Efficient Fully Collusion-Resilient Traitor Tracing Scheme

Sanjam Garg[1], Amit Sahai[1], and Brent Waters[2]

[1]UCLA
[2]UT Austin

Abstract

At Eurocrypt 2006, Boneh et al. [1] presented a fully collusion-resistant traitor tracing scheme. Their scheme is based on composite order bilinear groups, and its security depends on the hardness of the subgroup decision assumption. In this paper we present a new scheme which is based on prime order bilinear groups, and its security depends on the hardness of the decisional linear assumption. Because of this, for the same level of security, our scheme allows for better efficiency (depending on the parameters). For example, if encryption time was the major parameter of concern, then for the same level of security our scheme could encrypt approximately 18 times faster. Also, unlike the Boneh et al. scheme, our scheme does not require a trusted tracing party.

This paper provides general guidelines for transforming a scheme based on composite order bilinear groups to one based on prime order bilinear groups.

1 Introduction

Consider a setup in which a content distributor, like a cable/radio broadcaster, wants to broadcast content while making sure that only those users who have paid for the service have access to the content. In such a system, each user will need a decoder with a secret key in order to decrypt the content. A naive solution would be to use an encryption system such that the corresponding secret key is known to all legitimate users. The broadcasting authority can then encrypt the content and broadcast the ciphertext. All legitimate users with the secret key will be able to decrypt the content. But if a user's key gets hacked, then the attacker could build pirate decoders which it could then distribute. A malicious user could also use his own key to build pirate decoders. The problem is that in this system there is no way to identify *rogue* users. The *traitor tracing* system is designed to solve this problem. The purpose of a traitor tracing system, introduced by Chor et al. [2], is to help content distributors identify pirates.[1] Given a pirate decoder, a traitor tracing system can identify at least one of the users whose key must have been used to construct the pirate decoder. The distributor can then hold the corresponding user responsible for the loss incurred.

A first-hand solution to the problem just described would be to have N instances of the encryption system (in a system of N users) such that the i^{th} secret key is known to the i^{th} user. The broadcasting authority could encrypt the content under each public key and broadcast all the ciphertexts. Each legitimate user will then be able to decrypt the part of ciphertext corresponding to its private key. Given a pirate decoder, it is also possible for this system to identify at least one of the rogue users. But this system is very inefficient. For this system, the ciphertext size is linear in the number of users.

To overcome this limitation of inefficiency, many results with different levels of security guarantees have been proposed. One interesting security property is the *t-collusion-resistant traitor tracing.* A t-collusion-resistant tracing [3, 4, 5, 6, 7, 8, 9, 10] system will work as long as the pirate uses fewer than t user keys in building the pirate box. A t-collusion-resistant traitor tracing system with constant size ciphertexts [11] has also been constructed, but at the cost of increased private key sizes (quadratic in the collusion bound). These systems fail as soon as more than t user keys are used in constructing pirate decoders. A system that allows for traitor tracing for any value of t is called *fully collusion-resistant.* Achieving sublinear size ciphertexts for a fully collusion-resistant traitor tracing system is hard. Boneh

[1]It should be observed that a traitor tracing system does not help to protect content. A user receiving the content can just make copies and re-distribute them. The purpose of a traitor tracing system is to protect the broadcast channel itself.

et al. [1] presented a fully collusion-resistant traitor tracing system with $O(\sqrt{N})$ size ciphertexts, where N is the number of users.

Another issue of concern in these systems is the need for a tracing authority. Depending on whether the tracing party is trusted, traitor tracing systems can be of two types. First kind requires a trusted tracing party [1, 11] that uses a secret tracing key to identify rogue users. Second type of traitor tracing systems allow for a public tracing algorithm that does not require any secret inputs [10, 12, 13, 14, 15]. Traitor tracing systems with public tracing are preferable in certain scenarios.

As an extension to the traitor tracing, some systems combine traitor tracing with broadcast encryption [16] to obtain trace and revoke [17, 18, 19, 20, 21, 22] functionality. This allows the broadcasting authority to revoke a rogue users key, once it is identified.

1.1 Our Contribution

Composite order bilinear groups [23] are a very useful tool and have been used to construct schemes that were previously not known. Hardness assumptions in these groups are based on factoring. Because of sub-exponential attacks against factoring for appropriate security, much larger groups have to be used. The large size of these groups makes them impractical. For example, just a simple exponentiation in composite order bilinear groups is about 25 times slower than one in prime order groups. Also, one pairing operation in these larger composite order groups is approximately 30 times costlier than a pairing in prime order groups. The focus of this research has been to make schemes based on composite order bilinear groups practical.

Recently Boneh, Sahai and Waters [1] (which we refer to as the BSW scheme) came up with a traitor tracing system that was fully collusion-resistant and achieved $O(\sqrt{N})$ size ciphertexts. Their traitor tracing scheme was based on composite order bilinear groups [23]. Its security relied on the hardness of the subgroup hiding assumption. We present a new traitor tracing system that achieves the same properties, but does not use composite order bilinear groups. Instead our scheme is based on prime order bilinear groups, and its security depends on the hardness of the decisional linear assumption. This allows for shorter group elements and a more efficient scheme in certain scenarios (see Section 7 for details). Also, unlike the BSW scheme, our scheme does not need a trusted tracing party. It should be noted that Boneh and Waters [17] improve on [1] and obtain public tracing in their trace and revoke system, but their scheme requires $O(\sqrt{N})$ size private keys. On the other hand private keys in our system are of a constant size.

Now we roughly describe the key ideas used in converting a scheme based on composite order bilinear groups to one based on prime order bilinear groups. These ideas may be useful to convert other schemes as well. The key intuitive idea is that we can replace elements of subgroups in the composite order groups with corresponding analogs in carefully designed vector spaces. An analogous 2-D vector space does not preserve the hardness properties associated with composite order bilinear groups, but fortunately a 3-D vector space with a 2-D subspace and a 1-D subspace does. The elements of this 2-D subspace of the 3-D space differ in a fundamental way from their composite order counterparts. This difference limits when this transformation can be done, but also opens up the possibility of solving problems for which solutions were not previously known. These ideas are explained more formally in Section 4.

1.2 Roadmap

The remainder of this paper is organized as follows: In Sections 2 and 3, we review some definitions and present technical preliminaries. In Section 4, we highlight the new ideas presented in this paper. In Section 5 we give the new scheme and present its proof in Section 6.1. We discuss some of the consequences of the scheme in Section 7, and conclude in Section 8.

2 Traitor Tracing and PLBE

In Boneh et al. [1], a new primitive called *Private Linear Broadcast Encryption* (PLBE) was introduced and showed that a PLBE is sufficient for implementing a fully collusion-resistant traitor tracing scheme. We improve on the BSW PLBE scheme. Since we mainly focus on the PLBE system in our paper, we only give an informal treatment of the traitor tracing system and its relation to PLBE. However, we give details on PLBE definitions and its security properties.

2.1 Traitor Tracing

A traitor tracing system provides protection for a broadcast encrypter. It consists of four algorithms: *Setup, Encrypt, Decrypt* and *Trace*. The *Setup* algorithm generates the secret keys for all the users in the system and the public parameters for the system. By using these public parameters and the algorithm *Encrypt*, any user can encrypt a message to all the users in the system. A recipient can use his secret key and the *Decrypt* algorithm to decrypt a ciphertext. In case an authority discovers a pirate decoder, it can then use the $Trace$ algorithm to identify at least one of the users whose private key must have been used in the construction of the pirate decoder.

The desired security properties of a traitor tracing system are the following:

- **Semantic Security**: An adversary that does not have access to the secret key of any user should not be able to distinguish between encryptions of two messages of its choice.
- **Traceability Against Arbitrary Collusion**: Consider a case where an adversary has access to an arbitrary number of keys of its choice and generates a pirate decoder. Then the tracing algorithm should be able to use the pirate decoder and detect at least one of the users whose key must have been used to construct the pirate decoder.

2.2 PLBE

A *Private Linear Broadcast Encryption* (PLBE) system consists of four algorithms: $Setup_{PLBE}$, $Encrypt_{PLBE}$, $Decrypt_{PLBE}$, $TrEncrypt_{PLBE}$. These algorithms are very similar to the BSW PLBE system [1] except that our system does not need a tracing key.

- $(PK, K_1, K_2 \dots K_N) \xleftarrow{\$} Setup_{PLBE}(\lambda)$: $Setup_{PLBE}$ algorithm takes as input the security parameter λ and sets up the public parameters PK for the system along with generating the secret keys $(K_1, K_2 \dots K_N)$ for all the users in the system.
- $C \xleftarrow{\$} Encrypt_{PLBE}(PK, M)$: Any user can encrypt a message M using just the public key PK, and any user that possess one of the secret keys can decrypt the ciphertext.
- $M \leftarrow Decrypt(C, K_i, i)$: Any user i having access to the private key K_i can decrypt a ciphertext C and obtain the corresponding message M.
- $C \xleftarrow{\$} TrEncrypt_{PLBE}(PK, i, M)$: The $TrEncrypt_{PLBE}$ algorithm takes in a message M and encrypts it to ciphertext C such that only users $\{i \dots N\}$ with secret keys $(K_i, K_{i+1} \dots K_N)$ can decrypt the message. This algorithm is used only for tracing.

2.3 Desired Security Properties

A PLBE system is considered secure if no adversary has significant advantage in the following games:

- **Indistinguishability:** This property requires that the ciphertexts generated by $Encrypt_{PLBE}(PK, M)$ and $TrEncrypt_{PLBE}(PK, 1, M)$ are indistinguishable. The game between the adversary and the challenger proceeds as follows.
  - **Setup:** The challenger runs the $Setup_{PLBE}$ algorithm and sends the generated public key PK and the secret keys $K_1, K_2 \dots K_N$ to the adversary.
  - **Challenge:** The adversary sends a message M to the challenger. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. If $\beta = 0$, it then sets the ciphertext as $C \xleftarrow{\$} Encrypt_{PLBE}(PK, M)$, and as $C \xleftarrow{\$} TrEncrypt_{PLBE}(PK, 1, M)$ otherwise. It sends C to the adversary.
  - **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

The advantage of the adversary is $Adv_{ID} = |Pr[\beta' = \beta] - \frac{1}{2}|$.

- **Index Hiding:** This property prevents an adversary from distinguishing between $TrEncrypt_{PLBE}(PK, i, M)$ and $TrEncrypt_{PLBE}(PK, i+1, M)$ when the adversary knows all the secret keys except the i^{th} secret key. The game between the adversary and the challenger proceeds as follows. The game takes the index i as input which is given as input to both the challenger and the adversary.
  - **Setup:** The challenger runs the $Setup_{PLBE}$ algorithm and sends the generated public key PK and the secret keys $K_1, K_2 \dots K_{i-1}, K_{i+1} \dots K_N$ to the adversary. The adversary does not know K_i.
  - **Challenge:** The adversary sends a message M to the challenger. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. It sets the ciphertext as $C \xleftarrow{\$} TrEncrypt_{PLBE}(PK, i+\beta, M)$ and sends it to the adversary.
  - **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

  The advantage of the adversary is $Adv_{IH}[i] = |Pr[\beta' = \beta] - \frac{1}{2}|$.

- **Message Hiding:** This property requires that an adversary can not break semantic security when encryption is performed on input $i = N + 1$. The game between the adversary and the challenger proceeds as follows.
  - **Setup:** The challenger runs the $Setup_{PLBE}$ algorithm and sends the generated public key PK and the secret keys $K_1, K_2 \dots K_N$ to the adversary.
  - **Challenge:** The adversary sends messages M_0, M_1 to the challenger. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. It sets the ciphertext as $C \xleftarrow{\$} TrEncrypt_{PLBE}(PK, N+1, M_\beta)$ and sends it to the adversary.
  - **Guess:** The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

  The advantage of the adversary is $Adv_{MH} = |Pr[\beta' = \beta] - \frac{1}{2}|$.

Definition 2.1 *An N-user PLBE system is considered secure if for all polynomial time adversaries Adv_{ID}, $Adv_{IH}[i]$ for all $i \in \{1 \dots N\}$ and Adv_{MH} are negligible in the security parameter λ.*

2.4 Equivalence of Traitor Tracing and PLBE

We have presented an intuition behind the argument. A more formal argument appears in [1]. The tracing algorithm will be given a pirate decoder that is able to decrypt messages encrypted using $TrEncrypt(PK, 1, M)$ with significant probability. The probability of success of this pirate decoder, when encryption is done to user $N + 1$, should be negligible because of the message hiding game. The tracing algorithm of the traitor tracing scheme estimates the probability of success of the adversary when the ciphertext is generated using $TrEncrypt(PK, i, M)$ for every $i \in \{1 \dots N + 1\}$. Since the probability is being reduced from significant to negligible between encryptions to $TrEncrypt(PK, 1, M)$ and $TrEncrypt(PK, N + 1, M)$, the probability must fall significantly for some $i \in \{1 \dots N + 1\}$. We argue that the given pirate decoder could not have done this without the knowledge of the i^{th} key. If it didn't know the i^{th} key, then we could use this pirate decoder as an adversary in the Index Hiding game with parameter i and distinguish between $TrEncrypt(PK, i, M)$ and $TrEncrypt(PK, i + 1, M)$ with significant probability. But this can not be true for a secure PLBE. Hence, we can use a secure PLBE to construct a traitor tracing scheme.

3 Background on Bilinear Maps

Our scheme is based on bilinear groups of prime order. below, we provide a brief background on these groups. Since this paper improves on the BSW scheme [1] that was based on composite order bilinear groups. We also give an informal overview of the composite order bilinear groups.

3.1 Bilinear Groups

Bilinear Groups of Prime Order. Consider two multiplicative cyclic groups $\mathbb{G}, \mathbb{G}_T$ of prime order r. Let g be a generator of $\mathbb{G}$. We define a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ with the following properties:

- e is non-degenerate. In other words, $e(g, g)$ should not evaluate to the identity element of $\mathbb{G}_T$.
- The map is bilinear. More formally, $\forall u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_r$ we should have $e(u^a, v^b) = e(u, v)^{ab}$.

Bilinear groups G, for which group operations can be performed efficiently, and for which a target group $\mathbb{G}_T$ exists such that the corresponding bilinear map $e : \mathbb{G} \times \mathbb{G}$ can be computed efficiently are well known. It can be seen that this bilinear map is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

Bilinear Groups of Composite Order. Bilinear groups of composite order are very similar to the ones of prime order. The only difference is that the order of the groups $\mathbb{G}$ and $\mathbb{G}_T$ is composite. Lets say the order is n, where $n = pq$. p and q are large primes depending on the security parameter. In composite order bilinear groups, the non-degeneracy property requires the existence of a generator g of $\mathbb{G}$ such that $\mathbb{G}_T$ is also of order n. We will use $\mathbb{G}_p$ and $\mathbb{G}_q$ to denote the p and q subgroups of $\mathbb{G}$, respectively.

3.2 Complexity Assumptions

Let $\mathcal{G}$ be an algorithm that takes the security parameter λ as input and generates the tuple $(r, \mathbb{G}, \mathbb{G}_T, e)$.

Decision 3-party Diffie Hellman. This assumption is popular and has been used previously in a number of schemes including the BSW PLBE scheme [1]. A challenger generates a bilinear group $\mathbb{G}$ using $(r, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(\lambda)$. It generates a random generator g for the group $\mathbb{G}$. It chooses $a, b, c \xleftarrow{\$} \mathbb{Z}_r$.

An algorithm $\mathcal{A}$, solving the Decision 3-party Diffie Hellman problem is given $Z = (r, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^b, g^c)$. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. If $\beta = 0$, it then sets $T = g^{abc}$ and $T = R$ otherwise, where $R \xleftarrow{\$} \mathbb{G}$. It then sends T to $\mathcal{A}$. The adversary returns a guess $\beta' \in \{0, 1\}$ of β. The advantage of $\mathcal{A}$ in this game is $Adv_{D3DH} = |Pr[\beta = \beta'] - \frac{1}{2}|$. The Decision 3-party Diffie Hellman assumption states that this advantage is negligible in the security parameter.

Decisional Linear Assumption. This is a simple extension of the Decisional Diffie Hellman (DDH) Assumption introduced [24] for bilinear groups in which the DDH assumption is actually easy. A challenger generates a bilinear group $\mathbb{G}$ using $(r, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}(\lambda)$. It generates a random generator g for the group $\mathbb{G}$. It chooses $a, b, c, x, y \xleftarrow{\$} \mathbb{Z}_r$.

An algorithm $\mathcal{A}$, solving the Decisional Linear Assumption problem is given $Z = (r, \mathbb{G}, \mathbb{G}_T, e, g, g^a, g^b, g^c, g^{ax}, g^{by})$. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. If $\beta = 0$, it then sets $T = g^{c(x+y)}$ and $T = R$ otherwise, where $R \xleftarrow{\$} \mathbb{G}$. It then sends T to $\mathcal{A}$. The adversary returns a guess $\beta' \in \{0, 1\}$ of β. The advantage of $\mathcal{A}$ in this game is $Adv_{DLN} = |Pr[\beta = \beta'] - \frac{1}{2}|$. Decisional Linear Assumption states that this advantage is negligible in the security parameter.

Subgroup Decision Assumption. Since we do not use composite order groups in this paper, we do no delve deeply into this assumption. Instead we give an informal idea about the assumption. This problem was introduced by Boneh et al. [25] and states that for a bilinear group $\mathbb{G}$ of composite order $n = pq$, any algorithm $\mathcal{A}$, given a random element $g \in \mathbb{G}$ and a random element $g_q \in \mathbb{G}_q$, can not distinguish between a random element in $\mathbb{G}$ and a random element in $\mathbb{G}_q$.

4 Key Ideas

Consider a composite order bilinear group $\mathbb{G}_n$ of order n, where $n = pq$ and p, q are primes. Let us denote elements belonging to the p-order subgroup (called $\mathbb{G}_p$) and the q-order subgroup (called $\mathbb{G}_q$) of $\mathbb{G}_n$ by subscripts p and q, respectively. The BSW scheme [1] (and most other composite order bilinear group based schemes) relies on the fact that if $g_p \in \mathbb{G}_p$ and $g_q \in \mathbb{G}_q$, then $e(g_p, g_q) = 1$. The same effect can be obtained in a prime order group by using vector spaces. For a group $\mathbb{G}$ of prime order r, with generator g, consider tuples of elements (g^a, g^b) (analogous to g_q) and (g^{-b}, g^a) (analogous to g_p) belonging to the vector space $V = \mathbb{G}^2$ (analogous to $\mathbb{G}_n$), where a, b are random in $\mathbb{Z}_r$. Define vectors $\vec{v_1} = (a, b)$ and $\vec{v_2} = (-b, a)$. Note that they are orthogonal vectors. The subspace V_p (analogous to $\mathbb{G}_p$) corresponds to the set of elements $(g^{a\tilde{p}}, g^{b\tilde{p}})$ such that $\tilde{p} \in \mathbb{Z}_r$; and similarly subspace, V_q (analogous to $\mathbb{G}_q$) corresponds to the set of elements $(g^{-b\tilde{q}}, g^{a\tilde{q}})$ such that $\tilde{q} \in \mathbb{Z}_r$. It is easy to see that pairing an element of V_p with an element of V_q computed[2] as $e(g^a, g^{-b}) \cdot e(g^b, g^a)$ yields the identity element (analogous to $e(g_p, g_q) = 1$).

Now we need to build on an analog of the subgroup decision assumption (SDH). SDH informally states that given an element of $\mathbb{G}$ and an element of $\mathbb{G}_q$, it is hard to distinguish a random element in $\mathbb{G}_q$ from a random element in $\mathbb{G}$. But this assumption does not hold with V_p and V_q. Given an element $(u, v) \in V_q$, we can construct $(v^{-1}, u) \in V_p$. Using these two elements, it is trivial to distinguish an element in V_q from an element in V.

To fix this problem we consider a 3-dimensional vector space, $V = \mathbb{G}^3$. Consider $\vec{\boldsymbol{v_1}} = (a, 0, c)$, $\vec{\boldsymbol{v_2}} = (0, b, c)$ and $\vec{\boldsymbol{v_3}} = \vec{\boldsymbol{v_1}} \times \vec{\boldsymbol{v_2}}$, where a, b, c are random elements in $\mathbb{Z}_r$. Now let us define the subspace V_q by all elements $(g^{a\tilde{q}}, g^{b\tilde{q}'}, g^{c(\tilde{q}+\tilde{q}')})$ such that $\tilde{q}, \tilde{q}' \in \mathbb{Z}_r$, and let the subspace V_p be defined by elements $(g^{-bc\tilde{p}}, g^{-ac\tilde{p}}, g^{ab\tilde{p}})$ such that $\tilde{p} \in \mathbb{Z}_r$. For this system, also pairing an element of V_q with an element of V_p yields the identity element. This system also has an analog of the subgroup decision assumption. Given (g^a, g^b, g^c), we want it to be hard to distinguish a random element $(g^{a\tilde{q}}, g^{b\tilde{q}'}, g^{c(\tilde{q}+\tilde{q}')}) \in V_q$ from an element $(g^{x_1}, g^{x_2}, g^{x_3}) \in V$, where x_1, x_2, x_3 are random. This follows directly from the decisional linear assumption [24].

The main difference between the subspaces defined using composite order bilinear groups and subspaces defined using prime order bilinear groups is the flexibility in the way elements from the sub-spaces can be manipulated. In the case of composite order bilinear groups, it is easy to randomize elements form the sub-space V_q; but on the other hand, for prime order groups similar randomization is hard. This prevents the transformation from being applicable in general.

A direct compilation of the BSW traitor tracing scheme with the new ideas presented earlier doesn't work because of the reasons mentioned in the previous paragraph. But this can be fixed by allowing the encrypter to define the subspaces at the time of encryption. This was not possible in the BSW traitor tracing scheme [1] because the construction was dependent on the primes p, q. More generally, this trick allows, and in fact, necessitates a *late binding* of the parameters that define the subspaces. Other schemes satisfying this property should also be easy to simplify using our trick. Another crucial difference between our scheme and the BSW scheme is that our scheme does not have subspaces in the target group. Even some of the elements in the base group are not moved to the vector space.

5 Our PLBE Construction

Our construction of PLBE, just like the BSW scheme [1], obtains a fully collusion-resistant system with $O(\sqrt{N})$ size ciphertext. However, in our scheme, sublinear size ciphertexts are obtained in a novel way (as explained in Section 4). This allows our construction to rely just on prime order bilinear groups.

The number of users in the system, N, is assumed to be equal to m^2 for some m. If the number of users is not a perfect square, then we add some *dummy* users to pad N to the next perfect square. These *dummy* users do not take part in the system in any way. We arrange the users in an $m \times m$ matrix. The user $u : 1 \leq u \leq N$ in the system is identified by the (x, y) entry of the matrix, where $1 \leq x, y \leq m$ and $u = (x-1) \cdot m + y$.

The ciphertext generated by $Encrypt_{PLBE}$ or $TrEncrypt_{PLBE}$ consists of a ciphertext component for every row and a component for every column. For each row x the ciphertext consists of (A_x, B_x, $R_{x,1}$, $\widetilde{R}_{x,1}$, $R_{x,2}$, $\widetilde{R}_{x,2}$, $R_{x,3}$, $\widetilde{R}_{x,3}$) and for every column y the ciphertext consists of ($C_{y,1}$, $\widetilde{C}_{y,1}$, $C_{y,2}$, $\widetilde{C}_{y,2}$, $C_{y,3}$, $\widetilde{C}_{y,3}$).

An encryption to position (i, j) means that only users (x, y) with $x > i$ or $x = i$ & $y \geq j$ can decrypt the message. An encryption to position (i, j) is obtained in the following way. (It is further illustrated in Figure 1.)

- **Column Ciphertext Components:** Column ciphertext components for columns $y \geq j$ are *well formed* in both subspaces V_p and V_q, while for columns $y < j$ are *well formed* in V_q but are random in V_p.

- **Row Ciphertext Components:** Row ciphertext components for rows $x < i$ are completely random, and these recipients can not obtain the message information theoretically. For row $x = i$,

[2] $e((g^x, g^y), (g^{x'}, g^{y'}))$ is evaluated as $e(g^x, g^{x'}) \cdot e(g^y, g^{y'})$.

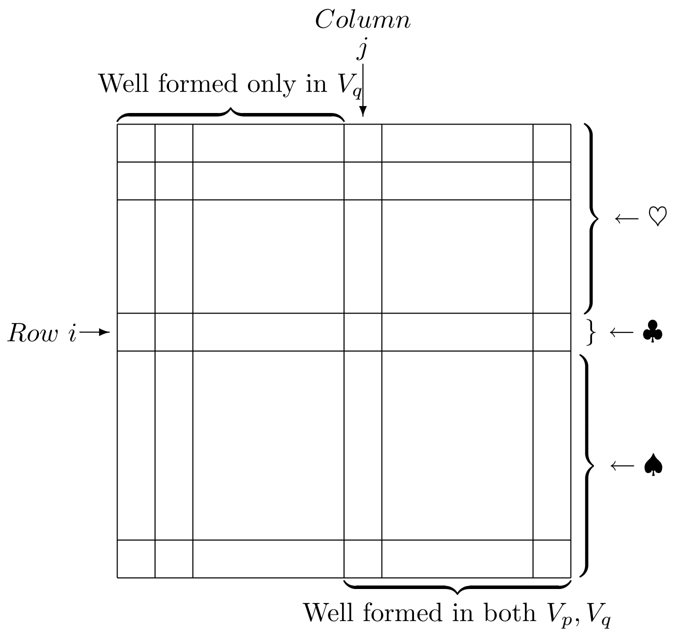

Figure 1: ♡ stands for "Random," ♣ stands for "Well formed in V_p and V_q," and ♠ stands for "Well formed in V_q."

the row ciphertext is *well formed* in both V_p and V_q. And for rows $x > i$ they are *well formed* in V_q and have no component in V_p.

A user in row i will be able to decrypt if the column ciphertext is also well formed in both V_p and V_q. However a user in rows $x > i$, will always be able to decrypt because the row ciphertexts for $x > i$ do not have any component in V_p, and the component of column ciphertexts in V_p will simply cancel out with the row ciphertexts.

Unlike [1] where primes p and q had to be fixed before the public parameters could be generated, our scheme doesn't fix the parameters a, b, c, and they are chosen randomly every time $Encrypt_{PLBE}$ or $TrEncrypt_{PLBE}$ is performed.

The Scheme

The PLBE scheme consists of the algorithms: $Setup_{PLBE}$, $Encrypt_{PLBE}$, $TrEncrypt_{PLBE}$, $Decrypt_{PLBE}$.

- $(PK, K_{(1,1)}, \cdots K_{(1,m)}, K_{(2,1)} \cdots K_{(m,m)}) \leftarrow Setup_{PLBE}\ (1^\lambda, N = m^2)$

  The $Setup_{PLBE}$ algorithm takes as input the security parameter λ and the number of users N in the system. The algorithm generates a bilinear group $\mathbb{G}$ of large prime order r (size depends on the security parameter). It chooses random generator $g \in \mathbb{G}$. It then chooses random $r_1, r_2, r_3, \ldots r_m, c_1, c_2 \ldots c_m, \alpha_1, \alpha_2 \ldots \alpha_m \in \mathbb{Z}_r$.

  The public key PK of the PLBE system (along with the group description) is set to:

$$\begin{aligned}&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots, E_m = g^{r_m},\\ &G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots, G_m = e(g,g)^{\alpha_m},\\ &H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots, H_m = g^{c_m}\end{aligned}$$

  The private key $K_{(x,y)}$ of user (x,y) is:

$$K_{(x,y)} = g^{\alpha_x} \cdot g^{r_x \cdot c_y}$$

  The exponents used in the public key are kept secret by the setup authority.

- $C \leftarrow Encrypt_{PLBE}(PK, M)$

  The $Encrypt_{PLBE}$ algorithm can be used by an user who knows the public key PK. And any recipient having access to a private key can decrypt the generated ciphertext.

  The algorithm chooses random t, η, $w_{1,k}$, $w_{2,k}$, $\ldots w_{m,k}$, s_1, $s_2 \ldots s_m \in \mathbb{Z}_r$ where $k = \{1, 2, 3\}$.

  It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\vec{\boldsymbol{v_1}} = (a, 0, c)$, $\vec{\boldsymbol{v_2}} = (0, b, c)$ and $\vec{\boldsymbol{v_3}} = (-bc, -ac, ab)$. It then sets $\vec{\boldsymbol{v_c}} = (v_{c,1}, v_{c,2}, v_{c,3})$ where $v_{c,1}, v_{c,2}, v_{c,3}$ are chosen randomly in $\mathbb{Z}_r$. For each row $x \in \{1 \cdots m\}$, it picks $\vec{\boldsymbol{v_x}} = \tilde{q}_x \vec{\boldsymbol{v_1}} + \tilde{q}'_x \vec{\boldsymbol{v_2}}$ where $\tilde{q}_x, \tilde{q}'_x$ are random $\in \mathbb{Z}_r$.

  The $Encrypt_{PLBE}$ algorithm sets row ciphertexts as:

  $$\begin{aligned}
  R_{x,k} &= {E_x}^{s_x v_{x,k}} = g^{r_x v_{x,k} s_x} : k = \{1, 2, 3\} \\
  \widetilde{R}_{x,k} &= {E_x}^{s_x v_{x,k} \eta} = g^{r_x v_{x,k} s_x \eta} : k = \{1, 2, 3\} \\
  A_x &= E^{s_x t (\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})} \\
  B_x &= M \cdot {G_x}^{s_x t (\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})} = M \cdot e(g, g)^{\alpha_x s_x t (\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})}
  \end{aligned}$$

  And for every column y the algorithm sets the column ciphertext components as:

  $$\begin{aligned}
  C_{y,k} &= H_y^{t v_{c,k}} g^{w_{y,k} \eta} = g^{c_y v_{c,k} t} g^{w_{y,k} \eta} : k = \{1, 2, 3\} \\
  \widetilde{C}_{y,k} &= g^{w_{y,k}} : k = \{1, 2, 3\}
  \end{aligned}$$

  It should be noted that A_x remains in the base group. Is is not moved into the vector space.

- $M \leftarrow Decrypt_{PLBE}(C, K_{(x,y)}, (x, y))$

  Recipient (x, y) uses the key $K_{(x,y)}$ to decrypt the ciphertext. It uses the parts of the ciphertext corresponding to row x and column y.

  $$M = \frac{B_x}{e(K_{(x,y)}, A_x)} \cdot \frac{\prod_{k=1}^{3} e(R_{x,k}, C_{y,k})}{\prod_{k=1}^{3} e(\widetilde{R}_{x,k}, \widetilde{C}_{y,k})} \tag{1}$$

- $C \leftarrow TrEncrypt_{PLBE}(PK, (i, j), M)$

  This algorithm allows the tracing party to encrypt a message to the recipients who have row value greater than i or those who have row value equal to i and column value greater than or equal to j.

  The algorithm chooses random t, η, $w_{1,k}$, $w_{2,k}$, $\ldots w_{m,k}$, s_1, $s_2 \ldots s_m \in \mathbb{Z}_r$ where $k = \{1, 2, 3\}$.

  It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\vec{\boldsymbol{v_1}} = (a, 0, c)$, $\vec{\boldsymbol{v_2}} = (0, b, c)$ and $\vec{\boldsymbol{v_3}} = (-bc, -ac, ab)$. It then sets $\vec{\boldsymbol{v_c}} = (v_{c,1}, v_{c,2}, v_{c,3})$ and $\vec{\boldsymbol{v'_c}} = \vec{\boldsymbol{v_c}} + v_{c,4} \vec{\boldsymbol{v_3}}$ where $v_{c,1}, v_{c,2}, v_{c,3}, v_{c,4}$ are chosen randomly in $\mathbb{Z}_r$. It can be seen that $\vec{\boldsymbol{v_c}}$ and $\vec{\boldsymbol{v'_c}}$ are fixed for one encryption. This is essential as it is only because of this that we can use $\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}}$ in the row ciphertexts.

  It also chooses random $z_{1,x,k}, z_{2,x}, z_{3,x} \in \mathbb{Z}_r$ where $1 \leq x < i$ and $k \in \{1, 2, 3\}$. For rows $x < i$ it sets the row ciphertext components as:

  $$\begin{aligned}
  R_{x,k} &= g^{z_{1,x,k}} : k = \{1, 2, 3\} \\
  \widetilde{R}_{x,k} &= g^{z_{1,x,k} \eta} : k = \{1, 2, 3\} \\
  A_x &= g^{z_{2,x}} \\
  B_x &= e(g, g)^{z_{3,x}}
  \end{aligned} \tag{2}$$

  It sets $\vec{\boldsymbol{v_i}} = \tilde{q}_i \cdot \vec{\boldsymbol{v_1}} + \tilde{q}'_i \cdot \vec{\boldsymbol{v_2}} + \tilde{p}_i \cdot \vec{\boldsymbol{v_3}}$ where $\tilde{q}_i, \tilde{q}'_i, \tilde{p}_i$ are random in $\mathbb{Z}_r$ and for row $x = i$ it sets row ciphertext components as:

  $$\begin{aligned}
  R_{i,k} &= g^{r_i v_{i,k} s_i} : k = \{1, 2, 3\} \\
  \widetilde{R}_{i,k} &= g^{r_i v_{i,k} s_i \eta} : k = \{1, 2, 3\} \\
  A_i &= g^{s_x t (\vec{\boldsymbol{v_i}} \cdot \vec{\boldsymbol{v_c}})} \\
  B_i &= M \cdot e(g, g)^{\alpha_i s_i t (\vec{\boldsymbol{v_i}} \cdot \vec{\boldsymbol{v_c}})}
  \end{aligned}$$

For each $x \in \{i+1 \cdots m\}$, it picks $\vec{v_x} = \tilde{q}_x \cdot \vec{v_1} + \tilde{q}'_x \cdot \vec{v_2}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$, and for row $x > i$ it sets row ciphertext components as:

$$\begin{aligned}
R_{x,k} &= g^{r_x v_{x,k} s_x} : k = \{1,2,3\} \\
\widetilde{R}_{x,k} &= g^{r_x v_{x,k} s_x \eta} : k = \{1,2,3\} \\
A_x &= g^{s_x t(\vec{v_x} \cdot \vec{v_c})} \\
B_x &= M \cdot e(g,g)^{\alpha_x s_x t(\vec{v_x} \cdot \vec{v_c})}
\end{aligned}$$

And for every column $y < j$

$$\begin{aligned}
C_{y,k} &= g^{c_y v'_{c,k} t} \cdot g^{w_{y,k} \eta} : k = \{1,2,3\} \\
\widetilde{C}_{y,k} &= g^{w_{y,k}} : k = \{1,2,3\}
\end{aligned}$$

And for every column $y \geq j$

$$\begin{aligned}
C_{y,k} &= g^{c_y v_{c,k} t} \cdot g^{w_{y,k} \eta} : k = \{1,2,3\} \\
\widetilde{C}_{y,k} &= g^{w_{y,k}} : k = \{1,2,3\}
\end{aligned}$$

$\vec{v_c}$ is a random element in the V while $\vec{v'_c}$ varies from $\vec{v_c}$ only in the component along V_p subspace. $\vec{v_i}$ corresponds to a random element in V while $\vec{v_x}$ for $x > i$ correspond to elements in V_q. For $x > i$, $\vec{v_x} \cdot \vec{v_c} = \vec{v_x} \cdot \vec{v'_c}$ while for $x = i$, $\vec{v_i} \cdot \vec{v_c} \neq \vec{v_i} \cdot \vec{v'_c}$. This allows a user in row $x > i$ to decrypt the message independent of the column, while for a user in the row $x = i$, decryption is possible only for columns with $y \geq j$.

The correctness of the scheme follows by inspection.

6 Security Proof

6.1 Index Hiding

Theorem 6.1 *If the Decision 3-party Diffie Hellman assumption and the decisional linear assumption hold, then no probabilistic polynomial time adversary can distinguish between an encryption to two adjacent recipients in the index hiding game for any (i, j) where $1 \leq i, j \leq m$ with non-negligible probability.*

Proof. We consider two possible cases. First, when the adversary tries to distinguish between ciphertexts encrypted to (i, j) and $(i, j+1)$ when $1 \leq j < m$. Second, when the adversary tries to distinguish between ciphertexts encrypted to (i, m) and $(i+1, 1)$ when $1 \leq i < m$. The first case follows by Lemma 6.2 and the second case follows by Lemma 6.3. □

6.2 Index Hiding between $(\mathbf{i}, \mathbf{j})$ and $(\mathbf{i}, \mathbf{j+1})$ when $\mathbf{1 \leq j < m}$

Lemma 6.2 *If the Decision 3-party Diffie Hellman assumption holds, then no probabilistic polynomial time adversary can distinguish between an encryption to recipient (i, j) and $(i, j+1)$ in the index hiding game for any (i, j) where $j < m$ with non-negligible probability.*

Proof. This proof is similar to proof of Lemma 5.2 of [1], though some of the public parameter settings are different. The details of the proof can be found in Appendix A. □

6.3 Index Hiding between $(\mathbf{i}, \mathbf{m})$ $(\mathbf{i+1}, \mathbf{1})$

Lemma 6.3 *If the Decision 3-party Diffie Hellman assumption and the decisional linear assumption hold, then no probabilistic polynomial time adversary can distinguish between an encryption to recipient (i, m) and $(i+1, 1)$ in the index hiding game for any $1 \leq i < m$ with non-negligible probability.*

Proof. The proof of this lemma follows from a series of claims that establish the indistinguishability of the following games.

- H_1 Encrypt to column[3] m, row i is the target row,[4] row $i+1$ is the greater-than row.[5]
- H_2 Encrypt to column $m+1$, row i is the target row, row $i+1$ is the greater-than row.
- H_3 Encrypt to column $m+1$, row i is the less-than row, row $i+1$ is the greater-than row (no target row).
- H_4 Encrypt to column 1, row i is the less-than row, row $i+1$ is the greater-than row (no target row).
- H_5 Encrypt to column 1, row i is the less-than row, row $i+1$ is the target row.

It can be observed that game H_1 corresponds to the encryption being done to (i, m) and game H_5 corresponds to encryption to $(i+1, 1)$. The indistinguishability of the games H_1 and H_5, which follows from claims 6.4, 6.5, 6.6, and 6.7, implies the lemma. □

Claim 6.4 *If the Decision 3-party Diffie Hellman assumption holds, then no probabilistic polynomial time adversary can distinguish between games H_1 and H_2 with non-negligible probability.*

Proof. This claim can be proved by applying the result of Lemma 6.2. □

Claim 6.5 *If the Decision 3-party Diffie Hellman assumption holds, then no probabilistic polynomial time adversary can distinguish between games H_2 and H_3 with non-negligible probability.*

Proof. The basic intuition behind the proof is to embed the problem in the $\vec{v_c^p}$ part of $\vec{v_c}$. Since all columns have a random component in V_p, we don't need to actually generate this part. The complete proof can be seen in Appendix B. □

Claim 6.6 *If the Decision 3-party Diffie Hellman assumption holds, then no probabilistic polynomial time adversary can distinguish between games H_3 and H_4 with non-negligible probability.*

Proof. This proof is very similar to the proof of Lemma 6.2. H_3 to H_4 can be expressed as a series of games $H_{3,m+1}$, $H_{3,m}$ $\cdots$ $H_{3,1}$. In the game $H_{3,j}$, all column ciphertexts $(C_y, \widetilde{C}_y)$ are well formed for all y such that $j \leq y \leq m$. It can be seen that $H_{3,1}$ is the same as H_4, and $H_{3,m}$ is the same as H_3. We prove the indistinguishability of games $H_{3,j}$ and $H_{3,j+1}$ for all j where $1 \leq j \leq m$. The proof for this is similar to that of Lemma 6.2. It is, in fact, easier because there is no target row. We show the details of this proof in Appendix C. □

Claim 6.7 *If the decisional linear assumption holds, then no probabilistic polynomial time adversary can distinguish between games H_4 and H_5 with non-negligible probability.*

Proof. We show the details of this proof in Appendix D. □

6.4 Message Hiding

Theorem 6.8 *No adversary can distinguish between two ciphertexts when the encryption is done to the $(m+1, 1)$.*

Proof. This means that all rows will be completely random and independent of the message. Hence, information theoretically the adversary has no way of identifying which message has been encrypted. □

---

[3] Columns greater than or equal to m are well formed, both in V_p and V_q.
[4] The row for which the row component of the ciphertext has well formed components, both in V_p and V_q.
[5] The first row with the row component of ciphertexts only in V_q.

	BSW Scheme [1]	Our Scheme
Encryption	4 exponentiations and 1 double exponentiation[6] in $\mathbb{G}$	$4, 6$ exponentiations[7] in $\mathbb{G}_1$, $\mathbb{G}_2$ respectively and 3 double exponentiation in $\mathbb{G}_1$
Decryption	3 pairings	7 pairings
Ciphertext Size	$5\sqrt{N}$ elements in $\mathbb{G}$	$7\sqrt{N}, 6\sqrt{N}$ elements in $\mathbb{G}_1$, $\mathbb{G}_2$ respectively

Table 1: Table comparing the key parameters of interest

	Symmetric Prime Order	Asymmetric Prime Order	Composite Order
Order (r) of $\mathbb{G}$	160 bits	less than 160 bits	1024 bits
Base Field (b) in $\mathbb{G}$	512 bits	170 bits in $\mathbb{G}_1$ and 510 bits in $\mathbb{G}_2$	a few bits longer than order of group
Exponentiation Time	$O(r \cdot b^2)$	$O(r \cdot b^2)$	$O(r \cdot b^2)$
Pairing Time[8]	25 ms	less than 64 ms[9]	757 ms

Table 2: Costs of different operations

7 Discussion

The BSW PLBE scheme [1] uses composite order bilinear groups. As pointed out in [26], currently, the only known way to generate composite order groups is by symmetric bilinear groups. Our new scheme of using prime order groups allows for more flexibility. We could use different underlying bilinear groups to achieve desired parameters. For example, if the system is bandwidth constrained and we desire shorter ciphertext size, then it might be a good idea to use Weil pairing based asymmetric bilinear groups [27].

As pointed out in [1], a real implementation of broadcast encryption will use a symmetric key cipher under some key K. But this key K still needs to be distributed and that is where our system will be used. By converting our encryption system to a Key Encapsulation Mechanism we can save on computation and ciphertext size. Under this optimization, we do not need to evaluate B_x or include it in the ciphertext. A user (x, y) can extract the key $K_x = e(K_{(x,y)}, A_x)\dfrac{\prod_{i=1}^{3} e(\widetilde{R}_x, \widetilde{C}_y)}{\prod_{i=1}^{3} e(R_x, C_y)}$. The ciphertext would now have to contain an encryption of K under each of the K_x. The user can then derive K from an encryption of it under K_x. The same trick is applicable in our system and we present the evaluation taking this optimization into account.

Encryption time. The most important limiting factor of the BSW scheme was the encryption time. This is because the scheme had to perform $O(\sqrt{N})$ exponentiations. One exponentiation in the (based on Table 2) symmetric prime order bilinear group is roughly $\frac{1024^3}{160 \cdot 512^2} \approx 25$ times faster than an exponentiation in the composite order bilinear group. This implies that the overall encryption is roughly 9.4 times faster[10]. Using asymmetric groups encryption is roughly $\frac{1024^3 \times 5}{160 \cdot 170^2 \times 10 + 160 \cdot 510^2 \times 6} \approx 18$ times faster.

Decryption time. The $Decrypt_{PLBE}$ algorithm as in [1] required three bilinear map operations. Even though decryption in our system requires seven pairing operations (shown in Table 1), our system will be more efficient. This is because a pairing in prime order bilinear groups is more efficient than a pairing in composite order bilinear groups (exact statistics can be found in Table 2).

[6]It involves evaluating $u^a v^b$ and is generally more efficient than two exponentiations.

[7]Setting column ciphertext components and all A_x to be in $\mathbb{G}_1$ and row ciphertext parts $R_{x,k}, \widetilde{R}_{x,k} : k = \{1, 2, 3\}$ to be in $\mathbb{G}_2$.

[8]These time estimates are for the PBC Library as presented online on its website. These are times corresponding to pairings with no preprocessing.

[9]The time 64 ms is for groups $\mathbb{G}_1$ of size 210 bits. But 170 bits are sufficient for reasonable security and the pairing time for these groups will be less than 64 ms.

[10]To simplify the analysis, we assume that double exponentiation costs exactly twice as much as a normal exponentiation.

Ciphertext Size. The ciphertext size in the BSW scheme was $5 \cdot 1024\sqrt{N} = 5120\sqrt{N}$ size. It is slightly larger in our system ($13 \cdot 512\sqrt{N}$). But our system allows use of asymmetric bilinear groups for which we could achieve better performance of ciphertext size with some compromise on decryption time performance. In that case, the elements from $\mathbb{G}_1$ will be 170 bits and elements from $\mathbb{G}_2$ will be 510 bits. This means a total of $4250\sqrt{N}$ bits. The encryption time in this case will be the best, but the decryption will be better for symmetric prime order bilinear groups.

Note that our new scheme is better than the BSW scheme, both in terms of encryption and decryption time under both symmetric and asymmetric prime order groups. It also produces slightly smaller ciphertext if asymmetric bilinear group is used.

Both the schemes rely on the hardness of the Decision 3-party Diffie Hellman assumption. But, our system's security does not depend on the subgroup hiding assumption (a stronger assumption than factoring). Instead it depends on the decisional linear assumption.

8 Conclusions and Ongoing Work

Boneh et al. [1] introduced a new primitive called the Private Linear Broadcast Encryption (PLBE) and used it to build a traitor tracing system. Their system relied on composite order bilinear groups, and its security depended on the hardness of the subgroup decision assumption. We present a new scheme which is based on prime order bilinear groups, and its security depends on the hardness of the decisional linear assumption. Because of this, our scheme also allows for better efficiency (depending on the parameters). Our system is secure under arbitrary collusion and does not need a secret tracing key.

Our paper provides general guidelines for transforming a scheme based on composite order bilinear groups to one based on prime order bilinear groups. These guidelines can be formalized and thus could be useful in other settings. On going work includes optimizing this transformation in the context of asymmetric prime order bilinear groups. Our traitor tracing scheme extends to the Trace and Revoke system of Boneh and Waters [17]. We are also working on a concrete implementation of the above system using the PBC Library.

References

[1] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT 2006, volume 4004 of LNCS*, pages 573–592. Springer-Verlag, 2006.

[2] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO '94: Proceedings of the 14th Annual International Cryptology Conference on Advances in Cryptology*, pages 257–270, London, UK, 1994. Springer-Verlag.

[3] Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 338–353, London, UK, 1999. Springer-Verlag.

[4] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, pages 145–157, 1998.

[5] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, pages 1–20, 2000.

[6] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT*, pages 450–465, 2002.

[7] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography*, pages 100–115, 2003.

[8] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. In *IEICE Trans. Fundamentals*, pages E85–A(2):481484, 2002.

[9] V. D. Tô, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *DRM '03: Proceedings of the 3rd ACM workshop on Digital rights management*, pages 67–76, New York, NY, USA, 2003. ACM.

[10] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[11] Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In *ACM Conference on Computer and Communications Security*, pages 501–510, 2008.

[12] Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In *CT-RSA*, pages 392–407, 2001.

[13] Birgit Pfitzmann. Trials of traced traitors. In *Information Hiding*, pages 49–64, 1996.

[14] Birgit Pfitzmann and Michael Waidner. Asymmetric fingerprinting for larger collusions. In *ACM Conference on Computer and Communications Security*, pages 151–160, 1997.

[15] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In *Digital Rights Management Workshop*, pages 32–50, 2002.

[16] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.

[17] Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220, New York, NY, USA, 2006. ACM.

[18] Dalit Naor, Moni Naor, and Jeffrey B. Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 41–62, London, UK, 2001. Springer-Verlag.

[19] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *FC '00: Proceedings of the 4th International Conference on Financial Cryptography*, pages 1–20, London, UK, 2001. Springer-Verlag.

[20] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 47–60, London, UK, 2002. Springer-Verlag.

[21] Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto 04, volume 2204 of LNCS*, pages 511–527. Springer-Verlag, 2004.

[22] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography*, pages 100–115, London, UK, 2003. Springer-Verlag.

[23] Dan Boneh. Bilinear groups of composite order. In *Pairing '07: Proceedings of the 1st International Conference on Pairing-Based Cryptography*, pages 1–1, Berlin, Heidelberg, 2007. Springer-Verlag.

[24] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Proceedings of CRYPTO .04, LNCS series*, pages 41–55. Springer-Verlag, 2004.

[25] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Second Theory of Cryptography Conference, TCC*, volume 3378 of *LNCS*, pages 325–341, 2005.

[26] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, 2008.

[27] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, London, UK, 2001. Springer-Verlag.

A Proof of Lemma 6.2

Consider an adversary $\mathcal{A}$ that succeeds in the index hiding game with a probability greater than ε. The adversary is considered successful if it can distinguish between encryptions made to positions (i,j) and $(i,j+1)$. We build a reduction $\mathcal{R}$ that uses $\mathcal{A}$ to solve the Decision 3-party Diffie Hellman problem. The reduction receives the Decision 3-party Diffie Hellman challenge as:

$$\mathbb{G}, g, A = g^a, B = g^b, C = g^c, T$$

and it is expected to guess if T is g^{abc} or if it is random.

Next, in the Setup phase the reduction based on the input (i,j) (the row and column the adversary will attack) sets up the public and the private parameters. The reduction chooses random $r_1, r_2, \dots r_m, c_1, c_2 \dots c_m, \alpha_1, \alpha_2 \dots \alpha_m \in \mathbb{Z}_r$. It sets up the public parameters as:

$$\begin{aligned}
&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \dots E_i = B^{r_i} \dots, E_m = g^{r_m},\\
&G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \dots, G_m = e(g,g)^{\alpha_m},\\
&H_1 = g^{c_1}, H_2 = g^{c_2}, \dots H_j = C^{c_j} \dots, H_m = g^{c_m}
\end{aligned} \tag{3}$$

And the private key $K_{(x,y)}$ of user (x,y) is:

$$\begin{aligned}
K_{(x,y)} &= g^{\alpha_x} \cdot B^{r_x \cdot c_y} : x = i, y \neq j\\
K_{(x,y)} &= g^{\alpha_x} \cdot C^{r_x \cdot c_y} : x \neq i, y = j
\end{aligned}$$

Note that the distribution of the public and private parameters matches the distribution of parameters in the real scheme.

In the challenge phase the adversary sends the message $M \in \mathbb{G}_T$ to the reduction. The reduction then chooses random $t, w_{1,k}, w_{2,k}, \dots w_{m,k}, s_1, s_2 \dots s_m \in \mathbb{Z}_r$ where $k = \{1,2,3\}$. It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\vec{v_1} = (a, 0, c)$, $\vec{v_2} = (0, b, c)$ and $\vec{v_3} = (-bc, -ac, ab)$.

Set $g^\eta = B$.

It then sets $\vec{v_c} = (v_{c,1}, v_{c,2}, v_{c,3})$ where $v_{c,1}, v_{c,2}, v_{c,3}$ are chosen randomly in $\mathbb{Z}_r$. Let $\vec{v^q}$ denote the projection of $\vec{v}$ along the plane formed by $\vec{v_1}$ and $\vec{v_2}$. And let $\vec{v^p}$ be the component along $\vec{v_3}$.

It also chooses random $z_{1,x,k}, z_{2,x}, z_{3,x} \in \mathbb{Z}_r$ where $1 \leq x < i$ and $k \in \{1,2,3\}$ and sets up the ciphertext as follows.

$$\begin{aligned}
x < i : \quad R_{x,k} &= g^{z_{1,x,k}} : k = \{1,2,3\}\\
\widetilde{R}_{x,k} &= g^{z_{1,x,k}\eta} : k = \{1,2,3\}\\
A_x &= g^{z_{2,x}}\\
B_x &= e(g,g)^{z_{3,x}}
\end{aligned} \tag{4}$$

It sets $\vec{v_i} = \tilde{q}_i \cdot \vec{v_3} + \tilde{q}'_i \cdot \vec{v_3} + \tilde{p}_i \cdot \vec{v_3}$ where $\tilde{q}_i, \tilde{q}'_i, \tilde{p}_i$ are random in $\mathbb{Z}_r$.

$$\begin{aligned}
x = i : \quad R_{i,k} &= g^{r_i v_{i,k} s_i} : k = \{1,2,3\}\\
\widetilde{R}_{i,k} &= B^{r_i v_{i,k} s_i} : k = \{1,2,3\}\\
A_i &= A^{s_i t(\vec{v_i^p} \cdot \vec{v_c^p})} \cdot g^{s_i t(\vec{v_i^q} \cdot \vec{v_c^q})}\\
B_i &= M \cdot e(g, A_i)^{\alpha_i}
\end{aligned}$$

For each $x \in \{i+1 \cdots m\}$, it picks $\vec{v_x} = \vec{v_x^q} = \tilde{q}_x \cdot \vec{v_1} + \tilde{q}'_x \cdot \vec{v_2}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$.

$$\begin{aligned}
x > i : \quad R_{x,k} &= g^{r_x v^q_{x,k} s_x} : k = \{1,2,3\}\\
\widetilde{R}_{x,k} &= B^{r_x v^q_{x,k} s_x} : k = \{1,2,3\}\\
A_x &= B^{s_x t(\vec{v_x^q} \cdot \vec{v_c})}\\
B_x &= M \cdot e(g, B)^{\alpha_x s_x t(\vec{v_x^q} \cdot \vec{v_c})}
\end{aligned}$$

Choose a random $z \in \mathbb{Z}_r$.

$$
\begin{aligned}
y < j: \quad & C_{y,k} = g^{z v^p_{c,k}} \cdot B^{w_{y,k}} : k = \{1,2,3\} \\
& \widetilde{C}_{y,k} = g^{-c_y t v^q_{c,k}} \cdot g^{w_{y,k}} : k = \{1,2,3\} \\
y = j: \quad & C_{y,k} = T^{c_y t v^p_{c,k}} \cdot B^{w_{y,k}} : k = \{1,2,3\} \\
& \widetilde{C}_{y,k} = C^{-c_y v^q_{c,k} t} \cdot g^{w_{y,k}} : k = \{1,2,3\} \\
y > j: \quad & C_{y,k} = B^{w_{y,k}} : k = \{1,2,3\} \\
& \widetilde{C}_{y,k} = A^{-c_y v^p_{c,k} t} \cdot g^{-c_y t v^q_{c,k}} \cdot g^{w_{y,k}} : k = \{1,2,3\}
\end{aligned}
$$

If T corresponds to g^{abc}, then the ciphertext corresponding to (i, j) is well formed; and if T is randomly chosen, then the encryption corresponds to $(i, j+1)$. The reduction will receive the guess γ from $\mathcal{A}$ and it passes on the same value to the Decision 3-party Diffie Hellman challenger. The advantage of the reduction is exactly equal to the advantage of the adversary $\mathcal{A}$.

B Proof of Claim 6.5

Consider an adversary $\mathcal{A}$ that can distinguish between H_2 and H_3 with a probability greater than ε. We build a reduction $\mathcal{R}$ that uses $\mathcal{A}$ to solve the Decision 3-party Diffie Hellman problem. The reduction receives the Decision 3-party Diffie Hellman challenge as:

$$\mathbb{G}, g, A = g^a, B = g^b, C = g^c, T$$

and it is expected to guess if T is g^{abc} or if it is random.

Next, in the Setup phase the reduction based on the input i (the row the adversary wants to attack) sets up the public and the private parameters. The reduction chooses random $r_1, r_2, \ldots r_{i-1}, r_{i+1} \ldots r_m$, $c_1, c_2 \ldots c_m$, $\alpha_1, \alpha_2 \ldots \alpha_{i-1}, \alpha_{i+1} \ldots \alpha_m \in \mathbb{Z}_r$. It sets $g^{\alpha_x} = g^{a \cdot b}$ and $g^{r_x} = B$. It doesn't know g^{ab} but can generate $G_i = e(A, B)$ and $K_{x,y} = g^{ab} g^{((c_y - a)b)} = B^{c_y}$. It sets up the public parameters as:

$$
\begin{aligned}
& g, E_1 = g^{r_1} \ldots E_i = B \ldots E_m = g^{r_m}, \\
& G_1 = e(g,g)^{\alpha_1}, \ldots G_i = e(A,B), \ldots, G_m = e(g,g)^{\alpha_m}, \\
& H_1 = g^{c_1} \cdot A^{-1}, H_2 = g^{c_2} \cdot A^{-1} \ldots H_m = g^{c_m} \cdot A^{-1}
\end{aligned} \tag{5}
$$

And the private key $K_{(x,y)}$ of user (x,y) is:

$$
\begin{aligned}
K_{(x,y)} &= g^{\alpha_x} \cdot (g^{c_y} \cdot A^{-1})^{r_x} : x \neq i \\
K_{(x,y)} &= B^{c_y} : x = i
\end{aligned}
$$

Note that the distribution of the public and private parameters matches the distribution of parameters in the real scheme.

In the challenge phase the adversary sends the message $M \in \mathbb{G}_T$ to the reduction. The reduction then chooses random t, η, $w_{1,k}$, $w_{2,k}$, $\ldots w_{m,k}$, s_1, $s_2 \ldots s_m \in \mathbb{Z}_q$ where $k = \{1,2,3\}$. It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\vec{v_1} = (a, 0, c)$, $\vec{v_2} = (0, b, c)$ and $\vec{v_3} = (-bc, -ac, ab)$. It then sets $\vec{u_c} = (u_{c,1}, u_{c,2}, u_{c,3})$ where $u_{c,1}, u_{c,2}, u_{c,3}$ are chosen randomly in $\mathbb{Z}_r$. Let $\vec{u^q}$ denote the projection of $\vec{u}$ along the plane formed by $\vec{v_1}$ and $\vec{v_2}$ and $\vec{u^p}$ denote the projection of $\vec{u}$ along $\vec{v_3}$. Let $g^{v^p_{c,k}} = C^{u^p_{c,k}}$. Note that by using this value of $v^p_{c,k}$, we will not be able to generate a column ciphertext that has the right component in V_p; but since all columns are random in V_p, we do not need to generate this term. Let $g^{v'^p_{c,k}} = g^{z \cdot u^p_{c,k}}$, where z is random in $\mathbb{Z}_r$. It also sets , $\vec{v_i} = \tilde{q}_i \cdot \vec{v_1} + \tilde{q}'_i \cdot \vec{v_2} + \tilde{p}_i \cdot \vec{v_3}$ where $\tilde{q}_i, \tilde{q}'_i, \tilde{p}_i$ are random in $\mathbb{Z}_r$. It also chooses random $z_{1,x,k}, z_{2,x}, z_{3,x} \in \mathbb{Z}_q$ where $1 \leq x < i$ and $k \in \{1,2,3\}$.

Then it creates the ciphertext as:

$$
\begin{aligned}
x < i: \quad & R_{x,k} = g^{z_{1,x,k}} : k = \{1,2,3\} \\
& \widetilde{R}_{x,k} = g^{z_{1,x,k}\eta} : k = \{1,2,3\} \\
& A_x = g^{z_{2,x}} \\
& B_x = e(g,g)^{3_{3,x}} \\
x = i: \quad & R_{i,k} = B^{v_{i,k}s_i} : k = \{1,2,3\} \\
& \widetilde{R}_{i,k} = B^{v_{i,k}s_i\eta} : k = \{1,2,3\} \\
& A_i = g^{s_i t(\vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{i}} \cdot \vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{c}})} \cdot C^{s_i t(\vec{\boldsymbol{v}}^{\boldsymbol{p}}_{\boldsymbol{i}} \cdot \vec{\boldsymbol{u}}^{\boldsymbol{p}}_{\boldsymbol{c}})} \\
& B_i = M \cdot e(A,B)^{s_i t(\vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{i}} \cdot \vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{c}})} e(g,T)^{t s_i(\vec{\boldsymbol{v}}^{\boldsymbol{p}}_{\boldsymbol{i}} \cdot \vec{\boldsymbol{u}}^{\boldsymbol{p}}_{\boldsymbol{c}})}
\end{aligned}
$$

For each $x \in \{i+1 \cdots m\}$, it picks $\vec{\boldsymbol{v}}_{\boldsymbol{x}} = \vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{x}} = \tilde{q}_x \cdot \vec{\boldsymbol{v}}_{\boldsymbol{1}} + \tilde{q}'_x \cdot \vec{\boldsymbol{v}}_{\boldsymbol{2}}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$.

$$
\begin{aligned}
x > i: \quad & R_{x,k} = g^{r_x v^q_{x,k} s_x} : k = \{1,2,3\} \\
& \widetilde{R}_{x,k} = g^{r_x v^q_{x,k} s_x \eta} : k = \{1,2,3\} \\
& A_x = g^{s_x t(\vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{x}} \cdot \vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{c}})} \\
& B_x = M \cdot e(g,g)^{\alpha_x s_x t(\vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{x}} \cdot \vec{\boldsymbol{v}}^{\boldsymbol{q}}_{\boldsymbol{c}})}
\end{aligned}
$$

$$
\begin{aligned}
C_{y,k} &= \left(g^{c_y} \cdot A^{-1}\right)^{t(v^q_{c,k} + {v'}^p_{c,k})} \cdot g^{w_{y,k}\eta} : k = \{1,2,3\} \\
\widetilde{C}_{y,k} &= g^{w_{y,k}} : k = \{1,2,3\}
\end{aligned}
$$

If T corresponds to g^{abc}, then the ciphertext corresponding to row i corresponds to the target row; and if T is randomly chosen, then the encryption corresponds to game H_3. The reduction will receive the guess γ from $\mathcal{A}$, and it passes on the same value to the Decision 3-party Diffie Hellman challenger. The advantage of the reduction is exactly equal to the advantage of the adversary $\mathcal{A}$.

C Proof of Claim 6.6

Consider an adversary $\mathcal{A}$ that solves the index hiding game with a probability greater than ε. The adversary is considered successful if it can distinguish between games $H_{3,j}$ and $H_{3,j+1}$. We build a reduction $\mathcal{R}$ that uses $\mathcal{A}$ to solve the Decision 3-party Diffie Hellman problem. The reduction receives the Decision 3-party Diffie Hellman challenge as:

$$\mathbb{G}, g, A = g^a, B = g^b, C = g^c, T$$

and it is expected to guess if T is g^{abc} or if it is random.

Next, in the Setup phase the reduction based on the input (i,j) (the row and column the adversary will attack) sets up the public and the private parameters. The reduction chooses random $r_1, r_2, \ldots r_m$, $c_1, c_2 \ldots c_m$, $\alpha_1, \alpha_2 \ldots \alpha_m \in \mathbb{Z}_r$. It sets up the public parameters as:

$$
\begin{aligned}
& g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots E_m = g^{r_m}, \\
& G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots, G_m = e(g,g)^{\alpha_m}, \\
& H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots H_j = C^{c_j} \ldots, H_m = g^{c_m}
\end{aligned} \tag{6}
$$

And the private key $K_{(x,y)}$ of user (x,y) is:

$$
\begin{aligned}
K_{(x,y)} &= g^{\alpha_x} \cdot g^{r_x \cdot c_y} : y \neq j \\
K_{(x,y)} &= g^{\alpha_x} \cdot C^{r_x \cdot c_y} : y = j
\end{aligned}
$$

Note that the distribution of the public and private parameters matches the distribution of parameters in the real scheme.

In the challenge phase the adversary sends the message $M \in \mathbb{G}_T$ to the reduction. The reduction then chooses random $t, w_{1,k}, w_{2,k}, \ldots w_{m,k}, s_1, s_2 \ldots s_m \in \mathbb{Z}_r$ where $k = \{1,2,3\}$. It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\vec{v_1} = (a, 0, c)$, $\vec{v_2} = (0, b, c)$ and $\vec{v_3} = (-bc, -ac, ab)$.

Set $g^\eta = B$.

It then sets $\vec{v_c} = (v_{c,1}, v_{c,2}, v_{c,3})$ where $v_{c,1}, v_{c,2}, v_{c,3}$ are chosen randomly in $\mathbb{Z}_r$. Let $\vec{v^q}$ denote the projection of $\vec{v}$ along the plane formed by $\vec{v_1}$ and $\vec{v_2}$. And $\vec{v^p}$ be the component along $\vec{v_3}$.

It chooses random $z_{1,x,k}, z_{2,x}, z_{3,x} \in \mathbb{Z}_r$ where $1 \le x < i$ and $k \in \{1,2,3\}$ and sets up the ciphertext as follows.

$$\begin{aligned} x \le i: \quad & R_{x,k} = g^{z_{1,x,k}} : k = \{1,2,3\} \\ & \widetilde{R}_{x,k} = g^{z_{1,x,k}\eta} : k = \{1,2,3\} \\ & A_x = g^{z_{2,x}} \\ & B_x = e(g,g)^{z_{3,x}} \end{aligned} \tag{7}$$

For each $x \in \{i+1 \cdots m\}$, it picks $\vec{v_x} = \vec{v_x^q} = \tilde{q}_x \cdot \vec{v_1} + \tilde{q}'_x \cdot \vec{v_2}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$.

$$\begin{aligned} x > i: \quad & R_{x,k} = g^{r_x v^q_{x,k} s_x} : k = \{1,2,3\} \\ & \widetilde{R}_{x,k} = B^{r_x v^q_{x,k} s_x} : k = \{1,2,3\} \\ & A_x = B^{s_x t (\vec{v_x^q} \cdot \vec{v_c})} \\ & B_x = M \cdot e(g,B)^{\alpha_x s_x t (\vec{v_x^q} \cdot \vec{v_c})} \end{aligned}$$

Choose a random $z \in \mathbb{Z}_r$.

$$\begin{aligned} y < j: \quad & C_{y,k} = g^{z v^p_{c,k}} \cdot B^{w_{y,k}} : k = \{1,2,3\} \\ & \widetilde{C}_{y,k} = g^{-c_y t v^q_{c,k}} \cdot g^{w_{y,k}} : k = \{1,2,3\} \\ y = j: \quad & C_{y,k} = T^{c_y t v^p_{c,k}} \cdot B^{w_{y,k}} : k = \{1,2,3\} \\ & \widetilde{C}_{y,k} = C^{-c_y v^q_{c,k} t} \cdot g^{w_{y,k}} : k = \{1,2,3\} \\ y > j: \quad & C_{y,k} = B^{w_{y,k}} : k = \{1,2,3\} \\ & \widetilde{C}_{y,k} = A^{-c_y v^p_{c,k} t} \cdot g^{-c_y t v^q_{c,k}} \cdot g^{w_{y,k}} : k = \{1,2,3\} \end{aligned}$$

If T corresponds to g^{abc}, then we are in game $H_{3,j}$; and if T is randomly chosen, then the encryption corresponds to the game $H_{3,j+1}$. The reduction will receive the guess γ from $\mathcal{A}$, and it passes on the same value to the Decision 3-party Diffie Hellman challenger. The advantage of the reduction is exactly equal to the advantage of the adversary $\mathcal{A}$.

D Proof of Claim 6.7

Consider an adversary $\mathcal{A}$ that can distinguish between games H_4 and H_5 with a probability greater than ε. We build a reduction $\mathcal{R}$ that uses $\mathcal{A}$ to solve the decisional linear problem. The reduction receives the decisional linear challenge as:

$$\mathbb{G}, g, g^a, g^b, g^c, g^{ax}, g^{by}, T$$

and it is expected to guess if T is $g^{c(x+y)}$ or if it is random.

Next, in the Setup phase the reduction based on the input i (the row the adversary will attack) sets up the public and the private parameters. The reduction chooses random $r_1, r_2, \ldots r_m, c_1, c_2 \ldots c_m, \alpha_1, \alpha_2 \ldots \alpha_m \in \mathbb{Z}_r$. It sets up the public parameters as:

$$\begin{aligned} & g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots E_m = g^{r_m}, \\ & G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots G_m = e(g,g)^{\alpha_m}, \\ & H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots H_m = g^{c_m} \end{aligned} \tag{8}$$

And the private key $K_{(x,y)}$ of user (x,y) is:

$$K_{(x,y)} = g^{\alpha_x} \cdot g^{r_x \cdot c_y} : \forall x, y$$

Note that the distribution of the public and private parameters matches the distribution of parameters in the real scheme.

It sets $g^{v_{1,1}} = g^a$, $g^{v_{1,2}} = g^0$, $g^{v_{1,3}} = g^c$, $g^{v_{2,1}} = g^0$, $g^{v_{2,2}} = g^b$ and $g^{v_{2,3}} = g^c$. A valid decisional linear tuple will lie in the subspace formed by vectors $\vec{\boldsymbol{v_1}}$ and $\vec{\boldsymbol{v_2}}$. A decisional linear problem tuple will be used for setting row ciphertext for row $i+1$. A valid tuple leads to encryption as in game H_4, and a random tuple will cause the encryption to be as in game H_5.

In the challenge phase the adversary sends the message $M \in \mathbb{G}_T$ to the reduction. The reduction then chooses random $t, \eta, w_{1,k}, w_{2,k}, \ldots w_{m,k}, s_1, s_2 \ldots s_m \in \mathbb{Z}_r$ where $k = \{1,2,3\}$. It then sets $\vec{\boldsymbol{v_c}} = (v_{c,1}, v_{c,2}, v_{c,3})$ where $v_{c,1}, v_{c,2}, v_{c,3}$ are chosen randomly in $\mathbb{Z}_r$.

$$g^{(\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})} = \prod_{k=1}^{3} [g^{v_{x,k}}]^{v_{c,k}}$$

It also chooses random $z_{1,x,k}, z_{2,x}, z_{3,x} \in \mathbb{Z}_r$ where $1 \le x \le i$ and $k \in \{1,2,3\}$. Then it creates the ciphertext as follows.

$$\begin{aligned} x \le i: \quad & R_{x,k} = g^{z_{q,x,k}} : k = \{1,2,3\} \\ & \widetilde{R}_{x,k} = g^{z_{1,x,k}\eta} : k = \{1,2,3\} \\ & A_x = g^{z_{2,x}} \\ & B_x = e(g,g)^{z_{3,x}} \end{aligned}$$

It sets $g^{v_{i+1,1}} = g^{ax}$, $g^{v_{i+1,2}} = g^{by}$ and $g^{v_{i+1,3}} = T$. For each $x \in \{i+2 \cdots m\}$, it picks $g^{v_{x,1}} = g^{a\tilde{q}_x}$, $g^{v_{x,2}} = g^{b\tilde{q}'_x}$ and $g^{v_{x,3}} = g^{c(\tilde{q}_x + \tilde{q}'_x)}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$.

$$\begin{aligned} x > i: \quad & R_{x,k} = g^{r_x v_{x,k} s_x} : k = \{1,2,3\} \\ & \widetilde{R}_{x,k} = g^{r_x v_{x,k} s_x \eta} : k = \{1,2,3\} \\ & A_x = g^{s_x t (\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})} \\ & B_x = M \cdot e(g,g)^{\alpha_x s_x t (\vec{\boldsymbol{v_x}} \cdot \vec{\boldsymbol{v_c}})} \end{aligned}$$

$$C_{y,k} = g^{c_y t v_{c,k}} \cdot g^{w_{y,k}\eta} \quad \widetilde{C}_{y,k} = g^{w_{y,k}} : k = \{1,2,3\}$$

If T corresponds to $g^{c(x+y)}$, then the ciphertext corresponds to game H_4; and if T is randomly chosen, then it corresponds to game H_5. The reduction will receive the guess γ from $\mathcal{A}$, and it passes on the same value to the decisional linear challenger. The advantage of the reduction is exactly equal to the advantage of the adversary $\mathcal{A}$.