Building Efficient Fully Collusion-Resilient Traitor Tracing and Revocation Schemes

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

[1] Department of Computer Science, UCLA, USA
{sanjamg,abishekk,sahai}@cs.ucla.edu

[2] Department of Computer Science, UT Austin, USA
bwaters@cs.utexas.edu

Abstract. In [8, 9] Boneh et al. presented the first fully collusion-resistant traitor tracing and trace & revoke schemes. These schemes are based on composite order bilinear groups and their security depends on the hardness of the subgroup decision assumption. In this paper we present new, efficient trace & revoke schemes which are based on prime order bilinear groups, and whose security depend on the hardness of the Decisional Linear Assumption or the External Diffie-Hellman (XDH) assumption. This allows our schemes to be flexible and thus much more efficient than existing schemes in terms a variety of parameters including ciphertext size, encryption time, and decryption time. For example, if encryption time was the major parameter of concern, then for the same level of practical security as [8] our scheme encrypts 6 times faster. Decryption is 10 times faster. The ciphertext size in our scheme is 50% less when compared to [8].
We provide the first implementations of efficient fully collusion-resilient traitor tracing and trace & revoke schemes. The ideas used in this paper can be used to make other cryptographic schemes based on composite order bilinear groups efficient as well.

1 Introduction

Consider a scenario 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 naïve solution to achieve this 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 dishonest user sells his key, then an attacker could build pirate decoders which it could then distribute, *allowing unauthorized users to decrypt all future broadcast content without ever having to communicate with the attacker again.* 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. A *traitor tracing* or *trace & revoke* system is designed to solve this problem. The purpose of a trace & revoke system, introduced by Chor et al. [11], is to help content distributors identify rogue users and revoke their secret keys. If revocation is not desired, one can have just traitor tracing schemes, which helps the distributor identify the keys used in a pirate decoder. The content distributor can then hold the corresponding rogue user responsible for the loss incurred.

It should be observed that a traitor tracing system is not designed to help to protect any particular content. The problem of traitor tracing is distinct from what is commonly referred to as "Digital Rights Management" (DRM). DRM systems have traditionally been concerned with protecting the widespread distribution of content that is *already* in the hands of the (perceived) attacker. Clearly, there are fundamental obstacles to achieving this goal, since the attacker can simply record what he sees and then retransmit this. In a trace & revoke system, an authority can use the tracing mechanisms to identify all of the key material (actively) used in a pirate box and then disable these keys from being used to access *future* broadcasts. The use of trace & revoke systems best fits application such as satellite radio or other active broadcast services where users are interested in having a device that can access the current broadcast, without having to be in constant communication with a dishonest party.[1] Given a pirate decoder, the challenge in a trace &

[1] Traitor tracing systems are not appropriate for systems where "protecting" released content is considered the highest priority.

revoke system is to identify at least one of the users whose key must have been used to construct the pirate decoder and then revoke that key from the system. As such, traitor tracing can be seen as providing a type of cryptographic method for digital forensics – once a decoding box is discovered in the wild, the associated cryptographic tracing algorithm allows one to (provably) associate a particular user's secret key with the box.

A naïve solution to the problem just described (in a system of N users) would be to have N instances of an off-the-shelf encryption system 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[2]. 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 whose key was used to build it. We could then revoke this key by simply not encrypting under it in future broadcasts. But this system is very inefficient. For this system, the ciphertext size is *linear* in the number of users. We provide an efficient implementation of this naïve solution using a fast Elgamal encryption scheme and compare it with the performance of our scheme in Section 7.2.

Previous Work. To overcome this limitation of inefficiency, many results with different levels of security have been proposed. A weak security property that has been the subject of the greatest amount of previous work is the *t-collusion-resistant traitor tracing.* A t-collusion-resistant tracing [3,10,12,17,18,20,22,25] system will work as long as the pirate uses fewer than t user keys in building the pirate box. Prior to [8], all such schemes required a ciphertext size blow-up at least linear in this parameter t.

A system that allows for traitor tracing regardless of how many users' keys are captured by the attacker is called *fully collusion-resistant.* Boneh, Sahai, and Waters [8] presented the first fully collusion-resistant traitor tracing system with $O(\sqrt{N})$ size ciphertexts and public keys. A fully-collusion-resistant traitor tracing system with constant size ciphertexts [7] has also been constructed, but at the cost of enormous private key sizes (quadratic in the number of users).

Another issue of concern in traitor tracing systems is the need for a tracing authority, e.g [7, 8] which use a secret tracing key to identify rogue users. [10, 16, 23, 24, 26] allow for a public tracing algorithm that does not require any secret inputs. Other systems such as the one in [4, 9] provide security only against a static adversary and achieve $O(1)$ size ciphertext and private key, but need $O(N)$ size public key (which is used in the decryption algorithm).

When considering only broadcast encryption, [9] acheive adaptive security with $O(1)$ size ciphertext and private key ($O(N)$ size public key) and also provide a system with $O(\sqrt{N})$ ciphertext and public key. [15] obtain adaptively secure broadcast encryption with $O(1)$ cipher-text, $O(N)$ private and public key. The recent work of [27] obtains identical parameters and also provides identity based encryption.

Building on [8], Boneh and Waters [9] presented a fully collusion resistant, publicly traceable trace & revoke scheme, representing the "state-of-the-art" prior to this work. However, [9] crucially makes use of composite order bilinear groups, which lead to significant losses in efficiency that make the scheme impractical in many settings. The goal of the present work is to build new techniques to achieve order-of-magnitude improvements in efficiency without sacrificing any security.

Our Contribution. We present a new traitor tracing system that achieves the same strong security properties as [8], but avoids the use of composite order bilinear groups. Instead, using new techniques, our scheme is based on prime order bilinear groups, and its security depends on the hardness of the widely believed decisional linear assumption. This allows for shorter group elements and much more efficient schemes (see Section 7). We also extend this to build publicly traceable trace & revoke schemes, improving similarly in efficiency over [9].

Hardness assumptions in composite order bilinear groups are limited by known attacks on factoring their modulii. Because of sub-exponential attacks against factoring, for appropriate security, large composite order groups must be used. When compared with prime order bilinear groups, for the same level of practical security (see Section 7 for details), 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 main contribution of this research is to present traitor tracing schemes based on prime order bilinear groups making them practical.

[2] Note that here, the content itself would be a secret key for a private-key encryption scheme (such as AES), which would then be used to encrypt the actual content.

We also implement our protocol using the PBC library [19] (see Section 7). We compare the efficiency our traitor tracing scheme with an implementation of [8]. We obtain encryption times up to 6 times better than [8] and ciphertexts that are 50% smaller. Decryption is 10 times faster.

We note that the techniques we use are general and can be used to convert other cryptosystems based on composite order groups to ones based on prime order bilinear groups. In this respect, our work is similar to generic methods described in a very recent concurrent and independent work by Freeman [13]. However, our schemes are different from the work of [13]. His work focuses on generality and while our work is on optimizing and implementing efficient traitor tracing systems. He provides a traitor tracing scheme using asymmetric bilinear groups while we provide schemes based on both symmetric and asymmetric groups. Also, our asymmetric construction is more efficient than his construction, which does not have any known implementation.

2 Preliminary Definitions

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. A publicly traceable scheme is one where the $Trace$ algorithm has no secret inputs, i.e there are no tracing secret keys.

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 Trace & Revoke

A Trace & Revoke system is a traitor tracing system that provides an additional property of user revocation. Once a set of rogue users are identified, the system allows for all honest parties to encrypt to the rest of the honest users securely. The system 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. The *Encrypt* algorithm can be used to encrypt a message to *any* subset of users of the system. *Decrypt* is used to decrypt a valid ciphertext. In a secure Trace & Revoke system, the *Decrypt* algorithm of a user succeeds if and only if the encryption was intended for him (he belongs to the set of users that the message was encrypted to). The $Trace$ algorithm is used to identify the key used inside a pirate decoder.

Boneh et al. [8] introduce a new primitive, *Private Linear Broadcast Encryption* (PLBE) and showed that a PLBE is sufficient for implementing a fully collusion-resistant traitor tracing scheme. In this paper, we give an informal treatment (see [8] for details) of traitor tracing systems and their relation to PLBE and present an improved PLBE scheme. However, we recall details on PLBE definitions and its security properties.

Boneh and Waters [9] introduce a new primitive, *Augmented Broadcast Encryption* (AugBE) and use an AugBE scheme (based on composite order bilinear groups) to implement a fully collusion-resistant trace & revoke scheme, secure against adaptive adversaries. We present an improved AugBE scheme based only on prime order groups.

2.3 PLBE

A *Private Linear Broadcast Encryption* (PLBE) system consists of four algorithms: $Setup_{PLBE}$, $Encrypt_{PLBE}$, $Decrypt_{PLBE}$, $TrEncrypt_{PLBE}$. The algorithms described below are similar to the BSW PLBE system [8] except that our system *does not* need a tracing key.

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

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 $K_1, K_2 \ldots K_N$ 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 $C \overset{\$}{\leftarrow} Encrypt_{PLBE}(PK, M)$, and as $C \overset{\$}{\leftarrow} TrEncrypt_{PLBE}(PK, 1, M)$ 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 $K_1, K_2 \ldots K_{i-1}, K_{i+1} \ldots K_N$ 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 $C \overset{\$}{\leftarrow} TrEncrypt_{PLBE}(PK, i + \beta, M)$ 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 $K_1, K_2 \ldots K_N$ 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 $C \overset{\$}{\leftarrow} TrEncrypt_{PLBE}(PK, N+1, M_\beta)$ 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 1. *An N-user PLBE system is considered secure if for all polynomial time adversaries Adv_{ID}, $Adv_{IH}[i]$ for all $i \in \{1 \ldots N\}$ and Adv_{MH} are negligible in the security parameter λ.*

2.4 AugBE

An *Augmented Broadcast Encryption* (AugBE) [9] system consists of three algorithms: $Setup_{AugBE}$, $Encrypt_{AugBE}$, $Decrypt_{AugBE}$.

- $(PK, K_1, K_2 \dots K_N) \overset{\$}{\leftarrow} Setup_{AugBE}(\lambda)$: $Setup_{AugBE}$ 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. N is the number of users in the system.
- $C \overset{\$}{\leftarrow} Encrypt_{AugBE}(S, PK, i, M)$: This algorithm takes as input a subset $S \subseteq \{1, \dots, N\}$ of users, the public key PK, and an index $1 \leq i \leq N + 1$, and a message M. The algorithms outputs a ciphertext which can be decrypted by any user belonging to the set $S \cap \{i, \dots, N\}$. the ciphertext.
- $M \leftarrow Decrypt_{AugBE}(S, j, K_j, C, PK)$: A user j having access to the private key K_j can decrypt a ciphertext C and obtain the corresponding message M. If he is not able to decrypt he outputs $\perp$.

AugBE and PLBE system consists of similar algorithms. The only difference between the AugBE and PLBE systems is that PLBE algorithms do not take set S as input. The set of all users is implied each time set S is referred to. We refer the reader to [8] for further details.

Desired Security Properties. We now describe the security properties required of an AugBE system. The security properties required of a PLBE system are implied by the ones for an AugBE system under the condition that the set S is the set of all users. An AugBE system is considered secure if no adversary has significant advantage in the following games:

- **Index Hiding:** This property prevents an adversary from distinguishing between $Encrypt_{AugBE}(S, PK, i, M)$ and $Encrypt_{AugBE}(S, i+1, PK, M)$ when the adversary knows all the secret keys except the i^{th} secret key. Also when $i \notin S$, an adversary with access to all the private keys in the system, should not be able to tell if the encryption has been done to index i or $i + 1$. 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 $Setup_{AugBE}$ 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.
  - **Query:** The adversary outputs a bit $s' \in \{0, 1\}$. If $s' = 1$, the challenger sends the adversary K_i, else he does nothing.
  - **Challenge:** The adversary sends a message M and a set $S \subseteq \{1, \dots, N\}$ to the challenger. The only restriction is if $s' = 1$ then $i \notin S$. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. It sets the ciphertext as $C \overset{\$}{\leftarrow} Encrypt_{AugBE}(S, PK, i + \beta, M)$ 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 $Setup_{AugBE}$ 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 and a set $S \subset \{1, \dots, N\}$ to the challenger. The challenger flips an unbiased coin and obtains a random $\beta \in \{0, 1\}$. It sets the ciphertext as $C \overset{\$}{\leftarrow} Encrypt_{AugBE}(S, PK, N + 1, M_\beta)$ 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. *An N-user AugBE 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.*

2.5 Equivalence of Traitor Tracing and PLBE

We have presented an intuition behind the argument. A more formal argument appears in [8]. 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

3.1 Bilinear Groups

Symmetric and Asymmetric Bilinear Groups of Prime Order. Consider three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime orders (possibly different). Let g_1 be a generator of $\mathbb{G}_1$ and g_2 a generator of $\mathbb{G}_2$. Let r be the order of G_1, the smaller of the two groups. We define an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with the properties: (1) e is non-degenerate: $e(g_1, g_2)$ should not evaluate to the identity element of $\mathbb{G}_T$. (2) The map is bilinear: $\forall u \in \mathbb{G}_1$, $\forall v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_r$ we should have $e(u^a, v^b) = e(u, v)^{ab}$. Such groups are refereed to as *Asymmetric Bilinear Groups of Prime Order.* Bilinear groups in which $\mathbb{G}_1 = \mathbb{G}_2 \equiv \mathbb{G}$ are called *Symmetric Bilinear Groups of Prime Order.* It can be seen that for such groups the 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 similar to the ones of prime order. The key 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. We will use $\mathbb{G}_p$ and $\mathbb{G}_q$ to denote the order 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 PLBE scheme [8]. 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 in [1] 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.

External Diffie Hellman Assumption. The External Diffie Hellman (XDH) assumption states that the Decisional Diffie Hellman (DDH) assumption is hard in the group G_1. (Not necessarily hard in G_2). This assumption is believed to be true in asymmetric pairings generated using special MNT curves [2,21].

Subgroup Decision Assumption. This problem was introduced by Boneh et al. [5] 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 $\mathbb{G}_q$. This assumption is for composite order groups. We do not use this assumption in this work.

4 Key Ideas

We now present the intuition behind the working of [8] for composite order bilinear groups and provide a generic construction to achieve the same properties using prime order bilinear groups. 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 [8] (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 $\boldsymbol{v_1} = (a, b)$ and $\boldsymbol{v_2} = (-b, a)$. 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[3] 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 $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = \boldsymbol{v_1} \times \boldsymbol{v_2}$, 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 [1].

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 from 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 [8] 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.

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

5 Our Construction

In this paper we present two new traitor tracing schemes and corresponding trace & revoke systems. As already pointed out in section 3 a PLBE scheme is sufficient to construct a traitor tracing system and an AugBE scheme is sufficient to construct a trace & revoke system. In this section we present our PLBE and AugBE improving on the previous schemes [8,9]. The schemes in the symmetric and the asymmetric prime order bilinear groups are fundamentally different. It should be noted that all our schemes allow for public traceability. The PLBE schemes can be obtained by dropping certain terms from the AugBE scheme which we describe towards the end of the section.

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 $Encrypt_{AugBE}$ consists of a ciphertext component for every row and a component for every column. For each row x the ciphertext consists of $(A_x,\ B_x,\ \boldsymbol{R_x},\ \widetilde{\boldsymbol{R}}_{\boldsymbol{x}})$ and for every column y the ciphertext consists of $(\boldsymbol{C_y},\ \widetilde{\boldsymbol{C}}_{\boldsymbol{y}})$.

Fully collusion resistant traitor tracing (or trace & revoke) is hard because we need to garble parts of the ciphertext making sure that it only impacts a certain subset of the users. This is made possible by having a ciphertext term have components along different subspaces. For the purposes of this paper, we use the notation V to represent the space of ciphertext elements. The elements in this space can have orthogonal components along V_q and V_p. The information about the sub-space V_q is public while the information for V_p is private.

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 $x = i$, 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.

In the AugBE scheme in addition to the above properties there is a set S that specifies the set of users to which encryption is done. In other words only users in that set can decrypt.

5.1 AugBE using Symmetric Bilinear Groups

We introduce some notation before we go further and describe the scheme. For a given vector $\boldsymbol{v} = (v_1, \ldots v_i)$, by $g^{\boldsymbol{v}}$ we mean the vector $(g^{v_1} \ldots g^{v_i})$. A pairing e on two vectors $\boldsymbol{R}$ and $\boldsymbol{C}$ is defined by multiplication after the componentwise pairing operation, i.e. $e(\boldsymbol{R}, \boldsymbol{C}) = \prod_{k=1}^{i} e(R_k, C_k)$, where e is the pairing operation on the underlying group elements. Given a set S of users to which encryption is to be done let $S_x = \{y : (x, y) \in S\}$.

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

- $(PK, K_{(1,1)}, \cdots K_{(1,m)}, K_{(2,1)} \cdots K_{(m,m)}) \leftarrow Setup_{AugBE}\ (1^\lambda, N = m^2)$
  The $Setup_{AugBE}$ algorithm takes as input the security parameter λ and the number of users N in the system. The algorithm generates a prime order groups $\mathbb{G}$ with a pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$. It outputs, g the generator of $\mathbb{G}$ and let r (depends on the security parameter) denote the size of $\mathbb{G}$. It then chooses

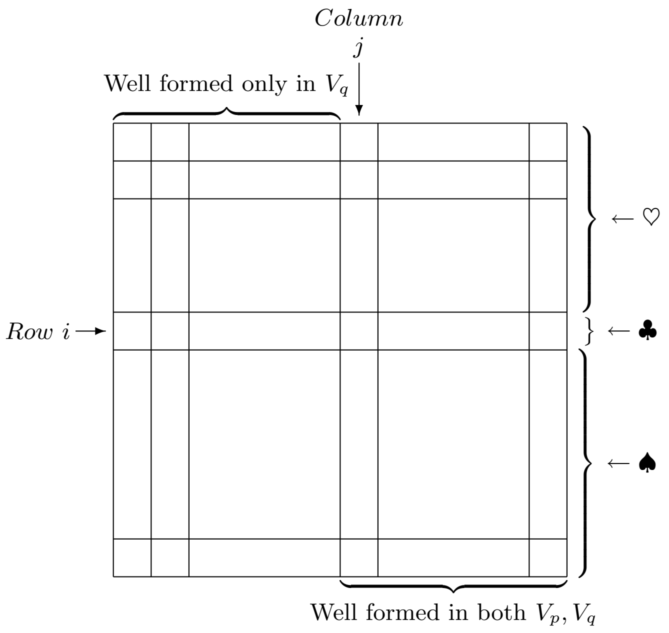


Fig. 1. $\heartsuit$ stands for "Random," $\clubsuit$ stands for "Well formed in V_p and V_q," and $\spadesuit$ stands for "Well formed in V_q."

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 AugBE system (along with the group description) is set to:

$$\begin{aligned}
&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots, E_m = g^{r_m},\\
&G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots, G_m = e(g,g)^{\alpha_m},\\
&g, H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots, H_m = g^{c_m}\\
&u_1, u_2, \ldots, u_m \in \mathbb{G}.
\end{aligned}$$

The secret key of each user (x,y) is $K_{(x,y)} = \{g^{\alpha_x} \cdot g^{r_x c_y} \cdot u_y^{\sigma_{x,y}}, g^{\sigma_{x,y}}, \forall i, (i \neq y), u_y^{\sigma_{x,y}}\}$.

– $C \leftarrow Encrypt_{AugBE}(PK, S, (i,j), M)$

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 and belonging to the set S. The algorithm chooses random t, η, $s_1, s_2 \ldots s_m \in \mathbb{Z}_r$. It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = \boldsymbol{v_1} \times \boldsymbol{v_2}$. All elements $g^{\boldsymbol{v}}$ when $\boldsymbol{v}$ is a linear combination of $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ define the V_q space. These elements define the space V_p when the vector $\boldsymbol{v}$ is parallel to $\boldsymbol{v_3}$. Choose $\boldsymbol{w_1}, \boldsymbol{w_2}, \ldots, \boldsymbol{w_m}, \boldsymbol{v_c} \in \mathbb{Z}_r^3$. Let $\boldsymbol{v'_c} = \boldsymbol{v_c} + v_{cr} \cdot \boldsymbol{v_3}$ be another vector, with v_{cr} randomly chosen from $\mathbb{Z}_r$.

For each row, $1 \leq x < i$, choose random $\boldsymbol{z_x} \in \mathbb{Z}_r^3$ and $a_x, b_x \in \mathbb{Z}_r$. The row cipher text components are,

$$\begin{aligned}
&\boldsymbol{R_x} = g^{\boldsymbol{z_x}} && \widetilde{\boldsymbol{R}}_{\boldsymbol{x}} = g^{\eta \boldsymbol{z_x}}\\
&A_x = g^{a_x} && B_x = G_x^{b_x}\\
&T_x = (\prod_{k \in S_x} u_k)^{a_x}
\end{aligned}$$

For row, $x = i$, pick random $\boldsymbol{v_i}$ randomly $\in \mathbb{Z}_r^3$. Note that $\boldsymbol{v_i} \cdot \boldsymbol{v'_c} \neq \boldsymbol{v_i} \cdot \boldsymbol{v_c}$. This prevents parties (i,y), with $y < j$ from decrypting the message.

The row cipher text component for $x = i$ is,

$$\boldsymbol{R_i} = g^{r_i s_i \boldsymbol{v_i}} \qquad \widetilde{\boldsymbol{R}}_{\boldsymbol{i}} = g^{\eta r_i s_i \boldsymbol{v_i}}$$
$$A_i = g^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})} \qquad B_i = M \cdot G_i^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})}$$
$$T_i = (\prod_{k \in S_i} u_k)^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})}$$

For rows, $x > i$, pick random $\boldsymbol{v_x} = \tilde{q}_x \boldsymbol{v_1} + \tilde{q}'_x \boldsymbol{v_2}$ where $\tilde{q}_x, \tilde{q}'_x$ are random $\in \mathbb{Z}_r$. Note that $\boldsymbol{v_x} \cdot \boldsymbol{v'_c} = \boldsymbol{v_x} \cdot \boldsymbol{v_c}$. This allows all parties (x, y) to decrypt the message, if $x > i$.
The row cipher text components for all $x > i$ are,

$$\boldsymbol{R_x} = g^{r_x s_x \boldsymbol{v_x}} \qquad \widetilde{\boldsymbol{R}}_{\boldsymbol{x}} = g^{\eta r_x s_x \boldsymbol{v_x}}$$
$$A_x = g^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})} \qquad B_x = M \cdot G_x^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})}$$
$$T_x = (\prod_{k \in S_x} u_k)^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})}$$

And for every column $y < j$, the column ciphertext components are,

$$\boldsymbol{C_y} = g^{c_y t \boldsymbol{v'_c}} \cdot g^{\eta \boldsymbol{w_y}} \qquad \widetilde{\boldsymbol{C}}_{\boldsymbol{y}} = g^{\boldsymbol{w_y}}$$

And for every column $y \geq j$, the column ciphertext components are,

$$\boldsymbol{C_y} = g^{c_y t \boldsymbol{v_c}} \cdot g^{\eta \boldsymbol{w_y}} \qquad \widetilde{\boldsymbol{C}}_{\boldsymbol{y}} = g^{\boldsymbol{w_y}}$$

– $M \leftarrow Decrypt_{AugBE}(C, S, K_{(x,y)}, (x, y))$
Let $K'_{(x,y)} = g^{\alpha_x} g^{r_x c_y} \prod_{k \in S_x} u_k^{\sigma_{x,y}}$ be the key used by recipient (x, y). Note that user (x, y) can always compute the product when $y \in S_x$ and cannot compute this product otherwise.

$$M = \frac{B_x}{\frac{e(K'_{(x,y)}, A_x)}{e(T_x, g^{\sigma_{x,y}})}} \cdot \frac{e(\boldsymbol{R_x}, \boldsymbol{C_y})}{e(\widetilde{\boldsymbol{R}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{C}}_{\boldsymbol{y}})}$$

The broadcast encryption procedure is to obtained by encrypting using $Encrypt_{AugBE}(PK, 0, M)$. This illustrates the public traceability of our system.The correctness of decryption follows by inspection.

5.2 AugBE using Asymmetric Bilinear Groups

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

– $(PK, K_{(1,1)}, \cdots K_{(1,m)}, K_{(2,1)} \cdots K_{(m,m)}) \leftarrow Setup_{AugBE}\ (1^\lambda, N = m^2)$
The $Setup_{AugBE}$ algorithm takes as input the security parameter λ and the number of users N in the system. The algorithm generates two prime order groups $\mathbb{G}_1, \mathbb{G}_2$ with a pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_{\mathbb{T}}$. It outputs, g_1 and g_2, generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ and let r (depends on the security parameter) denote the size of $\mathbb{G}_1$. 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 AugBE system (along with the group description) is set to:

$$g_1, E_1 = g_1^{r_1}, E_2 = g_1^{r_2}, \ldots, E_m = g_1^{r_m},$$
$$G_1 = e(g_1, g_2)^{\alpha_1}, G_2 = e(g_1, g_2)^{\alpha_2}, \ldots, G_m = e(g_1, g_2)^{\alpha_m},$$
$$g_2, H_1 = g_2^{c_1}, H_2 = g_2^{c_2}, \ldots, H_m = g_2^{c_m}$$
$$u_1, u_2, \ldots, u_m \in \mathbb{G}_2.$$

The secret key of each user (x, y) is $K_{(x,y)} = \{g_2^{\alpha_x} \cdot g_2^{r_x c_y} \cdot u_y^{\sigma_{x,y}}, g_2^{\sigma_{x,y}}, \forall i, (i \neq y), u_y^{\sigma_{x,y}}\}$.

– $C \leftarrow Encrypt_{AugBE}(PK, S, (i,j), M)$
The algorithm chooses random $t, \eta, s_1, s_2 \dots s_m \in \mathbb{Z}_r$ and $\boldsymbol{w_1}, \boldsymbol{w_2}, \dots, \boldsymbol{w_m}, \boldsymbol{v_1}, \boldsymbol{v_c} \in \mathbb{Z}_r^2$. Let $\boldsymbol{v_2}$ be a random vector $\in \mathbb{Z}_r^2$ such that $\boldsymbol{v_1} \cdot \boldsymbol{v_2} = 0$. Let $\boldsymbol{v_c'} = \boldsymbol{v_c} + v_{cr} \cdot \boldsymbol{v_2}$ be another vector, with v_{cr} randomly chosen from $\mathbb{Z}_r$. Note that, $\boldsymbol{v_c'} \cdot \boldsymbol{v_1} = \boldsymbol{v_c} \cdot \boldsymbol{v_1}$, a key property we will use in the correctness of our scheme. All elements $g^{\boldsymbol{v}}$ when $\boldsymbol{v}$ is a along $\boldsymbol{v_1}$ define the V_q space. These elements belong to the space V_p when the vector $\boldsymbol{v}$ is parallel to $\boldsymbol{v_2}$. Based on the XDH assumption the details about the V_p space are private. For each row, $1 \le x < i$, pick random $\boldsymbol{z_x} \in \mathbb{Z}_r^2$ and $a_x, b_x \in \mathbb{Z}_r$. The row cipher text components are,

$$\boldsymbol{R_x} = g_1^{\boldsymbol{z_x}} \qquad \widetilde{\boldsymbol{R}}_{\boldsymbol{x}} = g_1^{\eta \boldsymbol{z_x}}$$
$$A_x = g_1^{a_x} \qquad B_x = G_x^{b_x}$$
$$T_x = (\prod_{k \in S_x} u_k)^{a_x}$$

For row, $x = i$, pick random vector $\boldsymbol{v_i} \in \mathbb{Z}_r^2$. Note that $\boldsymbol{v_i} \cdot \boldsymbol{v_c'} \neq \boldsymbol{v_i} \cdot \boldsymbol{v_c}$. This is prevent parties (i, y), with $y < j$ from decrypting the message.
The row cipher text component for $x = i$ is,

$$\boldsymbol{R_i} = g_1^{r_i s_i \boldsymbol{v_i}} \qquad \widetilde{\boldsymbol{R}}_{\boldsymbol{i}} = g_1^{\eta r_i s_i \boldsymbol{v_i}}$$
$$A_i = g_2^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})} \qquad B_i = M \cdot G_i^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})}$$
$$T_i = (\prod_{k \in S_i} u_k)^{s_i t(\boldsymbol{v_i} \cdot \boldsymbol{v_c})}$$

For rows, $x > i$, pick random $v_x' \in \mathbb{Z}_r$, let $\boldsymbol{v_x} = v_x' \cdot v_1$. Note that $\boldsymbol{v_x} \cdot \boldsymbol{v_c'} = \boldsymbol{v_x} \cdot \boldsymbol{v_c}$. This allows all parties (x, y), for all values of y to decrypt the message, if $x > i$.
The row cipher text components for all $x > i$ are,

$$\boldsymbol{R_x} = g_1^{r_x s_x \boldsymbol{v_x}} \qquad \widetilde{\boldsymbol{R}}_{\boldsymbol{x}} = g_1^{\eta r_x s_x \boldsymbol{v_x}}$$
$$A_x = g_2^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})} \qquad B_x = M \cdot G_x^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})}$$
$$T_x = (\prod_{k \in S_x} u_k)^{s_x t(\boldsymbol{v_x} \cdot \boldsymbol{v_c})}$$

And for every column $y < j$, the column ciphertext components are,

$$\boldsymbol{C_y} = g_2^{c_y t \boldsymbol{v_c'}} \cdot g_2^{\eta \boldsymbol{w_y}} \qquad \widetilde{\boldsymbol{C}}_{\boldsymbol{y}} = g_2^{\boldsymbol{w_y}}$$

And for every column $y \ge j$, the column ciphertext components are,

$$\boldsymbol{C_y} = g_2^{c_y t \boldsymbol{v_c}} \cdot g_2^{\eta \boldsymbol{w_y}} \qquad \widetilde{\boldsymbol{C}}_{\boldsymbol{y}} = g_2^{\boldsymbol{w_y}}$$

– $M \leftarrow Decrypt_{PLBE}(C, S, K_{(x,y)}, (x, y))$
Let $K'_{(x,y)} = g_2^{\alpha_x} g_2^{r_x c_y} \prod_{k \in S_x} u_k^{\sigma_{x,y}}$ be the key used by recipient (x, y). Note that user (x, y) can always compute the product when $y \in S_x$ and cannot compute this product otherwise.

$$M = \frac{B_x}{\frac{e(A_x, K'_{(x,y)})}{e(T_x, g_2^{\sigma_{x,y}})}} \cdot \frac{e(\boldsymbol{R_x}, \boldsymbol{C_y})}{e(\widetilde{\boldsymbol{R}}_{\boldsymbol{x}}, \widetilde{\boldsymbol{C}}_{\boldsymbol{y}})}$$

The normal encryption procedure is to just encrypt to $Encrypt_{AugBE}(PK, 0, M)$. This illustrates the public traceability of our system. The correctness of decryption follows by inspection.

5.3 PLBE

The two AugBE schemes based on symmetric and asymmetric prime order groups respectively can be converted to the corresponding PLBE schemes by removing the u terms from the public key. We will also need to get rid of the u and σ terms in the secret key. Row ciphertexts will not include T_x terms and decryption will not require a pairing corresponding to the term T_x. Rest of the parts of the scheme remain the same. Details can be found in an earlier version [14] of this paper.

6 Security Proof

Here we only give the proof for the AugBE scheme using symmetric prime order bilinear groups. The proof for the AugBE scheme based on asymmetric prime order bilinear groups is also similar. The only difference is that security depends on the XDH assumption. The security of the PLBE schemes is implied by the security of the AugBE schemes.

6.1 Index Hiding

Theorem 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 1 and the second case follows by Lemma 2.

Lemma 1. *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 [8]. 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.

In this system two cases arise. The simulator guesses the challenge value s' and generates the public parameters correctly. In case the value of the s' does not match the value later provided by the adversary then the simulator aborts. Since the simulator will successfully guess the right value of s' with probability at least 1/2 the simulation will work with probability at least 1/2.

Case 1: $s' = 0$**:** 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 $r_1, r_2, \ldots r_m, c_1, c_2 \ldots c_m, \alpha_1, \alpha_2 \ldots \alpha_m, \delta_1, \delta_2 \ldots \delta_m \in \mathbb{Z}_r$. It also chooses $\sigma_{x,y} \in \mathbb{Z}_r$ for every $x, y \in \{1 \ldots m\}$. It sets up the public parameters as:

$$\begin{aligned}
&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots E_i = B^{r_i} \ldots, E_m = g^{r_m},\\
&G_1 = e(g, g)^{\alpha_1}, G_2 = e(g, g)^{\alpha_2}, \ldots, G_m = e(g, g)^{\alpha_m},\\
&H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots H_j = C^{c_j} \ldots, H_m = g^{c_m}\\
&u_1 = g^{\delta_1}, u_2 = g^{\delta_2} \ldots u_m = g^{\delta_m}
\end{aligned}$$

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

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

The private keys also contain $\{g^{\sigma_{x,y}}, \forall i, (i \neq y), u_y^{\sigma_{x,y}}\}$ for each user (x, y). 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 $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = (-bc, -ac, ab)$.

Set $g^{\eta} = B$.

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

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.

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

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

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

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

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

Choose a random $z \in \mathbb{Z}_r$ and for $k = \{1,2,3\}$.

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

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 $\mathcal{A}$.

Case 2: $s' = 1$: 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 $r_1, r_2, \ldots r_m, c_1, c_2 \ldots c_m, \alpha_1, \alpha_2 \ldots \alpha_m, \delta_1, \delta_2 \ldots \delta'_j \ldots \delta_m \in \mathbb{Z}_r$. It also chooses $\sigma_{x,y} \in \mathbb{Z}_r$ for every $x, y \in$

$\{1\dots m\}\&y \neq j$. It also chooses $\sigma'_{(x,j)} \in \mathbb{Z}_r$ for every $x \in \{1\dots m\}$. It sets up the public parameters as:

$$\begin{aligned}
&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \dots E_i = B^{r_i} \dots, E_m = g^{r_m},\\
&G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \dots, G_m = e(g,g)^{\alpha_m},\\
&H_1 = g^{c_1}, H_2 = g^{c_2}, \dots H_j = C^{c_j} \dots, H_m = g^{c_m}\\
&u_1 = g^{\delta_1}, u_2 = g^{\delta_2} \dots u_j = C^{\delta'_j} \dots u_m = g^{\delta_m}
\end{aligned} \tag{2}$$

In our system $\delta_j = c \cdot \delta'_j$. Set $\sigma_{(i,j)} = -\frac{br_x c_y}{\delta'_j} + \frac{\sigma'_{(i,j)}}{\delta'_j}$. And the private key $K_{(x,y)}$ of user (x,y) is:

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

The secret key also contain $\forall i, (i \neq y), u_y^{\sigma_{x,y}}$ for each user (x,y). 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 $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = (-bc, -ac, ab)$.

Set $g^{\eta} = B$.

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

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.

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

It sets $\boldsymbol{v_i} = \tilde{q}_i \cdot \boldsymbol{v_3} + \tilde{q}'_i \cdot \boldsymbol{v_3} + \tilde{p}_i \cdot \boldsymbol{v_3}$ where $\tilde{q}_i, \tilde{q}'_i, \tilde{p}_i$ are random in $\mathbb{Z}_r$. The ciphertext for row i can be easily generated because we do not need to use δ_j.

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

For each $x \in \{i+1 \cdots m\}$, it picks $\boldsymbol{v_x} = \boldsymbol{v_x^q} = \tilde{q}_x \cdot \boldsymbol{v_1} + \tilde{q}'_x \cdot \boldsymbol{v_2}$ where $\tilde{q}_x, \tilde{q}'_x$ are random in $\mathbb{Z}_r$. The terms with δ_j can be separated and evaluated as all values $\sigma_{(x,y)}$ for $x > i$ are known.

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

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 $\mathcal{A}$.

Lemma 2. *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 lemmas that establish the indistinguishability of the following games.

– H_1 Encrypt to column[4] m, row i is the target row,[5] row $i+1$ is the greater-than row.[6]
– 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 Lemmas 3, 4, 5, and 6, implies the lemma.

Lemma 3. *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 lemma can be proved by applying the result of Lemma 1.

Lemma 4. *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 $\boldsymbol{v^p_c}$ part of $\boldsymbol{v_c}$. Since all columns have a random component in V_p, we don't need to actually generate this part. 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 $r_1, r_2, \ldots r_{i-1}, r_{i+1} \ldots r_m$, $c_1, c_2 \ldots c_m$, $\alpha_1, \alpha_2 \ldots \alpha_{i-1}, \alpha_{i+1} \ldots\ \alpha_m, \delta_1, \delta_2 \ldots \delta_m \in \mathbb{Z}_r$. It also chooses $\sigma_{x,y} \in \mathbb{Z}_r$ for every $x, y \in \{1 \ldots m\}$. It sets

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

$g^{\alpha_x} = g^{a \cdot b}$ 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:

$$\begin{aligned}
&g, E_1 = g^{r_1} \dots E_i = B \dots E_m = g^{r_m}, \\
&G_1 = e(g,g)^{\alpha_1}, \dots G_i = e(A,B), \dots, G_m = e(g,g)^{\alpha_m}, \\
&H_1 = g^{c_1} \cdot A^{-1}, H_2 = g^{c_2} \cdot A^{-1} \dots H_m = g^{c_m} \cdot A^{-1} \\
&u_1 = g^{\delta_1}, u_2 = g^{\delta_2} \dots u_m = g^{\delta_m}
\end{aligned} \tag{4}$$

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

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

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, $w_{1,k}$, $w_{2,k}$, $\dots w_{m,k}$, s_1, s_2 $\dots s_m \in \mathbb{Z}_q$ where $k = \{1,2,3\}$. It also chooses random $a, b, c \in \mathbb{Z}_r$ and sets $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = (-bc, -ac, ab)$. It then sets $\boldsymbol{u_c} = (u_{c,1}, u_{c,2}, u_{c,3})$ where $u_{c,1}, u_{c,2}, u_{c,3}$ are chosen randomly in $\mathbb{Z}_r$. Let $\boldsymbol{u^q}$ denote the projection of $\boldsymbol{u}$ along the plane formed by $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ and $\boldsymbol{u^p}$ denote the projection of $\boldsymbol{u}$ along $\boldsymbol{v_3}$. 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 , $\boldsymbol{v_i} = \tilde{q}_i \cdot \boldsymbol{v_1} + \tilde{q}'_i \cdot \boldsymbol{v_2} + \tilde{p}_i \cdot \boldsymbol{v_3}$ 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:

$$\begin{aligned}
x < i: \quad & R_{x,k} = g^{z_{1,x,k}} : k = \{1,2,3\} \\
& \widetilde{R}_{x,k} = g^{z_{1,x,k}\eta} : k = \{1,2,3\} \\
& A_x = g^{z_{2,x}} \\
& B_x = e(g,g)^{3_{3,x}} \\
& T_x = A_x{}^{\sum_{k \in S_x} \delta_k} \\
x = i: \quad & R_{i,k} = B^{v_{i,k} s_i} : k = \{1,2,3\} \\
& \widetilde{R}_{i,k} = B^{v_{i,k} s_i \eta} : k = \{1,2,3\} \\
& A_i = g^{s_i t (\boldsymbol{v_i^q} \cdot \boldsymbol{v_c^q})} \cdot C^{s_i t (\boldsymbol{v_i^p} \cdot \boldsymbol{u_c^p})} \\
& B_i = M \cdot e(A,B)^{s_i t (\boldsymbol{v_i^q} \cdot \boldsymbol{v_c^q})} e(g,T)^{t s_i (\boldsymbol{v_i^p} \cdot \boldsymbol{u_c^p})} \\
& T_i = A_i{}^{\sum_{k \in S_i} \delta_k}
\end{aligned}$$

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

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

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

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 $\mathcal{A}$.

Lemma 5. *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. 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 1. 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 $r_1, r_2, \ldots r_m$, $c_1, c_2 \ldots c_m$, $\alpha_1, \alpha_2 \ldots \alpha_m \delta_1, \delta_2 \ldots \delta_m \in \mathbb{Z}_r$. It also chooses $\sigma_{x,y} \in \mathbb{Z}_r$ for every $x, y \in \{1 \ldots m\}$. It sets up the public parameters as:

$$\begin{aligned}
&g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots E_m = g^{r_m}, \\
&G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots, G_m = e(g,g)^{\alpha_m}, \\
&H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots H_j = C^{c_j} \ldots, H_m = g^{c_m} \\
&u_1 = g^{\delta_1}, u_2 = g^{\delta_2} \ldots u_m = g^{\delta_m}
\end{aligned} \tag{5}$$

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

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

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 $\boldsymbol{v_1} = (a, 0, c)$, $\boldsymbol{v_2} = (0, b, c)$ and $\boldsymbol{v_3} = (-bc, -ac, ab)$.

Set $g^{\eta} = B$.

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

It 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.

$$\begin{aligned}
x \leq i : \quad & R_{x,k} = g^{z_{1,x,k}} : k = \{1,2,3\} \\
& \widetilde{R}_{x,k} = g^{z_{1,x,k}\eta} : k = \{1,2,3\} \\
& A_x = g^{z_{2,x}} \\
& B_x = e(g,g)^{z_{3,x}} \\
& T_x = A_x^{\sum_{k \in S_x} \delta_k}
\end{aligned} \tag{6}$$

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

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

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 $\mathcal{A}$.

Lemma 6. *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. 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 $r_1, r_2, \ldots r_m, c_1, c_2 \ldots c_m, \alpha_1, \alpha_2 \ldots \alpha_m \delta_1, \delta_2 \ldots \delta_m \in \mathbb{Z}_r$. It also chooses $\sigma_{x,y} \in \mathbb{Z}_r$ for every $x, y \in \{1 \ldots m\}$. It sets up the public parameters as:

$$\begin{aligned}
& g, E_1 = g^{r_1}, E_2 = g^{r_2}, \ldots E_m = g^{r_m}, \\
& G_1 = e(g,g)^{\alpha_1}, G_2 = e(g,g)^{\alpha_2}, \ldots G_m = e(g,g)^{\alpha_m}, \\
& H_1 = g^{c_1}, H_2 = g^{c_2}, \ldots H_m = g^{c_m} \\
& u_1 = g^{\delta_1}, u_2 = g^{\delta_2} \ldots u_m = g^{\delta_m}
\end{aligned} \tag{7}$$

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

$$K_{(x,y)} = \{g^{\alpha_x} \cdot g^{r_x \cdot c_y} \cdot u_y^{\sigma_{x,y}}, g_2^{\sigma_{x,y}}, \forall i, (i \neq y), u_y^{\sigma_{x,y}}\} : \forall x, y$$

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 $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. 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 $\boldsymbol{v_c} = (v_{c,1}, v_{c,2}, v_{c,3})$ where $v_{c,1}, v_{c,2}, v_{c,3}$ are chosen randomly in $\mathbb{Z}_r$.

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

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

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

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 $\mathbb{Z}_r$.

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

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

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 $\mathcal{A}$.

6.2 Message Hiding

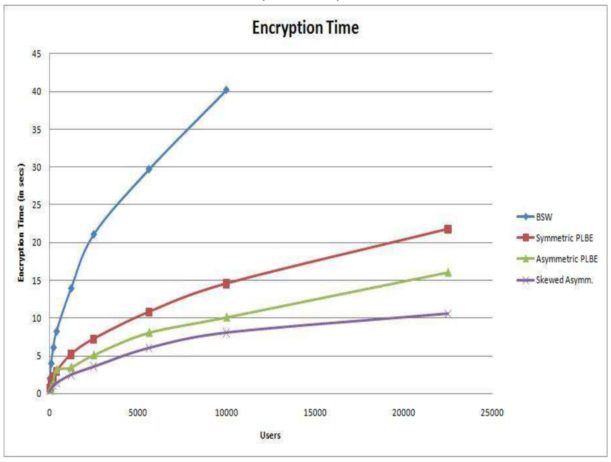
Theorem 2. *No adversary can distinguish between two ciphertexts when the encryption is done to the* $(m+1, 1)$.

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.

7 Implementation

We provide the first implementation of fully collusion resistant traitor tracing and trace & revoke schemes. We use only *prime* order bilinear groups in this implementation. We implement all of our schemes using the Pairing Based Crypto (PBC) library [19]. For schemes that use asymmetric bilinear groups, we generate them using MNT curves [21]. The group size is 170 bits long, the group representations are 512 bits long, and the security is equivalent to 1024 bits of discrete log. It is also believed that the XDH assumption holds on these curves (Section 8.1 [6]). For symmetric bilinear groups, we use super singular curves (with fastest pairing times but bad group element size). We use 512 bit group representations and have 1024 bits of discrete log security. One can choose other alternative symmetric groups that have smaller group size

Fig. 2. Encryption Time (in secs) of traitor tracing schemes



with faster exponentiation but slower pairing operations. This kind of tradeoff was not possible in previous systems [8, 9].

We contrast our schemes' efficiency with an implementation of [8]. [8] only provides traitor tracing functionality. We compare our traitor tracing scheme with [8] in Tables 1,2 and Figure 4. We also provide additional data on our trace & revoke implementation (Table 3)). Currently, the only known way to generate composite order groups is by using symmetric bilinear groups. Also, their subgroup decision assumption mandates that the order of the composite group be at least 1024 bits (to avoid sub-exponential factoring based attacks). We compare the encryption time, decryption time and ciphertext sizes as the number of users grow for all these schemes.

A real implementation of broadcast encryption will use a symmetric key cipher under some key K [8]. But this key K still needs to be distributed and one can use our schemes for key distribution. By converting our encryption system to a Key Encapsulation Mechanism we can save on computation. 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.

In Table 1 and 2 we provide a comparison of our PLBE scheme for the case of symmetric and asymmetric prime order groups with that of [8] (which uses composite order groups). The implementation was done on an Intel i3 2.9GHz quad core desktop PC with 2GB RAM. The groups were chosen to guarantee 1024 bits of discrete log security for encryption time and ciphertext size.

7.1 Encryption Time

The encryption time (Table 1, Figure 2) is heavily dependent on a large number of exponentiation operations, one for each row of ciphertext. It depends on the number of users as $O(\sqrt{N})$, explaining the parabolic nature of the graph(s). The cost of exponentiation operations in elliptic curves depend both on group representation size and the actual order of the group. The order of symmetric groups that we have chosen for this implementation are constructed to be of the form $2^a \pm 2^b \pm 1$, for some integers a, b. This makes exponentiation in them very efficient. The asymmetric order groups are efficient because of their smaller group size. The composite order groups perform significantly worse by a factor of 6.

Table 1. Encryption Time of traitor tracing schemes

Users	Boneh et al.	Symmetric PLBE	Asymmetric PLBE	Skewed Asymm.
25	1.977s	0.749s	0.494s	0.3611s
100	3.971s	1.503s	1s	0.694s
225	6.069s	2.183s	1.512s	1.081s
400	8.2s	2.922s	3.187s	1.424s
1225	13.898s	5.1885s	3.495s	2.523s
2500	21.046s	7.227s	5.104s	3.583s
5625	29.681s	10.797s	8.069s	6.056s
10000	40.189s	14.552s	10.099s	8.074s
22500	-	21.769s	16.028s	10.577s

Fig. 3. Ciphertext Size (in bytes) of traitor tracing schemes

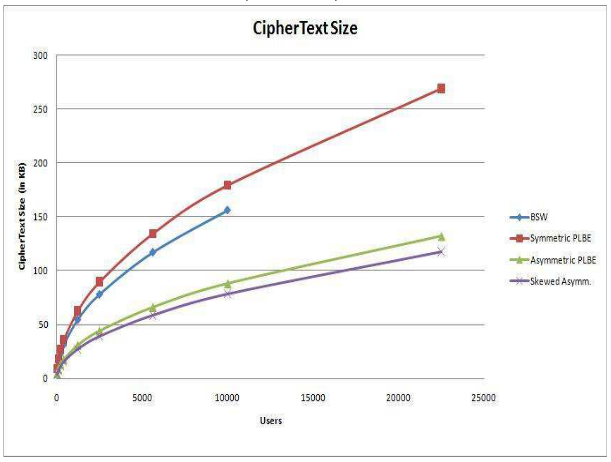


Fig. 4. Decryption time of traitor tracing schemes

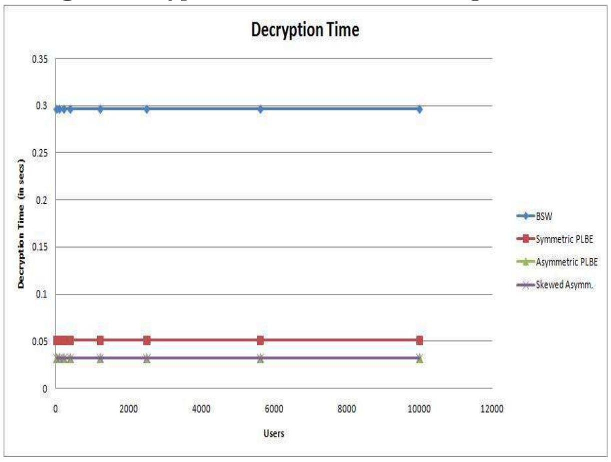


Table 2. CipherText Size (in bytes) of traitor tracing schemes

Users	Boneh et.al	Symmetric PLBE	Asymmetric PLBE	Skewed Asymm.
25	7800	8960	4400	3840
100	15600	17920	8800	7680
225	23400	26880	13200	11840
400	31200	35840	17600	15680
1225	54600	62720	30800	27360
2500	78000	89600	44000	39200
5625	117000	134400	66000	58720
10000	156000	179200	88000	78400
22500	-	268800	132000	117440

7.2 Ciphertext Size

The ciphertext size (Table 2, Figure 3) is dependent on the representation size of the elliptic curve and the number of group elements used. Our construction, although using a larger number of group elements, has smaller total ciphertext size (in the asymmetric case) because the group sizes are significantly smaller. The asymmetric groups, by their nature allows us to optimize ciphertext size by increasing the number of rows and decreasing the columns in (Fig. 1). We call this the Skewed Asymmetric group version. Note that by design, most of the group elements in the ciphertext are placed in the smaller group $\mathbb{G}_1$. Skewing has no effect on security proofs and allows us to optimize on ciphertext size.

Calculations show that using $25 \times 16 (= 400)$ rectangle for generating ciphertexts produces only 15680 bytes which gives us a 50% improvement compared to the scheme of [8]

Table 3. Encryption Time and CipherText size (in bytes) for Trace & Revoke in prime order groups

Users	Symm. Enc. Time	Asymm. Enc. Time	Symm. Ciphertext Size	Asymm. Ciphertext Size
25	0.611s	0.540s	9600B	4600B
100	1.179s	1.027s	19200B	9200B
225	1.695s	1.550s	28800B	13800B
400	2.213s	2.059s	38400B	18400B
1225	3.765s	3.594s	67200B	32200B
2500	5.272s	5.248s	96000B	46000B
5625	8.104s	7.759s	144000B	69000B

We provide and implement efficient broadcast, trace & revoke system. Table 3 provides encryption times and ciphertext size (in bytes) for up to 5625 users. The security guaranteed on the elliptic curves used are 1024 bit discrete log security.

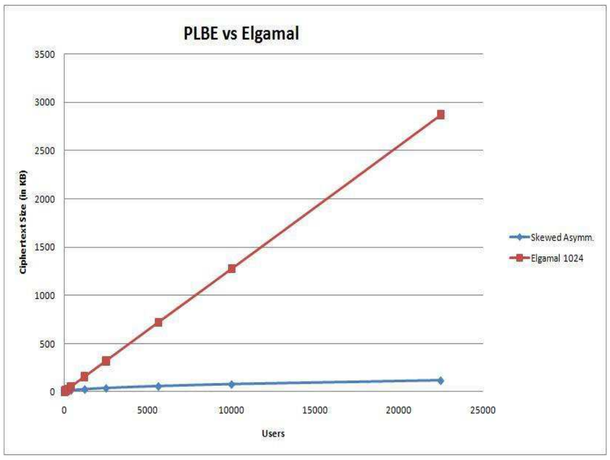
7.3 Decryption Time

The decryption time for the various scenarios (Figure 4) above are relatively constant and independent of the number of users for each scheme. This is because decryption time is dominated by the cost of pairing operations on the elliptic curves. The composite order schemes decrypt at $0.296s$ per ciphertext and the primer order symmetric and asymmetric groups decrypt at $0.051s$ and $0.032s$ respectively. Thus we see the prime order groups are relative similar w.r.t decryption times and are *10 times* faster due to faster pairing operations in these groups.

7.4 Comparison with the ElGamal Encryption

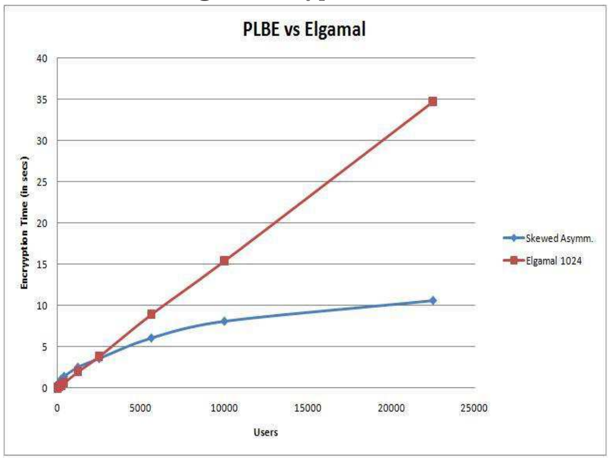
We compare the efficiency of our scheme with an implementation of a naïve (but optimized) ElGamal based traitor tracing scheme. The advantage of using an ElGamal based scheme is that the group that it works

Fig. 5. Ciphertext Size



on could support very efficient arithmentic operations (we choose the multiplicative group Z_p^* for a 1024 bit prime p) making encryption very fast. The disadvantage is that for N users ElGamal based systems use $O(N)$ steps whereas our scheme uses $O(\sqrt{N})$ steps. We observe that the ElGamal implementation has a huge ciphertext size overload compared to our scheme (Figure 5). We also observe that asymptotic improvements in the encryption time begin to show up for as few as 2500 users (Figure 6).

Fig. 6. Encryption Time



8 Conclusion

Boneh et al. [8,9] provide traitor tracing and trace & revoke systems using composite order bilinear groups. These groups have large exponentiation and pairing times making them impractical. We provide the first implementation of a traitor tracing and trace & revoke systems, using symmetric and asymmetric prime order bilinear groups. Our implementation and comparisons with [8] show that we achieve about 10 times faster decryption, 6 times faster encryption and 50% reduction in ciphertext size. The ideas presented in this work are general and can be applied to convert other composite order cryptosystems to efficient prime order based cryptosystems.

References

1. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of CRYPTO .04, LNCS series*, pages 41–55. Springer-Verlag, 2004.
2. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
3. D. Boneh and M. 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. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275, 2005.
5. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Second Theory of Cryptography Conference, TCC*, volume 3378 of *LNCS*, pages 325–341, 2005.
6. D. Boneh, B. Lynn, and H. 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.
7. D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In *ACM Conference on Computer and Communications Security*, pages 501–510, 2008.
8. D. Boneh, A. Sahai, and B. 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.
9. D. Boneh and B. 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.
10. H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.
11. B. Chor, A. Fiat, and M. 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.
12. Y. Dodis and N. Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography*, pages 100–115, 2003.
13. D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Preprint*, 2009.
14. S. Garg, A. Sahai and B. Waters. Efficient Fully Collusion-Resilient Traitor Tracing Scheme. Cryptology ePrint Archive, Report 2009/532, 2009.
15. C. Gentry and B. Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, pages 171–188, 2009.
16. A. Kiayias and M. Yung. Breaking and repairing asymmetric public-key traitor tracing. In *Digital Rights Management Workshop*, pages 32–50, 2002.
17. A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT*, pages 450–465, 2002.
18. K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, pages 145–157, 1998.
19. B. Lynn. The pairing-based cryptography library.
20. S. Mitsunari, R. Sakai, and M. Kasahara. A new traitor tracing. In *IEICE Trans. Fundamentals*, pages E85–A(2):481484, 2002.
21. A. Miyaji, M. Nakabayashi, and S. Takano. Characterization of elliptic curve traces under fr-reduction. In *ICISC*, pages 90–108, 2000.
22. M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, pages 1–20, 2000.
23. B. Pfitzmann. Trials of traced traitors. In *Information Hiding*, pages 49–64, 1996.
24. B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *ACM Conference on Computer and Communications Security*, pages 151–160, 1997.
25. 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.
26. Y. Watanabe, G. Hanaoka, and H. Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In *CT-RSA*, pages 392–407, 2001.
27. B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.