1-1-2006

An efficient single-key pirates tracing scheme using cover-free families

Joseph Tonien
University of Wollongong, dong@uow.edu.au

Rei Safavi-Naini
University of Calgary, rei@uow.edu.au

An efficient single-key pirates tracing scheme using cover-free families

Abstract

A cover-free family is a well-studied combinatorial structure that has many applications in computer science and cryptography. In this paper, we propose a new public key traitor tracing scheme based on cover-free families. The new traitor tracing scheme is similar to the Boneh-Franklin scheme except that in the Boneh-Franklin scheme, decryption keys are derived from Reed-Solomon codes while in our case they are derived from a cover-free family. This results in much simpler and faster tracing algorithms for single-key pirate decoders, compared to the tracing algorithms of Boneh-Franklin scheme that use Berlekamp-Welch algorithm. Our tracing algorithms never accuse innocent users and identify all traitors with overwhelming probability.

Keywords

key, families, single, free, efficient, cover, scheme, tracing, pirates

Disciplines

Engineering | Science and Technology Studies

Publication Details

Tonien, D. & Safavi-Naini, R. (2006). An efficient single-key pirates tracing scheme using cover-free families. 4th International Conference (ACNS 2006) Proceedings in LNCS 3989 (pp. 82-97). Germany: Springer Berlin Heidelberg.

An Efficient Single-Key Pirates Tracing Scheme Using Cover-Free Families

Dongvu Tonien
dong@uow.edu.au

Reihaneh Safavi-Naini
rei@uow.edu.au

School of IT & CS, University of Wollongong, NSW, 2522, Australia

Abstract

A cover-free family is a well-studied combinatorial structure that has many applications in computer science and cryptography. In this paper, we propose a new public key traitor tracing scheme based on cover-free families. The new traitor tracing scheme is similar to the Boneh-Franklin scheme except that in the Boneh-Franklin scheme, decryption keys are derived from Reed-Solomon codes while in our case they are derived from a cover-free family. This results in much simpler and faster tracing algorithms for *single-key pirate* decoders, compared to the tracing algorithms of Boneh-Franklin scheme that use Berlekamp-Welch algorithm. Our tracing algorithms *never* accuse innocent users and identify *all* traitors with overwhelming probability.

Keywords: Public-key traitor tracing, cover-free family.

1 Introduction

In a public key traitor tracing scheme, the encryption key is made public and everyone can use this public key to encrypt messages and broadcast the resulting ciphertexts to all users. Each user is given a unique secret key which can be used to decrypt the broadcasted ciphertexts. Malicious users may combine their decoder keys to construct a pirate decoder that can decrypt the broadcast. A pirate decoder contains a secret key different from all of the colluders' secret keys, or a different decryption algorithm. Pirate decoders can be sold to unauthorised users allowing them to illegally access the content. A tracing algorithm takes a pirate decoder and outputs one of the colluders. Typical applications of such systems are distribution of content in pay-per-view television and web-based content distribution.

Traitor tracing was first introduced by Chor, Fiat and Naor [4]. The first public key traitor tracing scheme was proposed by Boneh and Franklin [2]. In their scheme, two models of pirate decoders are considered. The first model is the *single-key pirate* model and assumes that there are two separate parties called the *key-builder* and the *box-builder*. The key-builder is a group of malicious users who combine their secret keys to create a new pirate decryption key. The pirate key is then handed over to the box-builder who implements the decryption box freely based on this single pirate key. The single-key pirate model is thus a simple but a realistic model of the pirate market. The second pirate model is more sophisticated and allows a pirate decoder with more than one pirate key. In Kiayias and Yung's model [15, 16], a pirate decoder may also have several built-in self protection functionalities, for example, remembering previous tracer queries, erasing internal

keys and shutting down when it "detects" that it is being queried by a tracer. "Crafty pirates" require more advanced tracing algorithms.

A common technique in tracing general pirate decoders is the black box confirmation technique which has been used in many schemes including [2, 26, 35, 22, 6, 19, 20]. Even though, this technique achieves the goal of tracing sophisticated pirate decoders, however, it is obviously not an efficient technique. If c denotes the maximum number of malicious users who have created a pirate decoder, a traitor tracing algorithm using the back box confirmation technique should implement a sub-procedure that takes a subset of c users and determines whether the subset contains the whole set of traitors or not. Thus, for a scheme of n users, up to $\binom{n}{c}$ executions of the sub-procedure may be required. While there has not been any known efficient tracing algorithm for the crafty pirate model, it may be argued that this pirate model is not very realistic as a self protection mechanism in a crafty pirate decoder usually requires the embedding of several keys [36, 37]. It remains as an open problem to design a public key traitor tracing with efficient tracing algorithm against crafty pirates.

In this paper, we only deal with single-key pirates. We propose a new public key traitor tracing scheme with an efficient *combinatorial* traitor tracing algorithm against single-key pirate decoders based on cover-free families. At present, Boneh-Franklin's tracing algorithm [2] is the most efficient algorithm for tracing single-key pirates. This is an *algebraic* algorithm which uses Berlekamp-Welch [1] decoding algorithm for generalized Reed-Solomon codes. Two other traitor tracing schemes [26, 18] also use Berlekamp-Welch algorithm. Our traitor tracing scheme is similar to the Boneh-Franklin scheme except that in the Boneh-Franklin scheme, decryption keys are derived from Reed-Solomon codes, but in our scheme, decryption keys are derived from a cover-free family, resulting in simpler and faster tracing algorithms compared to the tracing algorithms of Boneh-Franklin scheme.

Cover-free families (CFF) are well-studied combinatorial structures with many applications in computer science and cryptography such as information retrieval, data communication, magnetic memories, group testing, key distribution and authentication [14, 32, 31]. It is interesting to discover yet another application of cover-free families for traitor tracing. A c-CFF(m,n) is a pair $(\mathcal{S}, \mathcal{B})$ where $\mathcal{S}$ is a set of m *points* and $\mathcal{B}$ is a collection of n subsets (or *blocks*) of $\mathcal{S}$ with the property that the union of any c blocks cannot cover another block. A cover-free family can be constructed with large n and relatively small m. In our scheme, there are n users that are used to label the n blocks, and m modular linear equations that are used to label the m points. Secret keys of the n users are generated as vector solutions of a certain number of modular equations based on the incidence matrix of the cover-free family. Our tracing algorithms identify traitors by taking intersection of certain subsets derived from the cover-free family and so are simpler and faster than Boneh-Franklin tracing algorithms. The drawback is that our tracing algorithms may not identify all traitors, although we show that they will identify *all* traitors with an *overwhelming probability*. In addition, our algorithms are *error-free*, meaning that an innocent user is never wrongly accused by the algorithms.

Our method of generating secret keys using a number of modular linear equations is inspired by the work of Narayanan et al. [27], although in [27] the set of equations satisfied by a certain secret key is chosen randomly, whereas in our scheme the equations are deterministically determined using the incidence matrix of the cover-free family. In [27], an innocent user may be mistakenly identified as a traitor. In our scheme however, due to the cover-free property, the traitor tracing algorithms will *never* accuse innocent users. We also note that Narayanan et al's scheme is not a public key scheme. Finally, flaws in the key generation algorithm of Narayanan et al's scheme are reported

in [34].

Organization of the paper. Section 2 introduces cover-free families. Section 3 briefly presents our intuition behind the scheme. Section 4 describes our new traitor tracing scheme; and the tracing algorithms are presented separately in Section 5. We conclude our paper in Section 6.

2 Cover-Free Families

Cover-free families were first introduced in 1964 by Kautz and Singleton [14] to investigate superimposed binary codes. Since then, these combinatorial structures have been studied extensively and appeared to have many applications in information theory, combinatorics and cryptography including information retrieval, data communication, magnetic memories, group testing, key distribution and authentication [14, 3, 12, 24, 29, 32, 31].

Definition 1 *A* c-cover-free family *is a pair* $(\mathcal{S}, \mathcal{B})$*, where* $\mathcal{S}$ *is a set of* m *elements and* $\mathcal{B}$ *is a collection of* n *subsets (called* blocks*) of* $\mathcal{B}$ *with the following property: for any* $1 \leq c' \leq c$*, the union of any* c' *blocks cannot contain any other block. We use the notation* c*-CFF*(m, n) *to denote a* c*-cover-free family* $(\mathcal{S}, \mathcal{B})$ *with* $|\mathcal{S}| = m$ *and* $|\mathcal{B}| = n$*.*

For the ease of presentation, through out this paper, we assume $\mathcal{S} = \{1, 2, \ldots, m\}$. The following theorem gives a lower bound for the parameter m in term of parameters c and n. See [11, 13, 30] for different proofs of this theorem.

Theorem 1 *For a* c*-CFF*(m, n)*, it holds that*

$$m \geq \theta \frac{c^2}{\log c} \log n$$

for some constant θ*.*

The constant θ in Theorem 1 is shown to be approximately $1/2$ in [11], approximately $1/4$ in [13] and approximately $1/8$ in [30]. Slightly stronger bounds are given in [33]. A simple construction of cover-free families is based on concatenated codes [7, 8, 9, 10].

For our traitor tracing scheme construction, we want to choose a c-cover-free family with large n and small m since as we will see later, the parameter c becomes the collusion threshold, the parameter n becomes the number of users, and traitor tracing complexity depends on the parameter m.

Suppose we have a c-CFF(m, n) $(\mathcal{S}, \mathcal{B})$ with $\mathcal{S} = \{1, 2, \ldots, m\}$ and $\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$. We construct its *incidence matrix* $\mathcal{M}$ as follows. The matrix has n rows and m columns. Label n rows by n blocks of $\mathcal{B}$ and label m columns by m elements of the set $\mathcal{S}$. The entry $\mathcal{M}[i, j]$ at row labeled by B_i and column j is 1 if $j \in B_i$ and is 0 if $j \notin B_i$. The c-cover-free property is interpreted in the incidence matrix as follows. For any c' blocks $B_{i_1}, B_{i_2}, \ldots, B_{i_{c'}}$, where $1 \leq c' \leq c$, and any other block B_k, since $B_{i_1} \cup B_{i_2} \cup \cdots \cup B_{i_{c'}}$ does not contain B_k, there must exist $j \in B_k$ such that $j \notin B_{i_1}$, $j \notin B_{i_2}$, $\ldots$, and $j \notin B_{i_{c'}}$. It means that if we take arbitrary c' rows $i_1, i_2, \ldots, i_{c'}$ and any other row k, then there exists at least a column j such that $\mathcal{M}[i_1, j] = \mathcal{M}[i_2, j] = \cdots = \mathcal{M}[i_{c'}, j] = 0$ and $\mathcal{M}[k, j] = 1$. The *complementary incidence matrix* $\mathcal{M}'$ is obtained from the incidence matrix $\mathcal{M}$ by replacing the entries 1 by 0 and replacing 0 by 1. The following property of the complementary

incidence matrix $\mathcal{M}'$ plays the crucial role in constructing our new traitor tracing scheme. That is, for any $1 \leq c' \leq c$, if we take arbitrary c' rows and another row of $\mathcal{M}'$, then there exists at least a column whose entries on these c' rows are all 1 and the entry on the other row is 0.

$\mathcal{M}$

	$\ldots$	j	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_1		0	
i_2		0	
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i_{c'}$		0	
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k		1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$

$\mathcal{M}'$

	$\ldots$	j	$\ldots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
i_1		1	
i_2		1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$i_{c'}$		1	
$\vdots$	$\vdots$	$\vdots$	$\vdots$
k		0	
$\vdots$	$\vdots$	$\vdots$	$\vdots$

3 Idea

Suppose we want to construct a public key traitor tracing scheme with n users and c is the collusion threshold. Then we need to use a c-CFF(m, n) $(\mathcal{S}, \mathcal{B})$ with an $n \times m$ *complementary incidence matrix* $\mathcal{M}'$. We will generate m random *modular linear equations*:

$$\begin{array}{rll}
\textit{equation } 1 \quad (E_1): & \mu_{1,1}X_1 + \mu_{1,2}X_2 + \cdots + \mu_{1,t}X_t = 0 & (\text{mod } N_1) \\
\textit{equation } 2 \quad (E_2): & \mu_{2,1}X_1 + \mu_{2,2}X_2 + \cdots + \mu_{2,t}X_t = 0 & (\text{mod } N_2) \\
\vdots & \vdots & \\
\textit{equation } m \quad (E_m): & \mu_{m,1}X_1 + \mu_{m,2}X_2 + \cdots + \mu_{m,t}X_t = 0 & (\text{mod } N_m)
\end{array}$$

where parameters t and $N_1, N_2, \ldots, N_m$ will be described in details later. We now label m columns of $\mathcal{M}'$ by these m equations $E_1, E_2, \ldots, E_m$, and label n rows of $\mathcal{M}'$ by n user keys $\vec{v_1}, \vec{v_2}, \ldots, \vec{v_n}$.

$\mathcal{M}'$

	E_1	E_2	$\ldots$	E_m
$\vec{v_1}$	0	1	$\ldots$	0
$\vec{v_2}$	1	1	$\ldots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vec{v_i}$	1	0	$\ldots$	1
$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\vec{v_n}$	0	1	$\ldots$	1

User i decryption key has the form $\vec{v_i} = (v_{i,1}, v_{i,2}, \ldots, v_{i,t}) \in \mathbf{N}^t$ and is generated in such a way that, for each $1 \leq j \leq m$, if $\mathcal{M}'[i, j] = 1$ then $\vec{v_i}$ satisties the equation E_j, and if $\mathcal{M}'[i, j] = 0$ then $\vec{v_i}$ does not satisty the equation E_j. For example, if the row i of $\mathcal{M}'$ is $(1, 0, \ldots, 1)$ then $\vec{v_i} = (v_{i,1}, v_{i,2}, \ldots, v_{i,t})$ is generated such that $\vec{v_i}$ satisfies equation E_1, *not* satisfy equation E_2, $\ldots$, and satisfies equation E_m.

We will show that in our new traitor tracing scheme, if c' traitors $i_1, i_2, \ldots, i_{c'}$ collude then from their keys $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$ they can only create pirate key $\vec{v}_{pirate}$ that has the form

$$\vec{v}_{pirate} = \alpha_1 \vec{v_{i_1}} + \alpha_2 \vec{v_{i_2}} + \cdots + \alpha_{c'} \vec{v_{i_{c'}}},$$

where $\alpha_1, \alpha_2, \ldots, \alpha_{c'}$ are integer numbers such that $\alpha_1 + \alpha_2 + \cdots + \alpha_{c'} = 1$.

Consider the set $\mathcal{E}$ of equations that are satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$. The linearity implies that the pirate vector $\vec{v}_{pirate}$ also satisfies all equations in the set $\mathcal{E}$. However, from the property of the matrix $\mathcal{M}'$, any innocent user k, there exists at least one equation in the set $\mathcal{E}$ that is *not* satisfied by $\vec{v_k}$.

$\mathcal{M}'$

	...	*set* $\mathcal{E}$	...
$\vdots$		$\vdots\ \vdots\ \vdots\ \vdots$	
$\vec{v_{i_1}}$	...	1 1 1 1	...
$\vec{v_{i_2}}$	...	1 1 1 1	...
$\vdots$		$\vdots\ \vdots\ \vdots\ \vdots$	
$\vec{v_{i_{c'}}}$	...	1 1 1 1	...
$\vdots$		$\vdots\ \vdots\ \vdots\ \vdots$	
$\vec{v_k}$	...	0	...
$\vdots$		$\vdots\ \vdots\ \vdots\ \vdots$	
$\vec{v}_{pirate}$	...	1 1 1 1	...

Therefore, from a pirate key $\vec{v}_{pirate}$, we trace the traitors as follows. First, we identify the set $\mathcal{E}$ of equations that are satisfied by $\vec{v}_{pirate}$. Next, for each equation in $\mathcal{E}$, take the corresponding set of vectors that satisfy this equation. Finally, find the intersection of these sets. The set of indices of the vectors in this intersection identifies the traitors. From the above analysis, we can see that no vectors corresponding to innocent users can remain in the intersection because, a vector corresponding to an innocent user must fails at least one equation in the set $\mathcal{E}$.

4 The Proposed Traitor Tracing Scheme

In this section, we present a new public-key traitor tracing scheme based on the idea outlined in the previous section. We show that our proposed scheme is semantically secure against passive adversary assuming the difficulty of the standard DDH problem. The scheme has two tracing algorithms: open-box tracing and black-box tracing which will be presented in the next section.

4.1 Key generation

Let n be the number of users, c be the collusion threshold, and λ, Δ be security parameters.

1. Select a c-CFF(m, n) $(\mathcal{S}, \mathcal{B})$ with an $n \times m$ *complementary incidence matrix* $\mathcal{M}'$ where $m = \theta \frac{c^2}{\log c} \log n$ and θ is a small constant.
2. Choose a group G of Δ-bit order such that it is infeasible to find a multiple of order of G (we can choose G as the group $\mathbf{Z}_M^*$ where $M = pq$ is a RSA modulo). Choose a group element g of high order. Choose $2c + 1$ random numbers d, d_1, ..., d_{2c} such that $\gcd(d_{2c}, |G|) = 1$. Let $y = g^d$, $g_1 = g^{d_1}$, ..., $g_{2c} = g^{d_{2c}}$.
3. Set the *public encryption key* to be $PK = (y, g_1, \ldots, g_{2c})$.
4. Let $z = \lceil m/(2c - 2) \rceil$. Generate z random λ-bit primes p_1, p_2, ..., p_z. Pick m numbers $N_1, N_2, \ldots, N_m$ from $\{p_1, p_2, \ldots, p_z\}$ such that each prime is picked at most $2c - 2$ times.

5. Generate a random $m \times (2c-1)$ matrix $(\mu_{i,j})$ such that any $2c-2$ rows of the matrix are linear independent. Consider the following m random *modular linear equations*

$$\begin{array}{rlll} \textit{equation } 1 & (E_1): & \mu_{1,1}X_1 + \mu_{1,2}X_2 + \cdots + \mu_{1,2c-1}X_{2c-1} = 0 & (\text{mod } N_1) \\ \textit{equation } 2 & (E_2): & \mu_{2,1}X_1 + \mu_{2,2}X_2 + \cdots + \mu_{2,2c-1}X_{2c-1} = 0 & (\text{mod } N_2) \\ \vdots & & \vdots & \\ \textit{equation } m & (E_m): & \mu_{m,1}X_1 + \mu_{m,2}X_2 + \cdots + \mu_{m,2c-1}X_{2c-1} = 0 & (\text{mod } N_m) \end{array}$$

Label m columns of $\mathcal{M}'$ by m equations and label n rows of $\mathcal{M}'$ by n vectors $\vec{v_1}$, $\vec{v_2}$, ..., $\vec{v_n}$. Each vector is of the form $\vec{v_i} = (v_{i,1}, v_{i,2}, \ldots, v_{i,2c-1})$ and is generated in such a way that, for each $1 \le j \le m$, if $\mathcal{M}'[i,j] = 1$ then $\vec{v_i}$ satisfies E_j, and if $\mathcal{M}'[i,j] = 0$ then $\vec{v_i}$ does *not* satisfy E_j. By Chinese Remainder Theorem, we can choose each vector component $v_{i,k}$ as a natural number less than the product $(p_1 p_2 \ldots p_z)$.

6. For each user i, calculate

$$v_{i,2c} = d_{2c}^{-1}(d - d_1 v_{i,1} - d_2 v_{i,2} - \cdots - d_{2c-1} v_{i,2c-1}) \pmod{|G|}$$

and set the *secret decryption key* of user i to be $dk_i = (\vec{v_i}, v_{i,2c}) = (v_{i,1}, v_{i,2}, \ldots, v_{i,2c-1}, v_{i,2c})$.

Example. Let look at steps 4 and 5 in the following toy example with $m = 5$ and $c = 2$.

Step 4: $z = \lceil 5/2 \rceil = 3$. Generate 3 random primes p_1, p_2, p_3. Pick 5 numbers N_1, N_2, N_3, N_4, N_5 from $\{p_1, p_2, p_3\}$ such that each prime is picked at most 2 times. Let's pick $N_1 = N_2 = p_1$, $N_3 = N_4 = p_2$, $N_5 = p_3$.

Step 5: Generate 5 random *modular linear equations*

$$\begin{array}{rlll} \textit{equation } 1 & (E_1): & \mu_{1,1}X_1 + \mu_{1,2}X_2 + \mu_{1,3}X_3 = 0 & (\text{mod } p_1) \\ \textit{equation } 2 & (E_2): & \mu_{2,1}X_1 + \mu_{2,2}X_2 + \mu_{2,3}X_3 = 0 & (\text{mod } p_1) \\ \textit{equation } 3 & (E_3): & \mu_{3,1}X_1 + \mu_{3,2}X_2 + \mu_{3,3}X_3 = 0 & (\text{mod } p_2) \\ \textit{equation } 4 & (E_4): & \mu_{4,1}X_1 + \mu_{4,2}X_2 + \mu_{4,3}X_3 = 0 & (\text{mod } p_2) \\ \textit{equation } 5 & (E_5): & \mu_{5,1}X_1 + \mu_{5,2}X_2 + \mu_{5,3}X_3 = 0 & (\text{mod } p_3) \end{array}$$

Suppose the first row of $\mathcal{M}'$ is $(1,1,0,1,0)$ then the $\vec{v_1} = (v_{1,1}, v_{1,2}, v_{1,3})$ is generated so that

$$\begin{array}{rlll} \textit{equation } 1 & (E_1): & \mu_{1,1}v_{1,1} + \mu_{1,2}v_{1,2} + \mu_{1,3}v_{1,3} = 0 & (\text{mod } p_1) \\ \textit{equation } 2 & (E_2): & \mu_{2,1}v_{1,1} + \mu_{2,2}v_{1,2} + \mu_{2,3}v_{1,3} = 0 & (\text{mod } p_1) \\ \textit{equation } 3 & (E_3): & \mu_{3,1}v_{1,1} + \mu_{3,2}v_{1,2} + \mu_{3,3}v_{1,3} \neq 0 & (\text{mod } p_2) \\ \textit{equation } 4 & (E_4): & \mu_{4,1}v_{1,1} + \mu_{4,2}v_{1,2} + \mu_{4,3}v_{1,3} = 0 & (\text{mod } p_2) \\ \textit{equation } 5 & (E_5): & \mu_{5,1}v_{1,1} + \mu_{5,2}v_{1,2} + \mu_{5,3}v_{1,3} \neq 0 & (\text{mod } p_3) \end{array}$$

We first solve for $(v_{1,1}, v_{1,2}, v_{1,3})$ in (E_1) and (E_2) in modulo p_1, then solve for $(v_{1,1}, v_{1,2}, v_{1,3})$ in (E_3) and (E_4) in modulo p_2, and solve for $(v_{1,1}, v_{1,2}, v_{1,3})$ in (E_5) in modulo p_3, and finally, using Chinese Remainder Theorem to derive the final solution in modulo $p_1 p_2 p_3$.

Remark.

1. The public encryption key $PK = (y, g_1, \ldots, g_{2c})$ contains $2c+1$ group elements, so PK is approximately $(2c+1)\Delta$-bit long.

2. User decryption key $dk_i = (\vec{v_i}, v_{i,2c})$. Since each component of $\vec{v_i}$ is a natural number less than $p_1 p_2 \ldots p_z$, it is $z\lambda$-bit long. Thus, $\vec{v_i}$ is $(2c-1)z\lambda$-bit long. So dk_i is $\Delta + (2c-1)z\lambda \approx \Delta + \lambda\theta\frac{c^2}{\log c}\log n$-bit long.

4.2 Encryption and Decryption

Encryption. A message $M \in G$ is encrypted as

$$(M\, y^r, g_1^r, g_2^r, \ldots, g_{2c}^r),$$

where r is a random number.

Decryption. User i using the secret decryption key dk_i to decrypt

$$\frac{M\, y^r}{(g_1^r)^{v_{i,1}}\, (g_2^r)^{v_{i,2}}\, \ldots\, (g_{2c}^r)^{v_{i,2c}}} = M.$$

The correctness of the decryption algorithm can easily be verified as follows. In the step 6 of the key generation, we have

$$v_{i,2c} = d_{2c}^{-1}(d - d_1 v_{i,1} - d_2 v_{i,2} - \cdots - d_{2c-1} v_{i,2c-1}) \pmod{|G|},$$

so $d_1 v_{i,1} + d_2 v_{i,2} + \cdots + d_{2c-1} v_{i,2c-1} + d_{2c} v_{i,2c} = d \pmod{|G|}$. Thus

$$g^{d_1 v_{i,1}} g^{d_2 v_{i,2}} \ldots g^{d_{2c} v_{i,2c}} = g^d,$$

and

$$g_1^{v_{i,1}} g_2^{v_{i,2}} \ldots g_{2c}^{v_{i,2c}} = y.$$

Therefore,

$$\frac{M\, y^r}{(g_1^r)^{v_{i,1}}\, (g_2^r)^{v_{i,2}}\, \ldots\, (g_{2c}^r)^{v_{i,2c}}} = \frac{M\, y^r}{y^r} = M.$$

4.3 Security of the Encryption Scheme

We show that our encryption scheme is semantically secure against a passive adversary assuming the difficulty of the decision Diffie–Hellman problem in G.

The decision Diffie–Hellman problem in G is to distinguish between tuples of the form $(\nu, \nu^a, \nu^b, \nu^{ab})$ and the form $(\nu, \nu^a, \nu^b, \nu^c)$ where ν is chosen random from G and a, b, c are random number.

With the assumption that the decision Diffie–Hellman problem in G is hard we show that the probability for an adversary to win in the following game is negligible over one half. In this game, the challenger executes the key generation procedure and gives the public encryption key to the adversary. The adversary then produces two messages M_0 and M_1 and gives them to the challenger. The challenger randomly chooses $\delta \in \{0, 1\}$ and gives the adversary a ciphertext of M_δ. The adversary then answers $\delta' \in \{0, 1\}$ and she wins if $\delta' = \delta$.

Theorem 2 *The encryption scheme is semantically secure against a passive adversary assuming the difficulty of the DDH problem.*

A proof of the above theorem can be found in the appendix. Similar to the Boneh–Franklin [2] scheme, our scheme can be modified to achieve security against chosen ciphertext attacks using Cramer–Shoup [5] approach.

5 Traitor Tracing Algorithms

This section is divided into three parts. In the first part, we will show that if the traitors do not know a non-zero multiple of the order of the group G and the discrete log problem in G is hard then the only pirate key that the traitors can construct is a *convex* pirate key. Convex pirate key is a key of the type

$$dk_{pirate} = \alpha_1 dk_{i_1} + \alpha_2 dk_{i_2} + \cdots + \alpha_{c'} dk_{i_{c'}},$$

where $\alpha_1, \alpha_2, \ldots, \alpha_{c'}$ are integer numbers such that $\alpha_1 + \alpha_2 + \cdots + \alpha_{c'} = 1$. Here $dk_{i_1}, dk_{i_2}, \ldots, dk_{i_{c'}}$ are decryption keys of c' traitors with $1 \leq c' \leq c$.

In the second part, we present open-box traitor tracing algorithm. That is how to trace traitors given a convex pirate key dk_{pirate}. Finally, black-box traitor tracing algorithm is presented in the third part.

5.1 Pirate Keys

In the key generation procedure, the public key is set to $PK = (y, g_1, g_2, \ldots, g_{2c})$ where $y = g^d$, $g_1 = g^{d_1}, g_2 = g^{d_2}, \ldots, g_{2c} = g^{d_{2c}}$. A tuple $(e_1, e_2, \ldots, e_{2c}) \in \mathbf{Z}^{2c}$ is said to be a (discrete log) representation of y with respect to the base g_1, g_2, ..., g_{2c} if $y = g_1^{e_1} g_2^{e_2} \ldots g_{2c}^{e_{2c}}$, or equivalently,

$$e_1 d_1 + e_2 d_2 + \ldots + e_{2c} d_{2c} = d \pmod{|G|}.$$

It is clear that each user decryption key $dk_i = (\vec{v_i}, v_{i,2c}) = (v_{i,1}, \ldots, v_{i,2c-1}, v_{i,2c})$ is a representation of y with respect to $g_1, \ldots, g_{2c}$. Any representation $(e_1, e_2, \ldots, e_{2c})$ can be used for decrypting a ciphertext $(M\, y^r, g_1^r, g_2^r, \ldots, g_{2c}^r)$ as

$$\frac{M\, y^r}{(g_1^r)^{e_1} (g_2^r)^{e_2} \ldots (g_{2c}^r)^{e_{2c}}} = M.$$

A group of malicious users $\{i_1, i_2, \ldots, i_{c'}\}$, where $1 \leq c' \leq c$, can use their keys $dk_{i_1}, dk_{i_2}, \ldots, dk_{i_{c'}}$ to construct a pirate key as follows. They select random integer numbers $\alpha_1, \alpha_2, \ldots, \alpha_{c'}$ such that $\alpha_1 + \alpha_2 + \cdots + \alpha_{c'} = 1$ and calculate

$$dk_{pirate} = \alpha_1\, dk_{i_1} + \alpha_2\, dk_{i_2} + \ldots + \alpha_{c'}\, dk_{i_{c'}}.$$

It is easy to see that dk_{pirate} is a representation of y with respect to $g_1, g_2, \ldots, g_{2c}$ so it can be use as a pirate key for decryption.

In this construction of pirate key, we call $\{i_1, i_2, \ldots, i_{c'}\}$ as *active traitors* if all the linear coefficients α_1, α_2, ..., $\alpha_{c'}$ are *non-zero*. The purpose of traitor tracing is to identify these active traitors.

There may be some *inactive traitors* who support the collusion but they did not contribute their keys into the formation of pirate key. It is impossible to trace these inactive traitors. So we only focus on tracing active traitors. For this purpose, we define the following set

$$\mathsf{Convex}(i_1, i_2, \ldots, i_{c'}) = \{\alpha_1\, dk_{i_1} + \ldots + \alpha_{c'}\, dk_{i_{c'}} : \alpha_1, \ldots, \alpha_{c'} \in \mathbf{Z} \setminus \{0\}, \alpha_1 + \cdots + \alpha_{c'} = 1\}.$$

In the following lemma, we show that if the active traitors $\{i_1, i_2, \ldots, i_{c'}\}$ do not know a non-zero multiple of the order of the group G and the discrete log problem in G is hard then the only pirate keys that they can construct are convex pirate keys in the above set $\mathsf{Convex}(i_1, i_2, \ldots, i_{c'})$. A proof of the lemma is given in the appendix.

Lemma 1 *Let $(y, g_1, g_2, \ldots, g_{2c})$ be a public key. Suppose an adversary is given the public key and c private keys $dk_{i_1}, \ldots, dk_{i_c}$. If the adversary can generate a new representation of y with respect to g_1, g_2, ..., g_{2c} that is not in the set*

$$\bigcup_{U \subset \{i_1, i_2, \ldots, i_c\}} \mathsf{Convex}(U)$$

then either the adversary knows a non-zero multiple of $|G|$ or the adversary can effectively compute discrete logs in G.

5.2 Open-Box Tracing Algorithm

In open-box tracing, we assume that the tracer can open the pirate decoder and obtain the pirate key dk_{pirate}. Let $\vec{v}_{pirate}$ be the vector formed by the first $2c-1$ components of dk_{pirate}. Then

$$\vec{v}_{pirate} = \alpha_1 \vec{v_{i_1}} + \alpha_2 \vec{v_{i_2}} + \ldots + \alpha_{c'} \vec{v_{i_{c'}}}$$

where $\alpha_1, \alpha_2, \ldots, \alpha_{c'}$ are *non-zero* integers whose sum is equal to 1.

Recall that in the key generation algorithm, we generate n vectors $\vec{v_1}, \vec{v_2}, \ldots, \vec{v_n}$ and m equations $E_1, E_2, \ldots, E_m$ so that each of the vectors satisfies a number of equations based on the $n \times m$ matrix $\mathcal{M}'$.

For an equation E, let $\mathsf{Vector}(E)$ denote the set of all vectors that satisfy E.

Let denote by $\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})$ the set of all equations that are satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$, and similarly, let denote by $\mathsf{Equation}(\vec{v}_{pirate})$ the set of all equations that are satisfied by $\vec{v}_{pirate}$.

By linearity, any equation that is satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$ must be satisfied by $\vec{v}_{pirate}$. Thus, $\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})$ must be a subset of $\mathsf{Equation}(\vec{v}_{pirate})$.

The following theorem states that it is likely that these two sets are equal and the probability that $\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})$ is a proper subset of $\mathsf{Equation}(\vec{v}_{pirate})$ is negligible.

Theorem 3 *It must hold that*

1. $\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}) \subset \mathsf{Equation}(\vec{v}_{pirate})$;
2. $Pr_{\alpha_1, \ldots, \alpha_{c'}}[\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}) \neq \mathsf{Equation}(\vec{v}_{pirate})] < \frac{2m}{2^\lambda}$.

Let k be an innocent user (i.e. outside of the set of active traitors i_1, i_2, ..., $i_{c'}$). The special property of the matrix $\mathcal{M}'$ states that there must exist an equation E_j such that E_j is satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$ but E_j is *not* satisfied by $\vec{v_k}$. It means that there exists $E_j \in \mathsf{Equation}(\vec{v}_{pirate})$ such that E_j is *not* satisfied by $\vec{v_k}$.

This leads to the following tracing algorithm: first identify the set of all equations, $\mathsf{Equation}(\vec{v}_{pirate})$, that are satisfied by $\vec{v}_{pirate}$, then find the set V of all vectors among $\vec{v_1}$, $\vec{v_2}$, ..., $\vec{v_n}$ that satisfy all equations in $\mathsf{Equation}(\vec{v}_{pirate})$. The index set of the vector set V is then the set of traitors. This set V can be formulated as

$$V = \bigcap_{E \in \mathsf{Equation}(\vec{v}_{pirate})} \mathsf{Vector}(E).$$

5.2.1 The algorithm

Input: A convex pirate key dk_{pirate}

1. Form $\vec{v}_{pirate}$ from the first $2c-1$ components of dk_{pirate};
2. Go through m equations and identify the set $\mathsf{Equation}(\vec{v}_{pirate})$ of all equations that are satisfied by $\vec{v}_{pirate}$.
3. Each equation $E \in \mathsf{Equation}(\vec{v}_{pirate})$ has the associated set $\mathsf{Vector}(E)$. Find the intersection V of all these vector sets.
4. Output the index set X of V.

The following theorem guarantees the correctness of the open-box tracing algorithm.

Theorem 4 *Let $dk_{pirate} \in \mathsf{Convex}(i_1, i_2, \ldots, i_{c'})$ where $1 \leq c' \leq c$, and X be the output of the open-box tracing algorithm executed on the input dk_{pirate}. Then*

1. *X does not contains any innocent users, i.e. for all $1 \leq k \leq n$ if $k \notin \{i_1, \ldots, i_{c'}\}$ then $k \notin X$;*
2. *X is a subset of active traitors, i.e. $X \subset \{i_1, i_2, \ldots, i_{c'}\}$;*
3. *the probability that X contains* all *active traitors is close to 1, more specifically,*

$$Pr[X = \{i_1, \ldots, i_{c'}\}] > 1 - \frac{2m}{2^{\lambda}}.$$

5.2.2 Example

Let look at the following toy example with $c = 2$, $n = 5$, $m = 6$. We use a 2-CFF(6,5) $(\mathcal{S}, \mathcal{B})$ with $\mathcal{S} = \{1, 2, 3, 4, 5, 6\}$ and $\mathcal{B}$ has 5 blocks $B_1 = \{1\}$, $B_2 = \{2, 4\}$, $B_3 = \{3\}$, $B_4 = \{4, 5\}$ and $B_5 = \{6\}$ (*Note to readers:* generally n is much larger than m, please do not get the wrong impression by this toy example!).

$\mathcal{M}$

	1	2	3	4	5	6
$B_1 = \{1\}$	1	0	0	0	0	0
$B_2 = \{2, 4\}$	0	1	0	1	0	0
$B_3 = \{3\}$	0	0	1	0	0	0
$B_4 = \{4, 5\}$	0	0	0	1	1	0
$B_5 = \{6\}$	0	0	0	0	0	1

$\mathcal{M}'$

	E_1	E_2	E_3	E_4	E_5	E_6
$\vec{v_1}$	0	1	1	1	1	1
$\vec{v_2}$	1	0	1	0	1	1
$\vec{v_3}$	1	1	0	1	1	1
$\vec{v_4}$	1	1	1	0	0	1
$\vec{v_5}$	1	1	1	1	1	0

Based on the matrix $\mathcal{M}'$, we have six equations and five vectors are generated for five users. For example, $\vec{v_1}$ satisfies E_2, E_3, E_4, E_5, E_6 but does *not* satisfy E_1.

The associated Vector sets for these equations are:

$$\mathsf{Vector}(E_1) = \{\vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}\}, \quad \mathsf{Vector}(E_2) = \{\vec{v_1}, \vec{v_3}, \vec{v_4}, \vec{v_5}\},$$
$$\mathsf{Vector}(E_3) = \{\vec{v_1}, \vec{v_2}, \vec{v_4}, \vec{v_5}\}, \quad \mathsf{Vector}(E_4) = \{\vec{v_1}, \vec{v_3}, \vec{v_5}\},$$
$$\mathsf{Vector}(E_5) = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_5}\}, \quad \mathsf{Vector}(E_6) = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}\}.$$

Remark that these Vector sets are independent to the generation of equations and vectors. We can find these sets by either looking at matrix $\mathcal{M}'$ or $\mathcal{M}$. For example, based on matrix $\mathcal{M}'$ then $\mathsf{Vector}(E_1)$ is identified by the entries 1 on the first column, and based on matrix $\mathcal{M}$ then $\mathsf{Vector}(E_1)$ is identified by the entries 0 on the first column. These Vector sets can be easily precomputed based on the c-CFF $(\mathcal{S}, \mathcal{B})$.

Now suppose that user 2 and user 3 are active traitors, they construct dk_{pirate}. We will go through the open-box tracing algorithm step by step:

1. Form $\vec{v}_{pirate}$ from the first three components of dk_{pirate}; $\vec{v}_{pirate}$ must be an active convex combination of $\vec{v_2}$ and $\vec{v_3}$;
2. Go through six equations and identify the set of all equations that are satisfied by $\vec{v}_{pirate}$. Since $\vec{v_2}$ and $\vec{v_3}$ both satisfy E_1, E_5, E_6, $\vec{v}_{pirate}$ satisfies E_1, E_5, E_6. As stated in Theorem 3,

   $$\mathsf{Equation}(\vec{v}_{pirate}) \supset \mathsf{Equation}(\vec{v_2}, \vec{v_3}) = \{E_1, E_5, E_6\}.$$

   and it is likely that $\mathsf{Equation}(\vec{v}_{pirate}) = \{E_1, E_5, E_6\}$.

   We assume $\mathsf{Equation}(\vec{v}_{pirate}) = \{E_1, E_5, E_6\}$;
3. Identify the intersection of Vector sets associated with the equations E_1, E_5, E_6:

   $$\begin{aligned} V &= \mathsf{Vector}(E_1) \cap \mathsf{Vector}(E_5) \cap \mathsf{Vector}(E_6) \\ &= \{\vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}\} \\ &= \{\vec{v_2}, \vec{v_3}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}\} \\ &= \{\vec{v_2}, \vec{v_3}\}; \end{aligned}$$

4. Output the index set of V: $X = \{2, 3\}$ – these are active traitors.

5.2.3 Rationale

Firstly, in the step 2 of the above example, one can wonder what would happen if $\mathsf{Equation}(\vec{v}_{pirate})$ contains more than $\{E_1, E_5, E_6\}$, eventhough Theorem 3 asserts that this scenario only happens with a very small probability. The answer is, if this happens then we only catch a subset of active traitors. Indeed, suppose $\mathsf{Equation}(\vec{v}_{pirate}) = \{E_1, E_3, E_5, E_6\}$ then in step 3,

$$\begin{aligned} V &= \mathsf{Vector}(E_1) \cap \mathsf{Vector}(E_3) \cap \mathsf{Vector}(E_5) \cap \mathsf{Vector}(E_6) \\ &= \{\vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_4}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_5}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}\} \\ &= \{\vec{v_2}\}; \end{aligned}$$

Thus, the algorithm outputs one active traitor $X = \{2\}$, and does not detect the other active traitor. We would like to emphasize here that, in all cases, there will be *no* innocent users are mistakenly output as traitors.

Secondly, one can question the significance of the usage of the cover-free family. The answer is, if we do not use cover-free families then the algorithm will output innocent users as traitors.

Consider the following example where $\mathcal{B}$ has one more blocks $B_6 = \{2, 3\}$. Now $(\mathcal{S}, \mathcal{B})$ is no longer 2-cover-free because $B_6 = \{2, 3\}$ is covered by $B_2 = \{2, 4\}$ and $B_3 = \{3\}$. We have one more user,

user 6, and the new matrices are

$\mathcal{M}$

	1	2	3	4	5	6
$B_1 = \{1\}$	1	0	0	0	0	0
$B_2 = \{2,4\}$	0	1	0	1	0	0
$B_3 = \{3\}$	0	0	1	0	0	0
$B_4 = \{4,5\}$	0	0	0	1	1	0
$B_5 = \{6\}$	0	0	0	0	0	1
$B_6 = \{2,3\}$	0	1	1	0	0	0

$\mathcal{M}'$

	E_1	E_2	E_3	E_4	E_5	E_6
$\vec{v_1}$	0	1	1	1	1	1
$\vec{v_2}$	1	0	1	0	1	1
$\vec{v_3}$	1	1	0	1	1	1
$\vec{v_4}$	1	1	1	0	0	1
$\vec{v_5}$	1	1	1	1	1	0
$\vec{v_6}$	1	0	0	1	1	1

The new associated Vector sets are:

$$\begin{aligned}
&\mathsf{Vector}(E_1) = \{\vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}, \vec{v_6}\}, \qquad \mathsf{Vector}(E_2) = \{\vec{v_1}, \vec{v_3}, \vec{v_4}, \vec{v_5}\},\\
&\mathsf{Vector}(E_3) = \{\vec{v_1}, \vec{v_2}, \vec{v_4}, \vec{v_5}\} \quad , \qquad \mathsf{Vector}(E_4) = \{\vec{v_1}, \vec{v_3}, \vec{v_5}, \vec{v_6}\},\\
&\mathsf{Vector}(E_5) = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_5}, \vec{v_6}\}, \qquad \mathsf{Vector}(E_6) = \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_6}\}.
\end{aligned}$$

If user 2 and user 3 are active traitors and in step 2 of the tracing algorithm we have $\mathsf{Equation}(\vec{v}_{pirate}) = \{E_1, E_5, E_6\}$ then in step 3,

$$\begin{aligned}
V &= \mathsf{Vector}(E_1) \cap \mathsf{Vector}(E_5) \cap \mathsf{Vector}(E_6)\\
&= \{\vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_5}, \vec{v_6}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_5}, \vec{v_6}\} \cap \{\vec{v_1}, \vec{v_2}, \vec{v_3}, \vec{v_4}, \vec{v_6}\}\\
&= \{\vec{v_2}, \vec{v_3}, \vec{v_6}\}.
\end{aligned}$$

The algorithm has mistaken output user 6 as an active traitor.

5.2.4 Comparison with Boneh–Franklin's Scheme

While our encryption scheme is the same as the encryption scheme of Boneh–Franklin [2], our tracing algorithm is much simpler. Tracing algorithm in Boneh–Franklin's scheme involves solving a linear system of dimension n (the total number of users) and decoding BCH error-correcting codes using Berlekamp's [1] algorithm. Whereas, in our tracing algorithm, it only has two simple steps:

Step 1: Finding the set $\mathsf{Equation}(\vec{v}_{pirate})$ of equations that are satisfied by $\vec{v}_{pirate}$. There are totally $m = \theta \frac{c^2}{\log c} \log n$ equations. This step involves m number of testings whether the vector $\vec{v}_{pirate}$ satisfies each equation or not.

Step 2: Finding the intersection V of Vector sets associated with equations in $\mathsf{Equation}(\vec{v}_{pirate})$. This is a very simple step because m Vector sets associated with m equations are precomputed.

Let r be a small positive integer (for example $r = 2$). The intersection step is performed even faster if we precompute and store $\binom{m}{r}$ intersection sets

$$V_{\{i_1,i_2,\ldots,i_r\}} = \mathsf{Vector}(E_{i_1}) \cap \mathsf{Vector}(E_{i_2}) \cap \cdots \cap \mathsf{Vector}(E_{i_r}) \quad \text{where } 1 \le i_1 < i_2 < \cdots < i_r \le m.$$

These intersection sets have small cardinalities compared to n. If $|\mathsf{Equation}(\vec{v}_{pirate})| < r$ then V is an intersection of small number ($< r$) of sets Vector. If $|\mathsf{Equation}(\vec{v}_{pirate})| \ge r$ then V is the intersection of $|\mathsf{Equation}(\vec{v}_{pirate})|/r < m/r$ number of intersection sets $V_{\{i_1,i_2,\ldots,i_r\}}$.

With a much simpler tracing algorithm, our scheme achieves almost the same goals as the Boneh–Franklin scheme:

Error Free Tracing: There are no innocent users mistakenly output by the tracing algorithm as traitors. Output of the tracing algorithm are active traitors.

Full Tracing: While the tracing algorithm in the Boneh–Franklin scheme always outputs *all* active traitors, our tracing algorithm outputs all active traitors with probability almost near 1. Our algorithm outputs a proper subset of active traitors with only a negligible probability.

5.3 Black-Box Tracing Algorithm

A black-box tracing algorithm for single-key pirate can be developed using Boneh–Franklin's [2] approach. In this approach, we need to choose a underlying group G so that the tracer can efficiently solve the discrete log problem in the group such as those used in [28]. If this is the case, then suppose $dk_{pirate} = (v_1, v_2, \ldots, v_{2c})$ is a pirate key, we can find the values v_1, v_2, $\ldots$, v_{2c} as follows. Query the pirate device by invalid ciphertexts of the form $C' = (Y, g^{r_1}, \ldots, g^{r_{2c}})$. The pirate device will respond with the value $Y/g^{r_1 v_1 + \ldots + r_{2c} v_{2c}}$. Hence, we can calculate $g^{r_1 v_1 + \ldots + r_{2c} v_{2c}}$. After $2c$ queries, the tracer can calculate g^{v_1}, $\ldots$, $g^{v_{2c}}$, and with the above assumption, all the components of the pirate key v_1, $\ldots$, v_{2c} can be derived by the tracer. From here, the tracer can identify the set of active traitors as it does in the open-box tracing algorithm.

6 Conclusion

In this paper, we show yet another application of cover-free families in cryptography. We show how to use a cover-free family to construct a public-key traitor tracing scheme. The encryption system of our proposed traitor tracing scheme is similar to that of Boneh–Franklin [2] scheme, thus it is semantically secure against passive adversary assuming the intractability of the standard DDH problem. Our scheme can easily modified as the Boneh–Franklin's scheme to obtain chosen ciphertext security against active adversary. The main advantage of our scheme over the Boneh–Franklin is in traitor tracing algorithms. While tracing algorithm in Boneh–Franklin's scheme involves solving a linear system of dimension n (the total number of users) and decoding BCH error-correcting codes using Berlekamp's [1] algorithm, our tracing algorithm only has two simple steps related to $O(\frac{c^2}{\log c} \log n)$ number of modular linear equations (c is the collusion threshold).

References

[1] E.R. Berlekamp and L. Welch, *Error Correction of Algebraic Block Codes*, U.S. Patent No. 4633470, 1986.

[2] D. Boneh and M. Franklin, An Efficient Public Key Traitor Tracing Scheme, *Proceedings of CRYPTO 1999*, *Lecture Notes in Computer Science* 1666, 338–353.

[3] K.A. Bush, W.T. Federer, H. Pesotan and D. Raghavarao, New Combinatorial Designs and Their Application to Group Testing, *Journal of Statistical Planning and Inference* **10** (1984), 335–343.

[4] B. Chor, A. Fiat and M. Naor, Tracing traitors, *Proceedings of CRYPTO 1994*, *Lecture Notes in Computer Science* 839, 257–270, 1994.

[5] R. Cramer and V. Shoup, A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack, *Proceedings of CRYPTO 1998*, *Lecture Notes in Computer Science* 1462, 13–25.

[6] Y. Dodis and N. Fazio, Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack, *Proceedings of PKC 2003*, *Lecture Notes in Computer Science* 2567, 100–115, 2003.

[7] A.G. D'yachkov, V. Lebedev, P. Vilenkin and S. Yekhanin, Cover-Free Families and Superimposed Codes: Constructions, Bounds, and Applications to Cryptography and Group Testing, *Proceedings of ISIT 2001.*

[8] A.G. D'yachkov, A.J. Macula and V.V. Rykov, New Constructions of Superimposed Codes, *IEEE Transactions on Information Theory* **46** (2000), 284–290.

[9] A.G. D'yachkov, A.J. Macula and V.V. Rykov, New Applications and Results of Superimposed Code Theory Arising from the Potentialities of Molecular Biology. In the book: *Numbers, Information and Complexity*, Kluwer Academic Publishers, 265–282, 2000.

[10] A.G. D'yachkov, A.J. Macula, D.C. Torney, P.A. Vilenkin and S.M. Yekhanin, New Results in the Theory of Superimposed Codes, *Proceedings of ACCT-7*, Bansko, Bulgaria, 2000, 126–136.

[11] A.G. D'yachkov and V.V. Rykov, Bounds on the Length of Disjunctive Codes, *Problemy Peredachi Informatsii* **18** (1982), 7–13. [Russian]

[12] M. Dyer, T. Fenner, A. Frieze and A. Thomason, On Key Storage in Secure Networks, *Journal of Cryptology* **8** (1995), 189–200.

[13] Z. Füredi, On r-Cover-Free Families, *Journal of Combinatorial Theory A* **73** (1996), 172–173.

[14] W.H. Kautz and R.C. Singleton, Nonrandom Binary Superimposed Codes, *IEEE Transactions on Information Theory* **10** (1964), 363–377.

[15] A. Kiayias and M. Yung, Self Protecting Pirates and Black-Box Traitor Tracing, *Proceedings of CRYPTO 2001*, *Lecture Notes in Computer Science* 2139, 63–79, 2001.

[16] A. Kiayias and M. Yung, On Crafty Pirates and Foxy Tracers, *Proceedings of DRM 2001*, *Lecture Notes in Computer Science* 2320, 22–39, 2002.

[17] A. Kiayias and M. Yung, Traitor Tracing with Constant Transmission Rate, *Proceedings of EUROCRYPT 2002*, *Lecture Notes in Computer Science* 2332, 450–465, 2002.

[18] A. Kiayias and M. Yung, Breaking and Repairing Asymmetric Public-Key Traitor Tracing, *Proceedings of DRM 2002*, *Lecture Notes in Computer Science* 2696, 32–50, 2003.

[19] C.H. Kim, Y.H. Hwang and P.J. Lee, An Efficient Public Key Trace and Revoke Scheme Secure against Adaptive Chosen Ciphertext Attack, *Proceedings of ASIACRYPT 2003*, *Lecture Notes in Computer Science* 2894, 359–373, 2003.

[20] C. H. Kim, Y. H. Hwang and P. J. Lee, TTS without Revocation Capability Secure Against CCA2, *Proceedings of ACISP 2004*, *Lecture Notes in Computer Science* 3108, 36–49, 2004.

[21] K. Kurosawa and Y. Desmedt, Optimum Traitor Tracing and Asymmetric Schemes with Arbiter, *Proceedings of EUROCRYPT 1998*, *Lecture Notes in Computer Science* 1403, 145–157, 1998.

[22] K. Kurosawa and T. Yoshida, Linear Code Implies Public-Key Traitor Tracing, *Proceedings of PKC 2002*, *Lecture Notes in Computer Science* 2274, 172–187, 2002.

[23] T. Matsushita and H. Imai, A Public-Key Black-Box Traitor Tracing Scheme with Sublinear Ciphertext Size against Self-Defensive Pirates, *Proceedings of ASIACRYPT 2004*, *Lecture Notes in Computer Science* 3329, 260–275.

[24] C.J. Mitchell and F.C. Piper, Key Storage in Secure Networks, *Discrete Applied Mathematics* **21** (1988), 215–228.

[25] M. Naor and B. Pinkas, Threshold Traitor Tracing, *Proceedings of CRYPTO 1998*, *Lecture Notes in Computer Science* 1462, 502–517, 1998.

[26] M. Naor and B. Pinkas, Efficient Trace and Revoke Schemes, *Proceedings of Financial Cryptography 2000*, *Lecture Notes in Computer Science* 1962, 1–20, 2001.

[27] A. Narayanan, C.P. Rangan and K. Kim, Practical Pay TV Schemes, *Proceedings of ACISP 2003*, *Lecture Notes in Computer Science* 2727, 192–203, 2003.

[28] P. Paillier, Public-Key Cryptosystems Based on Discrete Logarithms Residues, *Proceedings of EUROCRYPT 1999*, *Lecture Notes in Computer Science* 1592, 223–238, 1999.

[29] K.A.S. Quinn, Bounds for Key Distribution Patterns, *Journal of Cryptology,* **12** (1999), 227–239.

[30] M. Ruszinkó, On the Upper Bound of the Size of the r-Cover-Free Families, *Journal of Combinatorial Theory A* **66** (1994), 302–310.

[31] R. Safavi-Naini and H. Wang, Multireceiver Authentication Codes: Models, Bounds, Constructions, and Extensions, *Information and Computation* **151** (1999), 148–172.

[32] D.R. Stinson, Tran van Trung and R. Wei, Secure Frameproof Codes, Key Distribution Patterns, Group Testing Algorithms and Related Structures, *Journal of Statistical Planning and Inference.*

[33] D.R. Stinson, R. Wei and L. Zhu, Some New Bounds for Cover-Free Families, *Journal of Combinatorial Theory A* **90** (2000), 224–234.

[34] D. Tonien, *On a Traitor Tracing Scheme from ACISP 2003*, Cryptology ePrint Archive 2005/371.

[35] W. Tzeng and Z. Tzeng, A Public-Key Traitor Tracing Scheme with Revocation Using Dynamic Shares, *Proceedings of PKC 2001*, *Lecture Notes in Computer Science* 1992, 207–224, 2001.

[36] J. Yan and Y. Wu, *An Attack on Black-box Traitor Tracing Schemes*, Rump session, IEEE Symposium on Security and Privacy, Oakland, USA, May 2001.

[37] J. Yan and Y. Wu, *An Attack on A Traitor Tracing Scheme*, Cryptology ePrint Archive Report 2001/067.

Appendix

Proof of Theorem 2. Assume that there exists an adversary, that given the public encryption key $PK = (y, g_1, \ldots, g_{2c})$, produces two messages $M_0, M_1 \in G$. Given the encryption C of M_δ, where δ is chosen random in $\{0, 1\}$, the adversary can identify δ with non-negligible advantage. We show that we can use such adversary to solve the DDH problem in G.

Indeed, given a tuple $(\nu, \nu^u, \nu^v, \nu^w)$, we will decide if $w = uv$.

Step 1. Choose $2c$ random numbers $k_1, \ldots, k_{2c}$. Let $y = \nu$, $g_1 = (\nu^u)^{k_1}, \ldots, g_{2c} = (\nu^u)^{k_{2c}}$.

Step 2. Give $PK = (y, g_1, \ldots, g_{2c})$ to the adversary.
The adversary returns two messages $M_0, M_1 \in G$.

Step 3. Pick a random $\delta \in \{0, 1\}$ and give the adversary the ciphertext

$$C = (M_\delta \, \nu^v, (\nu^w)^{k_1}, \ldots, (\nu^w)^{k_{2c}}).$$

The adversary returns $\delta' \in \{0, 1\}$.

Step 4. If $\delta' = \delta$ then output $w = uv$. Otherwise, output $w \neq uv$.

If $w = uv$ then the ciphertext C is an encryption of M_δ. If $w \neq uv$ then the ciphertext is an encryption of $M' = M_\delta \, \nu^{v - \frac{w}{u}}$, which can be considered as a random message. Therefore, a non-negligible success probability for the adversary implies a non-negligible success probability in solving DDH in G.

Proof of Lemma 1. Suppose there exists an adversary that, given the public key $PK = (y, g_1, g_2, \ldots, g_{2c})$ and c private keys $dk_{i_1}, \ldots, dk_{i_c}$, can generate a new representation of y with respect to $g_1, g_2, \ldots, g_{2c}$ that is not in the set

$$\bigcup_{U \subset \{i_1, i_2, \ldots, i_c\}} \mathsf{Convex}(U),$$

We prove that we can use such adversary to find a non-zero multiple of $|G|$ or to calculate discrete log in G.

Indeed, given $z = g^x$, we show how to use the adversary either to compute x or derive a multiple of $|G|$. First, choose random numbers $r_1, \ldots, r_c$, u, v and two random square matrices $A = (a_{i,j})$ and $B = (b_{i,j})$ of size c such that $\det(A) = \det(B) = 1$. Then, solve for $s_1, \ldots, s_c$, $r_{c+1}, \ldots, r_{2c}$ in the following equations

$$A \cdot \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_c \end{pmatrix} = \begin{pmatrix} u \\ u \\ \vdots \\ u \end{pmatrix} \qquad \text{and} \qquad A \cdot \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_c \end{pmatrix} + B \cdot \begin{pmatrix} r_{c+1} \\ r_{c+2} \\ \vdots \\ r_{2c} \end{pmatrix} = \begin{pmatrix} v \\ v \\ \vdots \\ v \end{pmatrix}.$$

Let $y = g^v z^u$, $g_1 = g^{r_1} z^{s_1}, \ldots, g_c = g^{r_c} z^{s_c}$, $g_{c+1} = g^{r_{c+1}}, \ldots, g_{2c} = g^{r_{2c}}$. For each $1 \leq i \leq c$, let $\alpha_i = (a_{i,1}, \ldots, a_{i,c}, b_{i,1}, \ldots, b_{i,c})$ be the vector formed by joining the ith row of the matrix A and the ith row of the matrix B, then it is easy to check that α_i is a representation of y with respect to $g_1, \ldots, g_{2c}$. Now, give the public key $(y, g_1, \ldots, g_{2c})$ and c representations $\alpha_1, \ldots, \alpha_c$ to the adversary. The adversary will return another representation $\alpha = (t_1, t_2, \ldots, t_{2c})$. Since α is not a convex combination of $\alpha_1, \ldots, \alpha_c$, we must have $t_1 s_1 + \ldots + t_c s_c \neq u$ with overwhelming probability. Since α is a representation, we have $x(t_1 s_1 + \ldots + t_c s_c) + (t_1 r_1 + \ldots + t_{2c} r_{2c}) = xu + v \pmod{|G|}$. So

$x(t_1s_1+\ldots+t_cs_c-u) = (v-(t_1r_1+\ldots+t_{2c}r_{2c}))\pmod{|G|}$. So either $(t_1s_1+\ldots+t_cs_c-u)$ is a non-zero multiple of $|G|$ or we can compute the discrete log $x = (v-(t_1r_1+\ldots+t_{2c}r_{2c}))(t_1s_1+\ldots+t_cs_c-u)^{-1}$.

Proof of Theorem 3. 1. For any equation $E \in \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})$, E must be satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$. Since $\vec{v}_{pirate}$ is a linear combination of $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$, $\vec{v}_{pirate}$ also satisfies the equation E, thus, $E \in \mathsf{Equation}(\vec{v}_{pirate})$. Therefore, $\mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}) \subset \mathsf{Equation}(\vec{v}_{pirate})$.

2. For each equation E_j, we will show that

$$Pr_{\alpha_1,\ldots,\alpha_{c'}}[E_j \in \mathsf{Equation}(\vec{v}_{pirate}), E_j \notin \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})] < \frac{2}{2^\lambda},$$

since there are m equations, we ontain

$$Pr_{\alpha_1,\ldots,\alpha_{c'}}[\mathsf{Equation}(\vec{v}_{pirate}) \neq \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})] < \frac{2m}{2^\lambda}.$$

Indeed, consider the equation E_j:

$$\mu_{j,1}X_1 + \mu_{j,2}X_2 + \cdots + \mu_{j,2c-1}X_{2c-1} = 0 \pmod{N_j}.$$

Let $\vec{\mu_j} = (\mu_{j,1}, \mu_{j,2}, \ldots, \mu_{j,2c-1})$, then $E_j \notin \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})$ implies that

$$\begin{aligned}
\xi_1 &= \vec{\mu_j} \cdot \vec{v_{i_1}} \neq 0 \pmod{N_j} \\
\xi_2 &= \vec{\mu_j} \cdot \vec{v_{i_2}} \neq 0 \pmod{N_j} \\
&\vdots \\
\xi_{c'} &= \vec{\mu_j} \cdot \vec{v_{i_{c'}}} \neq 0 \pmod{N_j}.
\end{aligned}$$

Let $\xi = \vec{\mu_j} \cdot \vec{v_{pirate}}$, then $\xi = \alpha_1\xi_1 + \alpha_2\xi_2 + \cdots + \alpha_{c'}\xi_{c'}$. Since $E_j \in \mathsf{Equation}(\vec{v}_{pirate})$ implies $\xi = 0 \pmod{N_j}$, we have

$$\begin{aligned}
&Pr_{\alpha_1,\ldots,\alpha_{c'}}[E_j \in \mathsf{Equation}(\vec{v}_{pirate}), E_j \notin \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})] \\
&< Pr_{\alpha_1,\ldots,\alpha_{c'}}[\xi = \alpha_1\xi_1 + \alpha_2\xi_2 + \cdots + \alpha_{c'}\xi_{c'} = 0 \pmod{N_j}] = \frac{1}{N_j} < \frac{1}{2^{\lambda-1}},
\end{aligned}$$

where the last inequality is derived from the fact that N_j is a λ-bit prime.

Proof of Theorem 4. 1. We prove by contradiction. Suppose $k \notin \{i_1, i_2, \ldots, i_{c'}\}$ and the output X of the tracing algorithm contains k. Since X is the intersection of all sets $\mathsf{Vector}(E)$ where $E \in \mathsf{Equation}(\vec{v}_{pirate})$, k must be contained in each of the set $\mathsf{Vector}(E)$. It means that $\vec{v_k}$ satisfies any equation E in $\mathsf{Equation}(\vec{v}_{pirate})$. However, the special property of the matrix $\mathcal{M}'$ states that there must exist an equation E_j such that E_j is satisfied by all of the vectors $\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}$ but E_j is *not* satisfied by $\vec{v_k}$. That is, there must exist $E_j \in \mathsf{Equation}(\vec{v}_{pirate})$ such that $\vec{v_k}$ does not satisfy E_j – a contradition. Thus, the tracing algorithm never outputs an innocent user.

2. Since for any $k \notin \{i_1, i_2, \ldots, i_{c'}\}$, we have $k \notin X$, it must follow that $X \subset \{i_1, i_2, \ldots, i_{c'}\}$.

3. We have

$$\bigcap_{E \in \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})} \mathsf{Vector}(E) = \{\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}}\}.$$

Thus,

$$Pr[X \neq \{i_1, \ldots, i_{c'}\}] \leq Pr_{\alpha_1,\ldots,\alpha_{c'}}[\mathsf{Equation}(\vec{v}_{pirate}) \neq \mathsf{Equation}(\vec{v_{i_1}}, \vec{v_{i_2}}, \ldots, \vec{v_{i_{c'}}})] < \frac{2m}{2^\lambda}.$$

It follows that

$$Pr[X = \{i_1, \ldots, i_{c'}\}] = 1 - Pr[X \neq \{i_1, \ldots, i_{c'}\}] > 1 - \frac{2m}{2^\lambda}.$$