ users, where each user has $l = q + 1$ base keys, and there are total of $v = q^2 + q$ base keys. In fact, using OAs, the SW schemes also has transversal property.

Using OAs as the key assignment scheme in the scheme presented here is not as efficient as in the CNFP one level scheme, since the data provider and each user has to perform extra computation due to the threshold secret sharing scheme.

Assume Shamir's (l, v) threshold secret sharing scheme is used. The data provider has $q^2 + q$ keys which is same as in CNFP scheme. But he also has to perform v polynomial evaluations to compute $s_i = a(x_i), 1 \leq i \leq v$. Each evaluation can be done using l multiplications and $l + 1$ additions in $\mathcal{F}_q$. With $v = q^2 + q$, $l = q + 1$, the data supplier has to perform $q^3 + 2q^2 + q$ multiplications, and $q^3 + 3q^2 + 2q$ additions in $\mathcal{F}_q$.

As in a CNFP scheme, each user needs at least l decryption to obtain l shares. But each user has to perform l multiplications and l additions to obtain s using Lagrange interpolation, while in a CNFP scheme, only $l - 1$ XORs are needed to obtain s from l shares.

Recall that the tracing algorithm makes rkv decryption attempts using $\mathcal{D}$. Assume $\mathcal{D}$ does the same amount of work as an user. The tracing algorithm would perform $rkv(l + 1)$ decryption operations, $rkvl$ additions and multiplications. The complexity is $O(kvlC + n)$, where C is the complexity of the decrypting operation of E. If an $OA(t, q + 1, q)$ is used, $v = q^2 + q$, $n = q^t$, $l = q + 1$, $k = \lfloor\sqrt{q/(t-1)}\rfloor$. The running time is $O(q^{3\frac{1}{2}}C + q^t)$. The one level open CNFP scheme has a tracing algorithm with complexity $O(vlC + n)$. Using an $OA(t, q + 1, q)$ as the key distribution scheme, the tracing algorithm has complexity of $O(q^3C + q^t)$.

The data redundancy is the same in both schemes. The tracing algorithm

Schemes	personal keys	base keys	data redundancy	tracing algorithm
One Level Open Scheme	$O(k^2 \log n)$	$O(k^4 \log n)$	$O(k^4 \log n)$	$O(k^6(\log^2 n)C + k^6 \log^2 n)$
Two Level Open Scheme	$O(k^2 \log^2 k \log \frac{n}{k})$	$O(k^3 \log^4 k \log \frac{n}{k})$	$O(k^3 \log^4 k \log \frac{n}{k})$	$O(k^5 \log^6 k(\log^2 \frac{n}{k})C + nk^2 \log^2 k \log \frac{n}{k}$
One Level Secret Scheme	$O(k \log \frac{n}{k})$	$O(k^2 \log \frac{n}{k})$	$O(k^2 \log \frac{n}{k})$	$O(k^4(\log^2 \frac{n}{k})C + nk^2 \log \frac{n}{k}$
Two Level Secret Scheme	$O(b \log \frac{n}{p})$ $b = \log \frac{4}{p}$	$O(kb^2 \log \frac{n}{p})$ if $k \leq \frac{1}{2p} \log \frac{4}{p}$	$O(kb^2 \log \frac{n}{p})$ if $k \leq \frac{1}{2p} \log \frac{4}{p}$	$O(kb(\log \frac{n}{p})C)$ + one level secret tracing
One Level Threshold	$O(\frac{k}{w} \log \frac{n}{p})$ $t = \log_w q$	$O(\frac{k^2}{w} \log \frac{n}{p})$	$4kt$	$O(\frac{tk^2}{w}(\log \frac{n}{p})C + kn \log \frac{n}{p}$
Two Level Threshold	$O(\frac{b}{w} \log \frac{n}{p})$ $b = \log\left(\frac{4ek}{p \log \frac{1}{p}}\right)$	$O(\frac{kb}{w} \log \frac{n}{p})$	$8ekt$	$O(\frac{tk}{b}C)$ + one level threshold tracing

Table 3.2: Efficiency Measures for CFNP Schemes

presented in this section needs v decryption attempts using a pirate decoder, which is same as in CNFP scheme.

3.9 Summary

Table 3.2 gives a summary of the efficiency measures of the six presented CFNP schemes. In the first four schemes the number of decryption operations each user (i.e. his decoder) has to perform in each session is same as the number of personal keys, since all personal keys are used in every session. In the two threshold schemes,

each user performs only $t = \log_w q$ decrypting operations. For the last four schemes, the measures are for p-resilient schemes that can be obtained by choosing hash functions randomly. Setting $p = 1$ we have the measures for fully resilient schemes whose existence have been proved.

Not surprising, we can see that the secret schemes are more efficient than open schemes in every aspect. While threshold schemes are less efficient than the secret schemes in terms of personal keys and total base keys by a factor of $\frac{1}{w}, q \leq w \leq 1$, they reduce the amount of work each user and the data supplier have to perform, as well as the data redundancy.

The one level open scheme has explicit constructions that are almost as efficient as the existence result. No explicit construction are known for the two level open scheme. The fully resilient version of the last four schemes do not have explicit construction either, while it is possible to construct p-resilient version of the schemes by choose hash functions randomly. But having a scheme which might identify an innocent user as a traitor with probability p is generally not acceptable unless p is really small, which leads to a substantial increase in almost every efficient measure.

Although the tracing algorithm in each scheme is able to identify at least one traitor, it works only if the pirate decoder does not have redundant keys. For example, in a one level open scheme, the tracing algorithm works only when the pirate decoder has exactly one key for each bucket. The algorithm can be easily defeated if the decoder has 2 keys in any bucket, by comparing the shares decrypted using the 2 keys. Once the decoder detects that it is being inquired by a tracing algorithm, it can fail the decoding intentionally. The assumption that no pirate decoder has any redundant key does not seem to be a very valid one. SW schemes suffer from a similar problem.

Schemes	personal keys	each user's work	base keys	users	tracing
CFNP using $OA(t, q+1, q)$	$q+1$	$q+1$ decrypt q XOR	q^2+q	q^t	$O(q^3C+q^t)$
SW using $BIBD$ $(q^2+q+1,$ $q+1, q)$	$q+1$	$(q+1)\times$, $(q+1)+$, $q+1$ decrypt	q^2+q+1	q^2+q+1	$O(q^{3\frac{1}{2}}C)$

Table 3.3: Comparison Between One Level Open CFNP and SW Schemes

Table 3.3 gives a comparison between one construction for CFNP and SW one level schemes. The CFNP scheme uses an $OA(t, q+1, q)$ as key distribution scheme while the SW scheme uses a $BIBD(q^2+q+1, q+1, q)$. Generally the CFNP scheme allows more users when $t > 2$. Even when $t = 2$ although the SW scheme accommodates more users, each user and the tracing algorithm has to do more work because SW uses a (l, v) secret sharing scheme.

All of the above schemes are symmetric, since the data supplier knows every user's personal key. If the data supplier is dishonest, he can easily frame any user as a traitor. This leads to asymmetric traceability schemes that will be presented in the next chapter.

Chapter 4

Asymmetric Tracing Scheme

All the schemes presented in the previous chapter are symmetric in the sense that all the secret shares are encrypted and decrypted using the same keys. Each user shares his personal key with the data supplier. One concern is that the data supplier might be dishonest and frame any user as a traitor. This motivates research on a tracing scheme that works like a public key cryptosystem, in which the encryption key is public while the decryption key is kept secret. The two schemes introduced in this chapter are *asymmetric* in the sense that the encryption key and the decryption key are different. However they do not fully qualify as a truly public-key system, as the decryption keys are assigned by the data supplier. Thus DS can still frame any user at his will. These schemes do not provide non-repudiation. We will discuss a truly public-key traceability system in the next chapter.

Both schemes presented here use an "asymmetric" encryption scheme to encrypt/decrypt session keys for each session. An off-the-shelf symmetric key encryption system is then used to encrypt/decrypt the actual contents using the session key. Thus, in the following we are only concerned with how a session key is en-

crypted by the data supplier and how it is decrypted by each user.

The first such scheme was due to Kurosawa and Desmedt in 1998 [10]. The encryption scheme is as secure as the ElGamal cryptosystem. However the tracing algorithm was broken shortly after it was published. We present the scheme here to illustrate a pitfall in designing an asymmetric key traceability scheme. The second scheme was proposed by Boneh and Franklin in 1999 [3]. The encryption scheme is based on the discrete log representation problem with respect to a fixed base of group elements. This scheme is secure if the decision Diffie-Hellman problem is hard in the underlying group. The tracing algorithm is equivalent to decoding a received word to a codeword of an error correcting linear code of distance $2k+1$. Given that a key is captured from a pirate decoder, the algorithm is able to trace *all* of the traitors who have contributed to the decoder.

4.1 Kurosawa-Desmedt Scheme

Key generation and distribution: Let p be a prime power, and q be a prime such that $q \mid p-1$, and $q \geq n+1$. Let g be a qth root of unity over $GF(p)$. p, q, g are public information. Session keys are randomly chosen from the cyclic multiplicative subgroup generated by g: $S =< g >= \{s \mid s = g^i, 0 \leq i \leq q-1\}$. The data supplier chooses a random polynomial:

$$f(x) = a_0 + a_1 x + \cdots + a_k x^k$$

where $a_0, a_1, \cdots, a_k \in \mathcal{Z}_q$. Then he gives $(i, f(i))$ to user i as i's personal key. The data supplier also computes:

$$y_0 = g^{a_0}, y_1 = g^{a_1}, \cdots, y_k = g^{a_k}$$

in $GF(p)$. The encryption key is $(y_0, y_1, \cdots, y_k)$.

Encryption/Decryption: To encrypt a session key s, the data supplier randomly chooses an integer r and construct an enabling block:

$$(g^r, sy_0^r, y_1^r, \cdots, y_k^r).$$

From the enabling block each user i computes:

$$\begin{aligned}
& \frac{sy_0^r \times (y_1^r)^i \times (y_2^r)^{i^2} \times \cdots \times (y_k^r)^{i^k}}{(g^r)^{f(i)}} \\
= & \frac{sg^{ra_0} \times g^{ra_1 i} \times g^{ra_2 i^2} \times \cdots \times g^{ra_k i^k}}{(g^r)^{f(i)}} \\
= & \frac{sg^{r(a_0 + a_1 i + a_2 i^2 + \cdots + a_k i^k)}}{(g^r)^{f(i)}} \\
= & \frac{sg^{rf(i)}}{(g^r)^{f(i)}} \\
= & s.
\end{aligned}$$

Thus each user can obtain s and decrypt the content.

Let us call the above encryption scheme the KD encryption scheme. It can be shown that this encryption scheme is as secure as ElGamal encryption scheme.

Lemma 4.1.1 *The KD encryption scheme is as secure as the ElGamal encryption scheme in $GF(p)$ if $k = O(\log p)$.*

Proof: Let $\mathcal{M}_1$ denote the problem of finding s given $(g, y), (g^r, sy^r)$ in $\mathcal{Z}_p$ where $y = g^a$, and a,r are secret. So $\mathcal{M}_1$ is the problem of breaking ElGamal encryption scheme. Let $\mathcal{M}_2$ denote the problem of breaking the KD encryption scheme, i.e., find s given $(g^r, sy_0^r, y_1^r, \cdots, y_k^r)$ in $GF(p)$ where $y_0 = g^{a_0}, y_1 = g^{a_1}, \cdots, y_k = g^{a_k}$, for random secret values $a_0, a_1, \cdots, a_k, r$. We would like to show there is an algorithm solving $\mathcal{M}_1$ if and only if there is algorithm solving $\mathcal{M}_2$.

First suppose there is an algorithm A_1 that solves $\mathcal{M}_1$. Inputting (g^r, sy_0^r) to A_1 will give s and hence solve $\mathcal{M}_2$.

Next suppose there is an algorithm A_2 that solves $\mathcal{M}_2$. We choose a_1, a_2, $\cdots$, $a_k \in \mathcal{Z}_q$ at random and compute

$$y_1^r = (g^r)^{a_1}, y_2^r = (g^r)^{a_2}, \cdots, y_k^r = (g^r)^{a_k}.$$

Inputting $(g^r, sy^r, y_1^r, \cdots, y_k^r)$ to A_2 will reveal s. The reduction is in polynomial time if $k = O(\log p)$. □

Tracing: The tracing algorithm is rather simple and does not support blackbox tracing. It assumes the decryption keys in the decoder can be revealed and the key has the form $(i, f(i))$. User i is declared as a traitor.

Let T be a coalition of at most k traitors. It can be showed that if T can construct a pirate decoder with decryption key $(u, f(u))$ where $u \notin T$, then the discrete logarithm problem can be solved.

Theorem 4.1.2 *Let T be a coalition of at most k traitors, where $k = O(\log p)$. If there is a polynomial time algorithm that allows the traitors in T to construct a pirate decoder with decryption key $(u, f(u))$ where $u \notin T$, then there is a polynomial time algorithm for the discrete log problem in $GF(p)$.*

Proof: Let $T = \{i_1, i_2, \cdots, i_{k'}\}$. Let A_1 be a polynomial time algorithm which takes the encryption key of the tracing scheme and traitor's personal keys as input and outputs a pirate decoder with decryption key $(u, f(u))$ where $u \notin T$. We are going to present a polynomial time algorithm A_2 which solves the discrete log problem using A_1 as a subroutine.

Suppose we are given an instance of the discrete log problem: $g, y = g^a$. A_2 first chooses $d_1, d_2, \cdots, d_{k'} \in GF(p)$ at random. Then there exists unique polynomial

$f(x) = a + a_1 x + \cdots + a_k x^{k'}$ such that $f(i_j) = d_j$, $1 \leq j \leq k'$. All the following arithmetic is done in $GF(p)$.

Define the matrix B as follows:

$$B = \begin{pmatrix} i_1 & i_1^2 & \cdots & i_1^{k'} \\ i_2 & i_2^2 & \cdots & i_2^{k'} \\ \vdots & \vdots & \vdots & \vdots \\ i_{k'} & i_{k'}^2 & \cdots & i_{k'}^{k'} \end{pmatrix}.$$

Then,

$$\begin{aligned} (d_1, d_2, \cdots, d_{k'})^T &= (f(i_1), f(i_2), \cdots, f(i_{k'}))^T \\ &= (a, a, \cdots, a)^T + B \times (a_1, a_2, \cdots, a_{k'})^T. \end{aligned}$$

Notice that B is a Vandermonde matrix, and hence it is nonsingular. So, we have

$$(a_1, a_2, \cdots, a_{k'})^T = B^{-1} \times (d_1 - a, d_2 - a, \cdots, d_{k'} - a)^T.$$

Let $(b_{j1}, b_{j2}, \cdots, b_{jk'})$ be the jth row of B^{-1}. Then

$$\begin{aligned} a_j &= b_{j1}(d_1 - a) + b_{j2}(d_2 - a) + \cdots + b_{jk'}(d_{k'} - a) \\ &= b_{j1}d_1 + b_{j2}d_2 + \cdots + b_{jk'}d_{k'} - (b_{j1} + b_{j2} + \cdots + b_{jk'})a. \end{aligned}$$

Hence,

$$\begin{aligned} g^{a_j} &= \frac{g^{b_{j1}d_1 + b_{j2}d_2 + \cdots + b_{jk'}d_{k'}}}{g^{(b_{j1} + b_{j2} + \cdots + b_{jk'})a}} \\ &= \frac{g^{b_{j1}d_1 + b_{j2}d_2 + \cdots + b_{jk'}d_{k'}}}{y^{(b_{j1} + b_{j2} + \cdots + b_{jk'})}}. \end{aligned}$$

Notice that $g, y, b_{j1}, \cdots, b_{jk'}, d_1, \cdots, d_{k'}$ are all known. Thus g^{a_j}, $1 \leq j \leq k'$, can be computed. Then $(y, g^{a_1}, g^{a_2}, \cdots, g^{a_{k'}})$ is a valid encryption key for a KD scheme. Further $(i_j, d_j = f(i_j))$, $1 \leq j \leq k'$, are k' valid decryption keys for the scheme.

Applying A_1, we obtain $(u, f(u))$, where $u \notin T$. Then A_2 can interpolate $f(x)$, using (i_1, d_1), (i_2, d_2), $\cdots$, $(i_{k'}, d_{k'})$ and $(u, f(u))$. Then $a = f(0)$ which is the discrete log of y.

A_2 runs in polynomial time as long as A_1 is a polynomial time algorithm, and $k = O(\log p)$. □

Everything seems sound. But, notice the tracing algorithm assumes that the key in any pirate decoder has the form $(u, f(u))$. However, a pirate decoder does not have to restrict itself to use a key in such a form, even though a legitimate decoder uses $(i, f(i))$ as its decryption key. A pirate decoder can use a key in any form as long as it allows the decryption of the content. We are going to present a key in a different form which still allows the decryption of session keys, but which prevents any traitors from being traced. This attack was adviced in [3] and [15].

Consider a coalition T of $t \geq 2$ users. Let $T = \{i_1, i_2, \cdots i_t\}$. Let $v_j = (f(i_j), 1, i_j, i_j^2, \cdots, i_j^k), 1 \leq j \leq t$. Consider a convex combination of v_j's:

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_t v_t, \sum_{j=1}^{t} \alpha_j = 1.$$

Let $w = (u, w_0, w_1, \cdots, w_k)$.

A pirate decoder can use w to decrypt a session key s from ciphertext

$$(g^r, s y_0^r, y_1^r, \cdots, y_k^r)$$

as follows: compute

$$\begin{aligned}
& (s y_0^r)^{w_0} \times ((y_1)^r)^{w_1} \times ((y_2)^r)^{w_2} \times \cdots \times ((y_k)^r)^{w_k} \\
= \; & (s y_0^r)^{\sum_{j=1}^{t} \alpha_j} \times ((y_1)^r)^{\sum_{j=1}^{t} \alpha_j i_j} \times ((y_2)^r)^{\sum_{j=1}^{t} \alpha_j i_j^2} \times \cdots \times ((y_k)^r)^{\sum_{j=1}^{t} \alpha_j i_j^k} \\
= \; & s^{\sum_{j=1}^{t} \alpha_j} \times g^{r a_0 \sum_{j=1}^{t} \alpha_j} \times g^{r a_1 \sum_{j=1}^{t} \alpha_j i_j} \times g^{r a_2 \sum_{j=1}^{t} \alpha_j i_j^2} \times \cdots \times g^{r a_k \sum_{j=1}^{t} \alpha_j i_j^k}
\end{aligned}$$

$$= s \times g^{r \sum_{j=1}^{t} \alpha_j \sum_{l=0}^{k} a_l i_j^l}$$
$$= s \times g^{r \sum_{j=1}^{t} \alpha_j f(i_j)}.$$

Now, multiplying it by the inverse of $(g^r)^u = g^{r \sum_{j=1}^{t} \alpha_j f(i_j)}$, we obtain s. Thus a pirate decoder does not have to use a key having the form $(u, f(u))$ to decrypt s. Hence the tracing algorithm devised above is useless in this situation.

Thus in designing a traceability scheme, one should not assume that all pirate decoders work in the same way as a legitimate decoder. It is important for the tracing algorithm to work regardless how a pirate decoder is implemented. Notice that there is no such problem in the symmetric traceability schemes in the previous chapter. If a secret share is encrypted using a key, then this key is usually the only key that can be used to recover the secret share.

One way to thwart the above attack is to choose $f(x)$ to be a polynomial of degree $2k$. Notice that any $2k+1$ of $v_j = (1, j, j^2, \cdots, j^{2k})$ are linearly independent. Suppose w is a linear combination of $t \leq k$ vectors in $V = \{v_j : 1 \leq j \leq n\}$: $w = \sum_{i=1}^{t} \alpha_i v_{j_i}$, $\alpha_i \neq 0, 1 \leq i \leq t$. Then w cannot be expressed as a linear combination (with non-zero coefficients) of a different set of less than $k+1$ vectors in V. Otherwise we have a set of less than $2k+1$ linearly dependent vectors.

If the above attack is used, then we are given w which is a convex combination of less than $k+1$ v_j's. We know the set of v_js can produce such w is unique (with non-zero coefficients). And the set of v_j's can be identified by using coding theory techniques. The Boneh-Franklin scheme introduced in the next session employs such techniques.

4.2 Boneh-Franklin Scheme

The Boneh-Franklin (BF) Scheme relies on the discrete log representation problem.

Definition 4.2.1 *When $y = \prod_{i=1}^{t} h_i^{\delta_i}$, we say $(\delta_1, \delta_2, \cdots, \delta_t)$ is a representation of y with respect to the basis $(h_1, h_2, \cdots, h_t)$.* □

Notice that if $d_1, d_2, \cdots, d_m$ are representations of y with respect to the same basis H, then any convex combination: $\overline{b} = \sum_{i=1}^{m} \alpha_i d_i$, where $\sum_{i=1}^{m} \alpha_i = 1$, is also a representation of y with respect to H.

Key generation and distribution: Let G_q be a group of order q, g a generator of G_q. G_q, g, q are public information. It is required that computing discrete logs in G_q is difficult.

For $i = 1, 2, \cdots, 2k$, the data supplier randomly chooses $r_i \in \mathcal{Z}_q$, and computes $h_i = g^{r_i}$ in G_q. The r_i's are kept in secret. Furthermore, the data supplier randomly chooses $\alpha_1, \alpha_2, \cdots, \alpha_{2k} \in \mathcal{Z}_q$ and computes $y = \prod_{i=1}^{2k} h_i^{\alpha_i} = g^{\sum_{j=1}^{2k} r_j \alpha_j}$. The α_i's are also kept secret by the data supplier. The encryption key is $(y, h_1, h_2, \cdots, h_{2k})$.

A collection Γ of n codewords of length $2k$ is made public. In order for tracing to work, Γ has to satisfy certain properties that will be discussed later. Let $\Gamma = \{\gamma^{(1)}, \gamma^{(2)}, \cdots, \gamma^{(n)}\}$. A private key is an element $\theta_i \in \mathcal{Z}_q$ such that $\theta_i \cdot \gamma^{(i)}$ is a representation of y with respect to the basis $H = (h_1, h_2, \cdots, h_{2k})$. User i's personal key, θ_i, is derived from the $\gamma^{(i)} = (\gamma_1, \gamma_2, \cdots, \gamma_{2k}) \in \Gamma$ by computing

$$\theta_i = \frac{\sum_{j=i}^{2k} r_j \alpha_j}{\sum_{j=i}^{2k} r_j \gamma_j} \qquad \mod q$$

Sometimes, we also refer to the personal key as the representation $d = \theta_i \cdot \gamma^{(i)}$, since a decoder requires a representation of y in order to decrypt.

Encryption and decryption To encrypt a session key $s \in G_q$, the data supplier picks a random element $a \in \mathcal{Z}_q$, and sets the enabling block to

$$(sy^a, h_1^a, h_2^a, \cdots, h_{2k}^a).$$

To obtain the session key, user i uses his secret key θ_i to compute

$$\begin{aligned}
& \frac{sy^a}{\left(\prod_{j=1}^{2k} (h_i^a)^{\gamma_j^{(i)}}\right)^{\theta_i}} \\
= & \frac{sy^a}{\prod_{j=1}^{2k} (g^{r_j})^{a\theta_i\gamma_j^{(i)}}} \\
= & \frac{sy^a}{g^{\sum_{j=1}^{2k} ar_j\theta_i\gamma_j^{(i)}}} \\
= & \frac{sy^a}{\left(g^{\theta_i \sum_{j=1}^{2k} r_j\gamma_j^{(i)}}\right)^a} \\
= & \frac{sy^a}{\left(g^{\sum_{j=1}^{2k} r_j\alpha_j}\right)^a} \\
= & \frac{sy^a}{y^a} \\
= & s
\end{aligned}$$

In fact any representation $(\delta_1, \delta_2, \cdots, \delta_{2k})$ of y with respect to H will allow the decryption of s, since

$$\prod_{j=1}^{2k} \left(h_j^a\right)^{\delta_i} = \left(\prod_{j=1}^{2k} h_j^{\delta_i}\right)^a = y^a.$$

Let us call the above encryption scheme the BF encryption scheme. It can be shown that this scheme is as secure as the ElGamal encryption scheme in G_q if k is in the order of $O(\log q)$.

Lemma 4.2.2 *Breaking the ElGamal encryption scheme reduces to breaking the BF encryption scheme in polynomial time in G_q, if $k = O(\log q)$.*

Proof: First let us state the problem of breaking the BF encryption scheme M1: given $(g, q, y, h_1, h_2, \cdots, h_{2k})$, $(sy^a, h_1^a, \cdots, h_{2k}^a)$, where $h_i = g^{r_i}$, $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$ for random secrets $r_i, \alpha_i \in \mathcal{Z}_q, i = 1, 2, \cdots, 2k$, find the value of s.

The problem of breaking ElGamal encryption scheme can be stated as M2: find s, given $(g, y), (sy^a, g^a)$, where $y = g^\alpha$, a and α are random secret elements in $\mathcal{Z}_q$.

Suppose we have an algorithm A1 to solve M1, and want to solve an instance of M2. We randomly genearate $r_2, r_3, \ldots r_{2k}$ and let $h_i = g^{r_i}$, $2 \leq i \leq 2k$. Further let $h_1 = g$. (Obviously there is a representation of y with respect to the basis $(h_1, h_2, \cdots, h_{2k})$. But A1 does not require the knowledge of any representation.) Then we compute $h_i^a = (g^a)^{r_i}$, $2 \leq i \leq 2k$. Now we have an instance of M1: $(g, q, y, h_1, h_2, \cdots, h_{2k})$, $(sy^a, h_1^a, \cdots, h_{2k}^a)$, where $h_1 = g, h_i = g^{r_i}$ for random $r_i, 2 \leq i \leq 2k$. Applying A1 will reveal the value of s.

$k = O(\log q)$ ensures that the reduction is taking polynomial time. □

More strongly, BF encryption is semantically secure against a passive adversary if the Decision Diffie-Hellman problem (DDH) is hard in G_q.

Definition 4.2.3 *The Decision Diffie-Hellman problem (DDH) is the task of deciding whether $y = g^{ab}$, given y, g^a, g^b, and g in a group G_q of order q, where g is a generator of G_q.*

A Decision Diffie-Hellman algorithm A for a group G_q of order q is an algorithm satisfying for some fixed $\alpha > 0$ and sufficiently large n,

$$\mid Pr[A(g, g^a, g^b, g^{ab}) = "true"] - Pr[A(g, g^a, g^b, g^c) = "true"] \mid > \frac{1}{n^\alpha}$$

where $c \neq ab \bmod p$.

We say that a group G_p *satisfies the DDH assumption if no polynomial time DDH algorithm exists for* G_p. □

Informally speaking, a DDH algorithm is an algorithm which solves the DDH problem with a probability better than $\frac{1}{2}$. If there is no efficient DDH algorithm in G_q,, then the BF encryption scheme is semantically secure (i.e. an adversary can learn nothing about the plaintext, except for its length, from ciphertext.).

Lemma 4.2.4 *The BF encryption scheme is semantically secure against a passive adversary in* G_q, *assuming that* G_q *satisfies the DDH assumption and* $k = O(\log p)$.

Proof: Suppose the BF encryption scheme is not semantically secure. Then given the public key $(y, h_1, h_2, \cdots, h_{2k})$, there exist two plaintexts $m_1, m_2 \in G_p$ such that given $(m_b y^a, h_1^a, h_2^a, \cdots, h_{2k}^a), b \in \{0,1\}$, an adversary can decide whether $b = 0$ or 1 with success rate $p_1 > \frac{1}{2}$. We are going to provide a DDH algorithm A.

Suppose we are given g, g^a, g^b, y and have to decide whether $y = g^{ab}$. Algorithm A first chooses random values $r_2, r_3, \cdots, r_{2k}$ from $\mathcal{Z}_q$ and computes $h_i = (g^a)^{r_i}$, $2 \leq i \leq 2k$. Then it sets $h_1 = g^a$. Let the public key by $(y, h_1, h_2, \cdots, h_{2k})$.

Now A picks b randomly from $\{0,1\}$ and constructs $C = (m_b g^b, y, y^{r_2}, \cdots, y^{r_{2k}})$. C is given to the adversary. The adversary outputs $b' \in \{0,1\}$. If $b = b'$, then A output "true"; otherwise A outputs "false".

If $y = g^{ab}$, then $y = (g^a)^b = h_1^b$, and $y^{r_i} = ((g^a)^{r_i})^b = h_i^b$, $2 \leq i \leq 2k$. Thus C is a valid encryption of m_b. Then $b = b'$ with probability p_1. If $y \neq g^{ab}$, then C is a encryption of a random message, and hence $b = b'$ with probability $\frac{1}{2}$.

Therefore, the probability that $A(g, g^a, g^b, g^{ab}) = "true"$ is p_1, and the probability that $A(g, g^a, g^b, g^c) = "true"$ is $\frac{1}{2}$, where $c \neq ab \bmod p$. Since $p_1 > \frac{1}{2}$, we have a DDH algorithm. Clearly this algorithm is polynomial if $k = O(\log p)$. □

However, note that the DDH assumption is a very strong assumption. There are groups where the computational Diffie-Hellman problem (given g, g^a, g^b, compute g^{ab}) is believed to be hard, but the DDH assumption does not hold.

An example where this might be true is $\mathcal{Z}_p^*$ for a prime p and generator g, The computational Diffie-Hellman problem is believed to be intractable in $\mathcal{Z}_p^*$. But, given g^a, g^b, one can easily compute the Legendre symbol of g^{ab}. This gives an immediate method to decide whether $y = g^{ab}$ given y, g^a, g^b with a success rate much better than $\frac{1}{2}$.

The following are some groups in which the DDH assumption holds:

1. Let $p = 2p'+1$ where p and p' are primes. Let Q_p be the subgroup of quadratic residues in $\mathcal{Z}_p$. It is a cyclic group of prime order.

2. Let $p = aq + 1$ where both p and q are prime and $q > p^{\frac{1}{10}}$. Let $Q_{p,q}$ be the subgroup of $\mathcal{Z}_p$ of order q.

3. Let $N = pq$ where p, q, $\frac{p-1}{2}$, $\frac{q-1}{2}$ are primes. Let T be the cyclic subgroup of order $(p-1)(q-1)$.

4. Let p be a prime and $E_{a,b}/\mathcal{F}_p$ be an elliptic curve where $\|E_{a,b}\|$ is a prime.

5. Let p be a prime and J be a Jacobian of a hyper elliptic curve over $\mathcal{F}_p$ with a prime number of reduced divisors.

Tracing algorithm: The above two lemmas show that the BF encryption scheme is "secure" if the adversary is given the encryption key and the ciphertext

only. What if the adversary has a set of decryption keys in his hand? Can he produce a different key which enbles the decryption without being traced? In order to recover the session key s, one has to obtain the value of y^a. An adversary can either compute a or find a representation of y with respect to basis H. It can be easily shown that if one can obtain a from a set of representations of y with respect to H then he can compute any discrete log problem in the group. So it is safe to assume that the adversary cannot compute a.

Can an adversary compute a presentation of y with respect to H? Let $D = \{d_1, d_2, \cdots, d_m\}$ be a set of known encryption keys. Here we think encryption keys as representations of y with respect to basis H. Recall that any convex combination of vectors in D is also a representation of y with respect to H. From D, an adversary can easily construct a convex combination of vectors in D. Boneh and Franklin suggested that these convex combinations are the only new representations of y with respect to H that can be efficiently constructed from D. In [3], Boneh and Franklin attempted to show that, if one can construct a new representation from D which is not a convex combination of vectors in D, then he can compute discrete log in the group. Although the proof was incorrect, it seems valid to assume the following:

Conjecture 4.2.5 *Given $y, H = \{h_1, h_2, \cdots, h_{2k}\}$, and D a set of representations of y with respect to H in G_q, the only new representation that can be constructed efficiently (i.e., in polynomial time) are the set of convex combinations of vectors in D.*

The tracing algorithm relies on the assumption that Conjecture 4.2.5 is true. Suppose a decryption key d (a representation of y with respect to H) is captured from a pirate decoder built by at most k traitors. If the conjecture is true, then d

must be a convex combination of a set of at most k encryption keys. But notice that a user's decryption key is a scalar multiple of a codeword in Γ. Then d is a linear combination of at most k codewords in Γ. Let $D \subseteq \Gamma$, $\mid D \mid \leq k$. We say d can be created by D if d can be written as a linear combination of codewords in D. We require that the intersection of all such D's not empty for any private key d. All members of the intersections are traced.

Let C be a linear code over G_q of length n, with dimension $l-2k$, and Hamming distance $2k+1$. Let B be a parity check matrix of C. Then B has dimension $2k \times n$, and any $2k$ columns of B are linearly independent. Let Γ be the set of columns of B. Now suppose the key d captured is a linear combination of $k' \leq k$ vectors in Γ. Then we can write $d = Bw$, where $w \in G_q^n$ with hamming weight at most k. Notice that such w is unique, otherwise we have a set of at most $2k$ linear dependent columns of B. Thus, d can be expressed as a linear combination of at most k codewords in Γ uniquely, given that all coefficients are non-zero. Let D denote the set of such codewords.

Now consider a vector $v \in G_q^n$ such that $Bv = d$. Notice that v can be found easily. Then $B(v-w) = 0$. So $r = v - w$ is a codeword in C and w is an error vector of weight at most k. Since C has dimension $2k+1$, it can correct any error of weight at most k. Therefore w can be found by decoding v to the nearest codeword in C. Hence, we can find D and trace back to the users to whom the codewords in D are assigned. Notice that in all previous tracing schemes, only one of the traitors are identified. In the BF scheme all traitors whose key was used to construct the pirate key can be identified.

In [3], BCH codes are used, and $O(n \log n \log\log n)$ group operations are required in the decoding. In practice, any linear code over G_q of length n, dimension $l-2k$, and Hamming distance $2k+1$ can be used, as long as the decoding algorithm

of the code is efficient.

Above, we assumed a key has been captured from a pirate decoder. But is it possible to recover the key based on how the decoder performs on different input ciphertexts?

Single-key Pirates: First, we consider the case in which the pirate decoder has only one representation d of y and always uses d in decoding. This is called *single-key pirate* since only a single decryption key is embedded in the pirate decoder. We would like to extract this key from the decoder while treating the decoder as a black box.

Let us consider $C = (S, h_1^{z_1}, \cdots, h_{2k}^{z_{2k}})$, where not all of $z_1, z_2, \cdots, z_{2k}$ have the same value. Then C is not a valid ciphertext in the encryption scheme, as the h_i's are raised to different powers. The basic idea of tracing is to observe the decoder's behavior on invalid ciphertexts. The following lemma shows that the decoder cannot distinguish invalid ciphertexts from valid ciphertexts assuming the difficulty of DDH in G_q. Hence, the decoder will always output

$$A = \frac{S}{\prod_{i=1}^{2k} (h_1^{z_1})^{\delta_i}}$$

where $d = (\delta_1, \delta_2, \cdots, \delta_{2k})$ is the single key possessed by the decoder, just as it does on an input of valid ciphertexts.

Lemma 4.2.6 *Suppose there is an adversary that can distinguish invalid ciphertexts in which basis elements are raised to different powers, from valid ciphertexts of the BF encryption scheme in group G_q with a non-negligible probability. Then the adversary can also solve the DDH problem in G_q, with the same probability. The reduction is in polynomial time, if $k = O(\log q)$.*

Proof: Recall that solving the DDH problem involves: given input (g, g^a, g^b, x), decide whether $x = g^{ab}$, where g is a generator of G_q.

Suppose there is an adversary that can distinguish invalid ciphertexts from valid ciphertexts with a non-negligible probability. First, we construct the public key of a BF encryption scheme as follows: pick random $r_i, s_i \in \mathcal{Z}_q$ and set $h_i = g^{r_i} g^{as_i}, 1 \leq i \leq 2k$; pick random $\alpha_1, \alpha_2, \cdots, \alpha_{2k} \in \mathcal{Z}_q$ and set $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$. The public key is $(y, h_1, \cdots, h_{2k})$.

Given an instance of DDH problem (g, g^a, g^b, x), we construct

$$C = (S, \left(g^b\right)^{r_1} x^{s_1}, \left(g^b\right)^{r_2} x^{s_2}, \cdots, \left(g^b\right)^{r_{2k}} x^{s_{2k}})$$

where S is a random element in G_q, and give C to the adversary.

Notice that, if $x = g^{ab}$, then

$$\begin{aligned} C &= (S, g^{br_1} g^{abs_1}, g^{br_2} x^{abs_2}, \cdots, g^{br_{2k}} x^{abs_{2k}}) \\ &= (S, (g^{r_1} g^{as_1})^b, (g^{r_2} g^{as_2})^b, \cdots, (g^{r_{2k}} g^{as_{2k}})^b) \\ &= (S, h_1^b, h_2^b, \cdots, h_{2k}^b). \end{aligned}$$

So, C is a valid ciphertext for plaintext $\frac{S}{y^b}$. And if $x \neq g^{ab}$, C is an invalid ciphertext.

Thus, if the adversary can decide correctly whether C is a valid ciphertext with a non-negligible probability, then he can certainly decide if $x = g^{ab}$ successfully with the same probability. The reduction is in polynomial time given that $k = O(\log q)$. □

Therefore feeding a pirate decoder an invalid ciphertext,

$$C = (S, h_1^{z_1}, \cdots, h_{2k}^{z_{2k}}),$$

we can assume that the decoder always outputs

$$A = \frac{S}{\prod_{i=i}^{2k} (h_i^{z_i})^{\delta_i}},$$

where $(\delta_1, \delta_2, \cdots, \delta_{2k})$ is the representation of y the pirate decoder has. Then $\prod_{i=i}^{2k} (h_i^{z_{2k}})^{\delta_i} = \frac{S}{A}$ can be easily computed. The tracer's task is to determine $(\delta_1, \delta_2, \cdots, \delta_{2k})$. Let $\overline{z}_1, \overline{z}_2, \cdots, \overline{z}_{2k}$ be $2k$ linearly independent vectors in $\mathcal{Z}_q^{2k}$, and let $\overline{z}_i = (z_{i,1}, z_{i,2}, \cdots, z_{i,2k})$. Then after feeding $C_i = (S_i, h_1^{z_{i,1}}, \cdots, h_{2k}^{z_{i,2k}}), 1 \leq i \leq 2k$, to the decoder, we are able to obtain $2k$ equations, $\prod_{i=1}^{2k} \left(h_i^{z_{j,i}}\right)^{\delta_i} = \frac{S_j}{A_j}, 1 \leq j \leq k$, where A_j is the decoder's output for ciphertext C_j. Then $h_i^{\delta_i}, 1 \leq i \leq 2k$ can be determined from the set of $2k$ equations. Recall that the data supplier knows the values of r_i such that $h_i = g^{r_i}, 1 \leq i \leq 2k$. Thus the tracer can compute $g_i^{\delta_i}$, for $1 \leq i \leq 2k$. Boneh and Franklin suggested to recover δ_i from $g_i^{\delta_i}$ by using trapdoors of the discrete log introduced in [11] and [12].

Here we are going to briefly describe how Paillier's algorithm in [12] works. What Paillier presented is a public key cryptosystem that relies on the one-way function $f(x, y) = g^x y^N mod N^2$, where $x \in \mathcal{Z}_N$, and $y \in \mathcal{Z}_N^*$. Let $N = p \times q$, where p, q are two large primes. Let $g \in \mathcal{Z}_{N^2}$ be an element of order N. The public key consists of (g), and the private key consists of (g, p, q). To encrypt a message $m < N$, Alice chooses a random $r < N$, and computes $c = g^m r^N$, where g is the public key of Bob. Upon receiving the message c, Bob can retrieve m by computing $\frac{L(c^\lambda \bmod N^2)}{L(g^\lambda \bmod N^2)} \bmod n$, where $L(u) = \frac{u-1}{N}$, and λ is Carmichael's function on N, i,e. $\lambda = lcm(p-1, q-1)$. The proof of soundness of the encryption scheme is beyond the scope of this thesis. Here we only describe how the system can be used to compute discrete logs in BF scheme.

Suppose the BF encryptions are done in $\mathcal{Z}_{N^2}^*$, where N is a product of two large primes p,q, and g has order N. Then given g^{δ_i}, the tracer can compute $\delta_i = \frac{L\left(\left(g^{\delta_i}\right)^\lambda \bmod N^2\right)}{L(g^\lambda \bmod N^2)}$, since here $r = 1$, given that the tracer knows the factorization of N. Thus by employing the trapdoor discrete log scheme in [12], a tracer can obtain the key in a pirate decoder. Then by using the tracing algorithm in the

previous section, we can identify all the traitors.

Arbitrary Pirates: Of course, pirates do not have to restrict the decoder to contain only one single representation of y. Instead, the decoder can contain up to k representations belonging to the traitors, and then arbitrarily choose a convex combination of these representations in each decoding session. Boneh and Franklin did not provide an efficient (polynomial time) tracing algorithm for this scenario. However they did suggest an algorithm called *black box confirmation.*

The idea is to enumerate all $\begin{pmatrix} n \\ k \end{pmatrix}$ subsets of users of size k, and test whether the subset is a superset of the set of traitors. Let $d_1, d_2, \cdots, d_k$ be the set of keys belonging to a subset T of size k. To test whether the set of traitors is a subset of T, the tracer queries the decoder with an invalid ciphertext, $C = (S, g^{z_1}, \cdots, g^{z_{2k}})$, such that $d_i \cdot z = w, 1 \leq i \leq k$, where $z = (z_1, \cdots, z_{2k})$, and w is a random element in $\mathcal{Z}_q$. The pirate decoder would output, $A = \frac{S}{\prod_{i=i}^{2k}(g^{z_i})^{\delta_i}}$. If T is indeed a superset of traitors, $\delta = (\delta_1, \delta_2, \cdots, \delta_{2k})$ would be a convex combination of a subset of $d_1, d_2, \cdots, d_k$, and $\delta \cdot z = w$, which implies $A = \frac{S}{g^w}$. Confidence in this test can be increased by making multiple queries, where each query is made independently using different S, z, and w. If for a coalition T, the pirate always outputs $A = \frac{S}{g^w}$, then the pirate must possess a subset of keys belonging to T. The intersection of all such T's is the set of traitors.

Note this algorithm does not require trapdoors of discrete log. It does not even require the decoding operation of the linear code mentioned above. But, the black box confirmation algorithm must test all $\begin{pmatrix} n \\ k \end{pmatrix}$ subsets of users of size k, and clearly it is not a polynomial time algorithm.

Efficiency Measurement: The data supplier has to store Γ which is an $n \times 2k$ matrix of elements in $\mathcal{Z}_q$. In addition, he also stores $y, r_1, r_2, \cdots, r_{2k}, h_1, h_2, \cdots, h_{2k}$. Note, the data supplier does not need to store $\alpha_1, \alpha_2, \cdots, \alpha_{2k}$ once all the personal keys are generated. So, the data supplier stores roughly $O(nk)$ elements of $\mathcal{Z}_q$.

For encryption, the data supplier computes $y^a, h_1^a, h_2^a, \cdots, h_{2k}^a$. So there are $2k+1$ exponentiations, each requiring $O(\log q)$ group multiplications. Thus encryption requires $O(k \log q)$ group multiplications, about $O(k \log^3 q)$ bit operations. The size of the enbling block is $(2k+1) \log q$.

Each user (i.e., his decoder) stores a representation $d = (\delta_1, \delta_2, \cdots, \delta_{2k})$ of y. d consists $2k$ elements of $\mathcal{Z}_q$ To decrypt a session key, a decoder has to compute $u_i = (h_i^a)^{\delta_i}$, $1 \leq i \leq 2k$, $U = \prod_{i=1}^{2k} u_i, U^{-1}$, and finally $sy^a U^{-1}$. Each exponentiation requires $O(\log q)$ group multiplications. So computing $u_i, 1 \leq i \leq 2k$ requires $O(k \log q)$ multiplications. Computing U needs another $2k-1$ multiplications. Inverting U can be done in $O(\log q)$ multiplications. Lastly, one more multiplications is needed to compute s. So the decryption requires $O(k \log q)$ group multiplications.

Given a key captured from a pirate decoder, the tracing algorithm can identify all owners of the representations that are used to construct the pirate key, using $O(n \log n \log \log n)$ group multiplications. To perform black box tracing on a single key pirate, the tracer has to make $2k$ queries to the pirate decoder. This requires $O(k^2 \log q)$ multiplications, as each query is equivalent to performing a decryption. Solving $2k$ equations to obtain $h_i^{\delta_i}, 1 \leq i \leq 2k$, requires $2k$ additions in $\mathcal{Z}_q$ which is relatively cheap, and up to k exponentiations and k inversions in G_q. So this step requries $O(k \log q)$ multiplications. To obtain g^{δ_i}, $1 \leq i \leq 2k$, another $2k$ exponentiations ($O(k \log q)$ multiplications) are needed. Finally to compute δ_i, $1 \leq i \leq 2k$, using Paillier's algorithm requires 2 exponentiations in G_q and a constant number of inversions and multiplications in $\mathcal{Z}_q$ for each δ_i. So the final

step needs roughly $O(k \log q)$ multiplications in G_q. In total, to perform a black box tracing on a single key pirate decoder, $O(k^2 \log q)$ multiplications in G_q which is equivalent to $O(k^2 \log^3 q)$ bit operations are required.

Chapter 5

Attempts of Public Key Traceability Schemes

The schemes presented in the previous chapter are asymmetric in the sense that the encryption key and decryption key are different. But since the data supplier knows each user's decryption key, he can still frame any user at his will. There have been attempts to construct a truly asymmetric key traceability scheme in which a user's decryption key is kept secret. We will call these schemes *public key traceability schemes*. In this chapter we present the attempts by Pfitzmann [13], and Kurosawa and Desmedt [10]. Then we derive a public scheme from the Boneh-Franklin traceability scheme discussed in Section 4.2.

5.1 Pfitzmann's Schemes

In [13], Pfitzmann proposed three asymmetric schemes in which the data supplier and users do not share any secrets. In this model there is an entity called *judge* to

whom evidence should be submitted in order to prove the accused user is indeed a traitor. The first scheme employs a public key cryptosystem to encrypt/decrypt session keys. The scheme itself is very simple. However, the enabling block size of the scheme is linear in terms of the number of users, which is very inefficient. The remaining two schemes require an underlying symmetric traitor tracing scheme. The basic idea is to have a trusted third party (TTP) to assign the personal keys so that the data supplier does not know who gets which personal key. Unfortunately all these schemes are flawed.

Scheme A: with linear-sized enabling blocks. In this scheme, each user has two pairs of keys from a public key cryptosystem: e_i, d_i for encryption/decryption, and e'_i, d'_i, for signature generation/verification. Let E, D be the encryption and decryption function of the system respectively. So to encrypt a message m for user i, one computes $c = E_{e_i}(m)$, and user i computes $D_{d_i}(c)$ to retrieve m. Suppose E and D are also used for signature verification and generation as well, eg. as with RSA. Thus, to sign a message m, user i computes $sig_i = D_{d'_i}(H(m))$, to verify the signature, one computes $E_{e'_i}(sig_i)$ and compare it with $H(m)$, where H is a suitable cryptographic has function such as SHA-1.

Each user gives the data supplier his two public keys e_i, e'_i and his signature $sig_i = D_{d'_i}(H(d_i))$ on his decryption key d_i, using signing key d'_i. The data supplier encrypts a session key s using keys e_i, $1 \leq i \leq n$. This leads to an enabling block $\mathcal{B}_e = \|_{i=1}^{n} E_{e_i}(s)$ of size $O(n)$. Each user can decrypt s by using his private key d_i, $s = D_{d_i}(E_{e_i}(s))$.

Suppose a pirate decoder is captured. Pfitzmann assumed that the decoder's decryption key can be obtained, and no black box tracing was provided. The data supplier finds u such that sig_u is a valid signature of d, by comparing $H(d)$ with $E_{e'_i}(sig_i)$, $1 \leq i \leq n$. He can then submit d, e'_u, sig_u to a judge as the proof that u is

a traitor. Since the data supplier does not know the private encryption key of any user, it is unlikely that he can find a key d which has the same signature as a user's private key signed by the user's signing key. However a user can easily submit a false signature of his private encryption key, so that the data supplier would not find a match between d and the message produced by verifying the signature submitted by the user. The data supplier has no way to verify the signature without knowledge of the user's private encryption key.

However, the data supplier can perform the tracing in a similar way as in the CFNP schemes. For each i, $1 \leq i \leq n$, replace $E_{e_j}(s)$, $j \neq i$ in the enabling block by some random data. If the decoder can still decode then it must have key d_i, and user i is a traitor. With this modification we don't even require each user to have a pair of keys for signature generation/verification. But, notice that if the decoder has more than one key, it can detect tracing by decrypting s using two or more keys. So strictly speaking, the scheme can trace a traitor coalition only of size 1. But if we assume the detection is infeasible, maybe due to the limitation of a decoder's computing power, the scheme can trace a coaltion of any size. Nevertheless the enabling block size makes the scheme very inefficient. And there is no elegant proof that can be submitted to the judge, expect for performing the tracing in front of him.

Scheme B: Transform a symmetric scheme to a public-key scheme: This scheme is obtained from a symmetric scheme by using a cryptographic primitive called a *secure 2-party protocol*. Such a protocol achieves the following goals: two parties have secret input x_1, x_2 respectively. Both of them want to know $g(x_1, x_2)$, where g is a function known to both of them. However, party one should not gain any information on x_2, nor party 2 on x_1, except for what is revealed by the value of $g(x_1, x_2)$.

Suppose we have a symmetric tracing scheme. Let $P = \{P_1, P_2, \cdots, P_n\}$ denote the set of personal keys. Instead of assigning P_i to user i, we want to assign the keys in a way such that the data supplier does not know the identity of the owner of any specific personal key.

User i is going to secretly select an id, id_i (all users should have distinct ids). Then user i inputs his secret id_i to the 2-party protocol we call PP. The data supplier inputs the set of personal keys to PP. The protocol returns a personal key P_{id_i} to user i and uses a known one way hash function f to compute $h_i = f(id_i)$ which is returned to the data supplier who will keep a list of h_i's.

To trace a traitor, the tracing algorithm of the symmetric scheme was first run. The algorithm does not trace an actual traitor, but identifies a personal key, say P_t which belongs to a traitor. The data supplier then compares $f(t)$ with h_i, $1 \leq i \leq n$. If $f(t) = h_c$, c is a traitor.

We notice that this scheme can be broken by the data supplier who can easily compute $f(i)$, $1 \leq i \leq n$. Then he does a simple comparison between $f(i)$'s and h_i's to find out who is the actual owner of each personal key. Thus this scheme does not provide any more protection on users from being framed by the data supplier than the underlying symmetric tracing scheme. The third scheme is very similar to this scheme, and hence suffers the same problem.

5.2 Kurosawa and Desmedt's Scheme

Kurosawa and Desmedt suggested a public key tracing scheme in [10]. This scheme was derived from the asymmetric scheme discussed in section 4.1. Although the asymmetric scheme has been broken, here we are more interested in the way the

authors transform a non-public scheme to a public scheme.

In the new scheme, there are c entities called agents: A_1, A_2, $\cdots$, A_c. Each agent A_j generates a random polynomial

$$f_j(x) = a_{j,0} + a_{j,1}x + a_{j,2}x^2 + \cdots + a_{j,k}x^k$$

$a_{j,0}$, $a_{j,1}$, $\cdots$, $a_{j,k} \in \mathcal{Z}_q$. Let $f(x) = \sum_{j=1}^{c} f_j(x)$. Each agent A_j also computes $f_j(i)$, $1 \leq i \leq n$, and distributes $f_j(i)$'s to user i through a secure channel. User i computes $f(i) = \sum_{j=1}^{c} f_j(i)$ and uses it as his private key. The data supplier receives $y_{j,i} = g^{a_{j,i}}$, $0 \leq i \leq k$ from A_j through a secure channel. He then computes $y_i = \prod_{j=1}^{c} y_{j,i}$. The public key is $(y_0, y_1, \cdots, y_k)$. Encryption and decryption works in the same way as in the non-public scheme discussed in section 4.1.

Notice that in the non-public scheme, the data supplier is the one who generates $f(x)$ and computes each user's private key. But, here the c agents act together as the key generator/distributor of the scheme.

We know that a coalition of more than one traitor can build a decoder without being traced. But, here we are only concerned with whether the data supplier can obtain any user's private key from the values $y_{j,i}$, $1 \leq i \leq c$, $1 \leq j \leq k$. It can be easily shown that if the data supplier can obtain $f(i)$ for any $i \in \{1, 2, \cdots, n\}$, then he can compute discrete logs in the group $S =< g >$. Thus the data supplier can not frame any user, if the discrete log problem is hard in S. Furthermore, unless all c agents collude, or at least $k+1$ users collaborate, $f(x)$ cannot be computed. Thus the transformation works in the sense that it does prevent the data supplier from framing users.

5.3 Boneh-Franklin public scheme

Let us consider what features the schemes in the previous two sections have in common. Notice that in both cases, there is a TTP which does key distribution or/and generation, which was done by the data supplier in the non-public schemes. In Pfitzmann's scheme, the 2-party protocol serves as the TTP, and in Kurosawa-Desmedt's public scheme, the c agents act together as a TTP. In all the non-public schemes presented, the data supplier does both key generation and distribution. In order to convert a non-public scheme to a public scheme, we need to relieve the data supplier from both duties. In Pfitzmann's scheme, although the 2-party protocol performs the key distribution, the data supplier still generates all personal keys, which leads to an easy attack on the scheme.

Here we present a method to derive a public scheme from Boneh-Franklin's scheme which was discussed in Section 4.2. We assume the scheme does not have the trapdoor which makes it easy to compute discrete logs in the underlying group G_q.

Recall that in the BF scheme, the data supplier randomly chooses α_1, α_2, $\cdots$, $\alpha_{2k} \in \mathcal{Z}_q$ and computes $y = \prod_{i=1}^{2k} h_i^{\alpha_i}$. In the new scheme this step is performed by a TTP. The TTP also has the generator of G_q: g, and the values of r_i, $1 \leq i \leq 2k$, such that $h_i = g^{r_i}$. These values enable the TTP to compute each user's private key as the data supplier does in the original BF scheme.

In the new scheme, the data supplier does not know any representation of y. In order to frame a user, the data supplier first has to construct a decoder which has a representation of y as its decryption key. It is easy to show that if the data supplier can construct a representation of y with respect to h_1, h_2, $\cdots$, h_{2k} from y and h_1, h_2, $\cdots$, h_{2k} alone, then he can solve discrete logs in G_q. We require a BF

scheme without the trapdoor so the data supplier can not frame any user.

As in Kurosawa-Desmedt's public scheme, we can have several agents to act together as a TTP, so that a user can be framed only if all agents collaborate. Each agent A_j randomly generates $\alpha_{j,1}, \alpha_{j,2}, \cdots, \alpha_{j,2k} \in \mathcal{Z}_q$ and computes $y_j = \prod_{i=i}^{2k} h_i^{\alpha_{j,i}}$ and

$$\theta_{i,j} = \frac{\sum_{l=1}^{2k} r_l \alpha_{j,k}}{\sum_{l=1}^{2k} r_l \delta_l^{(i)}}.$$

y_j is made public and $\theta_{i,j}$ is given to user i through a secure channel.

The data supplier and each user computes

$$\begin{aligned} y &= \prod_{j=1}^{c} y_j \\ &= \prod_{j=1}^{c} \prod_{i=1}^{2k} h_i^{\alpha_{j,i}} \\ &= \prod_{j=1}^{c} g^{\sum_{i=1}^{2k} r_i \alpha_{j,i}} \\ &= g^{\sum_{j=1}^{c} \sum_{i=1}^{2k} r_i \alpha_{j,i}}. \end{aligned}$$

Each user i computes his personal key

$$\begin{aligned} \theta_i &= \sum_{j=1}^{c} \theta_{i,j} \\ &= \frac{\sum_{j=1}^{c} \sum_{l=1}^{2k} r_l \alpha_{j,k}}{\sum_{l=1}^{2k} r_l \delta_l^{(i)}}. \end{aligned}$$

Then

$$\begin{aligned} \prod_{l=l}^{2k} (h_l^{\delta_l^{(i)}})^{\theta_i} &= g^{\theta_i \sum_{l=l}^{2k} r_l^{\delta_l^{(i)}}} \\ &= g^{\sum_{j=1}^{c} \sum_{l=1}^{2k} r_l \alpha_{j,l}} \\ &= y. \end{aligned}$$

Therefore $\theta_i = (\delta_1^{(i)}, \delta_2^{(i)}, \cdots, \delta_{2k}^{(i)})$ is a presentation of y with respect to basis: $(h_1, h_2, \cdots, h_{2k})$. Encryption and decryption works in the same way as in the original BF scheme.

All c agents have to collaborate in order to obtain any representation of y. So, we have derived a public scheme from a non-public BF scheme. Since we assume the BF scheme has no trapdoors to help computing discrete log in G_q, the new scheme does not support black box tracing. But given a key found in a pirate decoder, the same tracing algorithm in original BF scheme can be used to trace all the traitors whose private key is used in constructing the key captured in the pirate decoder.

Chapter 6

Other Traceability Schemes

In this chapter, we present a couple of other traceability schemes that have been suggested. Chameleon is a stream cipher designed by Anderson and Manifavas [2] to allow traitor tracing. Digital signet was proposed by Dwork, Lotspiech and Noar [7], whose goal was to motivate users to be self-policing.

6.1 Chameleon

In 1996, Ross Anderson and Charalamps Manifavas introduced a stream cipher called Chameleon which allows traitor tracing. The main idea is to give each user a slightly different decryption key that had the effect of producing slightly different plaintexts.

The scheme is built upon a pseudorandom number generator (PRNG), or any block cipher in output feedback mode. A fingerprinting scheme is required. Fingerprinting involves uniquely marking and registering each copy of a piece of data so that, given a copy of the data, the distributor is able to trace it back to its owner.

A mark is a position which can be in one of q states in the data. A fingerprint is a collection of marks, and can be thought as a codeword of length L over an alphabet of size q, L is the number of marks in the data. An (N, L) fingerprinting scheme is a set of N fingerprints of length L. However a coalition of users may detect some of the marks, namely the ones in which their copies differ. They can then change these marks arbitrarily hoping to mask their identities. We call an (N, L) fingerprinting scheme is *totally k-secure* if there exists a tracing algorithm A satisfying the following condition: if a coalition T of size at most k generates a copy x by changing the marks where their copies differ then $A(x) \in T$. If we want to make Chameleon a (k, n) traitor tracing scheme, we would require a totally k-secure fingerprinting scheme which can produce n copies of data. For a detailed discussion of fingerprinting schemes, see [4]. In [2], the authors randomly generated 4000 marks in a piece of data of size 512KB, to achieve a $(4, n)$ traitor tracing scheme.

In Chameleon, instead of planting fingerprints into the content, each user's personal key is embedded with a unique fingerprint so that when a decoder decrypts the ciphertext, the fingerprint would be generated in the output. Notice that here only one copy of the ciphertext is broadcasted, while an ordinary fingerprinting scheme would require several slightly different ciphertexts and each of them is distributed to an unique user.

Key Generation/Distribution: Each user should have two keys: The first key, which we call A, is the same for all users. A is used as a seed for the PRNG, and can be broadcasted publicly. Each user i also gets B_i which is a table of l r-bit words - a total of lr bits of data. In the example in [2] B_i is a table of 2^{16} 64-bit words - i.e., 512 KB of data. Each B_i is obtained by introducing a unique fingerprint to the master key B of the same size owned by the data supplier. B_i is never changed after it is assigned.

Encryption/Decryption: To encrypt a r-bit content, the data supplier takes the next $\log l$ (assume l is a power of 2) bits of output of the PRNG and uses it to select one word of B. How the word is selected is public information. The selected word is then XORed with the content to produce a ciphertext. In the same way, the ciphertext can be decrypted. But each user uses his personal B_i instead of B in decryption. Notice that any marking in the chosen word of B_i is inherited by the output.

Tracing: The tracing relies on the tracing algorithm of the underlying fingerprinting scheme. Suppose a coalition T of at most k traitors collaborated and built a pirate decoder. The B key in the pirate decoder is generated by changing the marks at which the coalition members' B keys differ. Since the B keys are produced by a totally k-secure fingerprinting scheme, the tracing algorithm of the scheme is able to trace one member of the T given the marks in the B key of the pirate decoder. Suppose a pirate decoder was captured. The data supplier can use the decoder to produce a content embedded with the fingerprint in the decoder's B key. The tracing algorithm of the fingerprinting scheme should be able to trace one member of the coalition.

In the example used in [2], 4000 marks are generated at random in each user's B key. It is expected that the example scheme can trace a traitor if there are at most four traitors.

In general, the security level of the tracing scheme relies on the underlying fingerprinting scheme. Besides the fingerprinting scheme, the rest of Chameleon is very simple, as only a PRNG is needed. It does not even require an encryption scheme to encrypt the content, as content is XORed with the words chosen from the B key. The scheme also supports a straightforward black box tracing algorithm as long as the tracing algorithm of the fingerprinting scheme is available.

Unfortunately, in [4], it was shown that, for $c \geq 2$ and $n \geq 3$, there are no totally c-secure (n, l) fingerprinting schemes. This forces us to settle for schemes which trace a traitor with a small error probability ϵ. Such schemes are called k-secure schemes with ϵ-error. The following two theorems were proved in [4].

Theorem 6.1.1 *For $n \geq 3$ and $\epsilon > 0$, there exists an $(n, 2n^2 \log(2n/\epsilon)$ fingerprinting scheme which is n-secure with ϵ-error.*

Theorem 6.1.2 *For $n \geq 3$ and $\epsilon > 0$, there exists an $(n, k^4 \log(n/\epsilon) \log(1/\epsilon))$ fingerprinting scheme which is k-secure with ϵ-error.*

Notice that the length of the fingerprinting scheme in Theorem 6.2 is roughly the same as the size of the base keys in the one level open CFNP traceability schemes. Thus Chameleon schemes require very large personal keys. On the other hand, Chameleon does not require an enabling block as long as the data supplier and users agree on the A key which can be broadcasted to each user. As well, the speed of Chameleon is very fast.

6.2 Digital Signet

In 1996, digital signet was proposed by Dwork, Lotspiech and Naor [7]. In this scheme, each session key is encrypted using each user's personal key, which includes some sensitive private information (such as credit card number) related to the owner. The scheme tried to force the owner either to reveal the sensitive information or to redistribute the session key which is as long as the content, i.e., redistributing it would require the same bandwidth as the content distribution channel.

Each user i must submit sensitive information u_i to the data supplier so that the DS can compute a signet α_i for user i using a *authentication funtion*. Using u_i and α_i user i is able to obtain session key s using a public *extrication function* h, $s = h(u_i, \alpha_i)$. s is required to be much longer in length than u_i and α_i.

The authors require f to be *incompressible*. A function $h(t)$ is incompressible if $\|h(x)\| >> \|x\|$ for all x and in order for A to communicate $h(x)$ to B in $o(\|x\|)$ bits, A must reveal x. In order words, there is no feasible computable short message that allows B to learn $h(x)$ while simultaneously protecting x.

The incompressible function chosen by the authors is very similar to the encryption/ decryption function used in Kurosawa and Desmedt's (KD) traceability scheme in Section 4.1. Let G be a group of order p, let q be a prime such that $q \mid p-1$. Let g be an element of order q in G.

As in KD scheme, the data supplier chooses a random function:

$$h(x) = a - (b_1 x + b_2 x^2 + \cdots + b_k x^k).$$

and $Z = (g, y_1 = g^{b_1}, y_2 = g^{b_2}, \cdots, y_k = g^{b_k})$ is made public. g^a comprises one block of session key. The authentication function takes u_i and computes $\alpha_i = f(u_i)$, where u_i is the personal information submitted by user i. Then α_i is returned to user i as his signet.

Using u_i and α_i, user i is able to compute g^a:

$$g^{\alpha_i} \prod_{j=1}^{k} y_i^{u_i^j} = g^{a - \sum_{j=1}^{k} b_j u_i^j + \sum_{j=1}^{k} b_j u_i^j} = g^a$$

The full session key is obtained by choosing g_1, g_2, $\cdots$, and concatenating g_1^a, g_2^a, $\cdots$. So

$$Z_j = (g_j, y_{1,j} = g_j^{b_1}, y_{2,j} = g_j^{b_2}, \cdots, y_{k,j} = g_j^{b_k}), j = 1, 2, \cdots$$

are also required to be made public. This scheme seems highly inefficient: to encrypt one block of session key g_i^a, the amount of data needed to be made public has size $(k+1)\|g_i^a\|$. However in this scheme, the authors attempted to prevent the traitors from redistributing the session keys, while this problem was not addressed in any previously discussed schemes.

Clearly, the attack on SD scheme can be also employed to attack the signet scheme. A coalition T of traitors can publish a convex combination of $f(u_i)$, u_i, u_i^2, $\cdots$, u_i^k, $i \in T$. The published information would allow the decoding of the session keys, but protect the u_i's from being exposed. In an attempt to reduce the computation cost and thwart the attack, the authors proposed a modification of the above scheme.

Let $m = 2k$, choose random $B = \{b_1, b_2, \cdots, b_m\}$, and make

$$Z = (g, y_1 = g^{b_1}, y_2 = g^{b_2}, \cdots, y_m = g^{b_m})$$

public. Let $P : U \rightarrow \{1, 2, \cdots, m\}^s$, be a random function. Suppose $P(i) = (c_1, c_2, \cdots, c_s)$, then user i's signet $\alpha_i = a - (\sum_{j=1}^{s} b_{c_j} u_i^j)$, i.e. $\alpha_i = f_i(u_i)$, where

$$\begin{aligned} f_i(x) &= a - (\sum_{j=1}^{s} b_{c_j} x^j) \\ &= a - (\sum_{j=1}^{m} \sum_{r:P(i)_r = r} b_j x^r), \end{aligned}$$

where $P(i)_r$ denotes the rth entry of $P(i)$. α_i together with $P(i)$ are distributed to user i. User i can compute $g^a = g^{\alpha_i} \prod_{j=1}^{s} y_{c_j}^{u_i^j}$. A full session key is obtained by concatenating $g_1^a, g_2^a, \cdots$ as in the original scheme. This modified scheme reduces the amount of computation during the generation of the signets and session keys, but increases the amount of public information required.

The above modification does not prevent the attack outlined in Section 4.1. WLOG, let $\{1, 2, \cdots t\}$ be a coalition of traitors. The coalition computes a convex combination of $f_i(u_i)$, $1 \leq i \leq t$: $v_0 = \sum_i^t c_i f_i(u_i)$, where $\sum_i^t c_i = 1$. And for each j, $1 \leq j \leq m$, the coalition also computes $v_j = \sum_{i=1}^t \sum_{r:P(i)_r=j} c_i u_i^r$. To obtain g^a, one can compute

$$\begin{aligned}
& g^{v_0} \prod_{j=1}^{m} y_j^{v_j} \\
= \; & g^{\sum_i^t c_i f_i(u_i)} g^{\sum_{j=1}^m b_j \sum_{i=1}^t \sum_{r:P(i)_r=j} c_i u_i^r} \\
= \; & g^{\sum_i^t c_i a} g^{-\sum_{i=1}^t c_i \sum_{j=1}^m \sum_{r:P(i)_r=j} b_j u_i^r} g^{\sum_{j=1}^m b_j \sum_{i=1}^t \sum_{r:P(i)_r=j} c_i u_i^r} \\
= \; & g^a g^{-\sum_{i=1}^t \sum_{j=1}^m \sum_{r:P(i)_r=j} b_j c_i u_i^r} g^{\sum_{i=1}^t \sum_{j=1}^m \sum_{r:P(i)_r=j} b_j c_i u_i^r} \\
= \; & g^a
\end{aligned}$$

The authors claimed that there is a high probability that for a random set of k traitors, there exist a b_j such that b_j is only assigned to exactly one traitor say t. So u_t can be found easily giving v_j. Let us consider this probability. There are k traitors, each assigned s elements from the set $B = \{b_1, b_2, \cdots, b_m\}$ with replacement. So there are a total of m^{sk} ways to do the assignment. WLOG, say the first number assigned to traitor t is b_j which is not assigned to any other traitor. There are m ways to choose b_j, and there are $m^{(s-1)}$ ways to assign $s-1$ more numbers to t, as b_j can be assigned to t more than once. For the rest of the $k-1$ users there are $m-1$ elements available, and hence there are $(m-1)^{s(k-1)}$ possible assignments. Thus the probability that one traitor is assigned a b_j which is not assigned to any other traitor is:

$$\begin{aligned}
pr &= \frac{m m^{s-1} (m-1)^{s(k-1)}}{m^s} \\
&= \frac{(m-1)^{s(k-1)}}{m^{s(k-1)}}
\end{aligned}$$

$$\begin{aligned} &= \left(1-\frac{1}{m}\right)^{s(k-1)} \\ &= \left(\left(1-\frac{1}{m}\right)^{m}\right)^{s\frac{k-1}{m}} \\ &\leq \left(e^{-1}\right)^{s\frac{k-1}{m}} \end{aligned}$$

pr is less than $\frac{1}{4}$ if we take $s = 3$ and $m = 2(k-1)$. And it is less than $\frac{3}{4}$ if we take $s = 3$ and $m = 10(k-1)$. Furthermore this is the probability for a *randomly chosen* coalition. The probability that there is a coalition of size k which does not have such property would be extremely high.

Even if there is a b_j such that b_j is assigned to traitor t only. Then $v_j = a_t \sum_{r:P(t)_r=j} u_t^r$. But without the knowledge of a_t, it is impossible to find u_t. And v_j could even be a linear combination of the powers of some other u_i's. Thus we believe the attack still works against this modified version of the signet scheme.

Signet is quite similar to the Kurosawa-Desmedt scheme, except that signet attempted to address the problem of a pirate redistributing session keys. Consequently, the public information required is $O(k)$ times the actual content to be broadcasted. This seems very impractical.

Chapter 7

Conclusion

We have surveyed several traceability schemes in the previous chapters. We attempted to categorize them into three types: symmetric, asymmetric and public, based on how the session keys are encrypted and decrypted.

CFNP and SW schemes are symmetric schemes in the sense that the session keys are encrypted using symmetric encryption schemes: the keys used to encrypt and decrypt each share are the same. The existence of six types of CFNP were proven, while actual constructions for the open one level scheme were also given. The constructions arose from combinatorial objects such as transversal designs and orthogonal arrays.

Stinson-Wei's scheme is similar to the one level open CFNP scheme. The authors provided constructions that are as efficient as the constructions for the CFNP scheme. Stinson-Wei's constructions are based on combinatorial objects such as balanced incomplete block designs.

Given the set of keys in a pirate decoder, a straightforward algorithm is available to identify at least one traitor. Although black box tracing algorithms were also

provided, they were much more complicated, and even worse, they can be detected by a private decoder having extra keys. The importance of the requirement of a black box tracing algorithm might be overstated. If the traitors can access their decoders to find out their personal keys, it seems reasonable to assume that the data supplier would be also able to access the pirate decoder to determine the key without a black box algorithm. It does not make sense if the pirate decoder would provide more tamper resistance than a legitimate decoder, and hence cost more money to make.

The amount of base keys the data supplier has to store is in the order of $O(k^4 \log n)$ (using a one level open scheme). The same amount of decryptions have to be performed to decrypt a session key. Each user has to store $O(k^2 \log n)$ keys and perform the same amount of decryptions to obtain a session key.

The Kurosawa-Desmedt and Boneh-Franklin schemes are asymmetric schemes. The keys used to encrypt and decrypt a session key are different. In fact, the encryption scheme alone is a valid public key encryption scheme. But, due to the requirement of tracing, each user's private key is known to the data supplier. Thus the DS can impersonate any user at his will.

The security of encryption schemes in the both of these schemes rely on the discrete logarithm problem. The tracing algorithm in the Kurosawa-Desmedt scheme was broken. The authors made an unrealistic assumption that all pirate decoder's decryption keys must be of the same form as the legitimate decoders.

The Boneh-Franklin scheme also requires an assumption on the structure of the key in all pirate decoders. The soundness of the tracing algorithm relies on Conjecture 4.2.5. If the conjecture holds, then given a key captured from a pirate decoder, the Boneh-Franklin scheme employs coding theory techniques to trace all

traitors who helped in constructing the key. A black box tracing algorithm was provided to trace the traitors in a single key pirate only if trapdoors can be planted to help the data supplier computing discrete logarithms in the underlying group. No efficient black box tracing algorithm is available for arbitrary pirates.

The biggest advantage of Boneh-Franklin scheme is that it can trace all traitors who contributed in building the key in a pirate decoder. In the CFNP scheme, only one of the traitors is identified. In the Boneh-Franklin scheme, each user's personal key has size $O(k \log q)$ bits, and the same amount of group multiplications are required in decryption. q is usually very large: e.g., in the order of 2^{80}, which is much larger than n. The amount of information each user has to store in the one level open CFNP scheme is slightly more then what a user has to store the Boneh-Franklin scheme. The data supplier has to store $O(nk \log q)$ bits of data in the Boneh-Franklin scheme while in the CFNP scheme $O(k^4 \log n)$ keys are stored. Assuming each key is roughly $\log q$ bits in size, and k is much smaller than n, CFNP scheme has an advantage in the amount of information has to be stored by the data supplier.

There are a couple of attempts to construct public traceability schemes which require the data supplier and users share no secrets at all. One successful attempt is to build upon an asymmetric scheme (such as Boneh-Franklin scheme) and having a trusted third party to handle the key generation and distribution, so that the data supplier has no knowledge on the user's personal keys. One open problem is to construct a public traceability scheme without the requirement of a trusted third party.

Chameleon is a stream cipher designed to support traitor tracing. It is a very attractive scheme since its encryption/decryption is extremely fast. The encryption/ decryption process involves a PRNG and XOR operations only. However

Chameleon uses fingerprinting schemes which require each user's personal key to be extremely large: as large as the size of all base keys in a one level open CFNP scheme.

Digital signet employed the same encryption function as the Kurosawa-Desmedt scheme. Hence it is vulnerable to the same attack which broke the Kurosawa-Desmedt scheme. The digital signet scheme attempted to address the problem of session key redistribution by a pirate. But, in doing so, the amount of information that has to be made public is $O(k)$ times as long as the actual content to be broadcasted. This makes digital signet highly impractical.

Bibliography

[1] N. Alon, J. H. Spencer, P Erdős, **The Probabilistic Method**, Wiley, 1992.

[2] R. Anderson, C. Manifavas *Chameleon - A New Kind of Steam Cipher* "Fast software encryption: 4th International Workshop, FSE '97 Proceedings", Lecture notes in computer science, 1267, (1997), pp. 107-113.

[3] D. Boneh, M. Franklin, *An Efficient Public Key Traitor Tracing Scheme*, "Advances in Cryptology - Crypto '99", Lecture notes in computer science, 1666, pp. 338-353.

[4] D. Boneh and J. Shaw, *Collusion-secure fingerprinting for digital data*, "Advances in Cryptology - Crypto '95", Lecture notes in computer science, 963, pp. 452-465.

[5] B. Chor, A. Fiat, M. Naor, *Tracing Traitors*, full version, preprint.

[6] M. Naor, B. Pinkas, *Threshold Traitor Tracing* "Advances in Cryptology - Crypto '98", Lecture notes in computer science, 1462, pp. 502-517.

[7] C. Dwork, J. Lotspiech, and M. Naor, *Digital Signets: Self-Enforcing Protection of Digital Information*, 28th Symposium on the Theory of Computation 1996, pp. 489-498.

[8] A. Fiat, T. Tassa, *Dynamic Traitor Tracing*, "Advances in Cryptology - Crypto '99", Lecture notes in computer science, 1666, pp. 354-371.

[9] E. Gafni, J. Staddon, Y.L. Yin, *Efficient Methods for Integrating Traceability and Broadcast Encryption*. "Advances in Cryptology - Crypto '99", Lecture notes in computer science, 1666, pp. 372-387.

[10] K. Kurosawa and Y. Desmedt, *Optimum Traitor Tracing and Asymmetric Schemes*, "Advances in Cryptology - Eurocrypt '98", Lecture notes in computer science, 1403, pp. 452-465. '98, pp. 145-157.

[11] T. Okamato, S.Uchiyama, *A new public key cryptosystem as secure as factoring*, "Advances in Cryptology - Eurocrypt '98", Lecture notes in computer science, 1403, pp. 308-318.

[12] P. Paillier, *Public-key cryptosystem based on discrete logarithm residues*, "Advances in Cryptology - Eurocrypt '99", Lecture notes in computer science, 1592, pp.223-238.

[13] B. Pfitzmann, *Trials of Traced Traitors*, "Information hiding : first international workshop" Lecture notes in computer science, 1174 (1996), pp. 49-64.

[14] D. R. Stinson, R. Wei, *Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes*. SIAM J. Discrete Math 11 (1998), pp. 41-53.

[15] D. R. Stinson, R. Wei, *Key Preassigned Traceability Schemes for Broadcast Encryption*. In the Proceedings of SAC '98, Lecture notes in computer science, 1556, (1999), pp. 144-156.