Fully Collusion Resistant Traitor Tracing With Short Ciphertexts and Private Keys

Abstract

We construct the first fully collusion resistant tracing traitors system with sublinear size ciphertexts and constant size private keys. More precisely, let N be the total number of users. Our system generates ciphertexts of size $O(\sqrt{N})$ and private keys of size $O(1)$. We build our system by first building a simpler primitive called *private linear broadcast encryption* (PLBE). We then show that any PLBE gives a tracing traitors system with the same parameters. Our system uses bilinear maps in groups of composite order.

1 Introduction

Tracing traitors systems, introduced by Chor, Fiat, and Naor [CFN94], help content distributors identify pirates. Consider a content distributor who broadcasts encrypted content to N legitimate recipients. Recipient i has secret key K_i that it uses to decrypt the broadcast. As a concrete example, imagine an encrypted satellite radio broadcast that should only be played on certified radio receivers. The broadcast is encrypted using a broadcaster key BK. Any certified player can decrypt using its embedded secret key K_i. Certified players, of course, can enforce digital rights restrictions such as "do not copy" or "play once."

The risk for the distributor is that a pirate will hack a certified player and extract its secret key. The pirate could then build a pirate decoder that will extract the cleartext content and ignore any relevant digital rights restrictions. Even worse, the pirate could make its pirate decoder widely available so that anyone can extract the cleartext content for themselves. DeCSS, for example, is a widely distributed program for pirating encrypted DVD content.

This is where tracing traitors systems come in — when the pirate decoder is found, the distributor (or the police) can run a *tracing* algorithm that interacts with the pirate player and outputs the index i of at least one of the keys K_i that the pirate used to create the pirate decoder. The distributor can then try to take legal action against the owner of this K_i. Even better, the distributor can revoke K_i so that the pirate decoder will not decode future content.

We give a precise description of tracing traitors systems in the next section. For now we give some intuition that will help explain our results. A Tracing traitors system consists of four algorithms *Setup*, *Encrypt*, *Decrypt*, and *Trace*. The setup algorithm generates the broadcaster's key BK, a tracing key TK, and N recipient keys $K_1, \ldots, K_N$. The encrypt algorithm encrypts the content using BK and the decrypt algorithm decrypts using one of the K_i. The tracing algorithm is the most interesting — it is an oracle algorithm that takes TK as input and interacts with a pirate decoder. It outputs the index $i \in \{1, \ldots, N\}$ of a key K_i that was used to create the pirate decoder. Tracing traitors systems can have a number of properties informally defined below (we give precise definitions in the next section):

- **Public vs. Secret BK:** Some tracing traitors systems require that the broadcaster's key BK remain secret; Other systems [KD98, BF99, NP01, KY02b, DF03, MSK02, TSNZ03, CPP05] support a public BK so that anyone can create broadcast data. Combinatorial systems such as [CFN94, NP98, SW98a, SW98b, FT99, GSY99, CFNP00, SNW00, BPS00, SSW00, SSW01, NNL01] are typically designed for the secret BK settings, but can be made public-key by replacing the underlying ciphers by public key systems.

- **Public vs. Secret TK:** Many tracing traitors systems (including ours) assume that the tracer is a trusted party and require that the tracer's key TK be kept secret. Some exceptions are [Pfi96, PW97, WHI01, KY02a, CPP05].

- **Collusion resistance:** a tracing traitors system is said to be t-collusion resistant if tracing will work as long as the pirate has fewer than t user keys at his disposal. If $t = N$ the system is said to be *fully collusion resistant*.

- **Black box tracing:** the tracing algorithm must do its job by treating the pirate decoder as a black-box device. In particular, the tracing algorithm should not look into the inner-workings of the decoder. The pirate is free to build the decoder however he wants — it could obfuscate the code, use tamper resistant hardware, randomize the keys, etc. It is therefore safest to treat the pirate decoder as a black box: the tracer gives encrypted content to the decoder and the decoder answers valid or invalid (i.e. it either plays the content or does not). The tracer learns nothing else.

- **Stateful vs. Stateless decoders:** a stateless decoder is one that does not keep state between decryptions. For instance, decoders published as software (such as DeCSS) cannot keep any state. However, pirate decoders embedded in tamper resistant hardware (e.g. a pirate cable box) can keep state between successive decryptions. When the decoder detects that it is being traced it could shutdown and refuse to decrypt further inputs (a software decoder cannot do that). Kiayias and Yung [KY01] show how to convert any tracing system for stateless decoders into a tracing system for stateful decoders by embedding robust watermarks in the content. Consequently, most tracing systems in the literature (as do we) focus on the stateless settings and ignore the stateful case.

Our results. In this paper we only consider fully collusion resistant tracing traitors systems. When setting $t = N$, existing t-collusion resistant systems require ciphertext size linear in the number of users N, which is no better than the trivial tracing traitors system. We construct the first practical fully collusion resistant tracing traitors system that has sub-linear size ciphertext. Our system has the following characteristics:

$$\text{ciphertext-length} = O(\sqrt{N}) \qquad \text{and} \qquad \text{private-key-length} = O(1)$$

Furthermore, decryption time is constant (i.e. depends on the security parameter, but not on N). Other properties of this system include: (1) the broadcaster's key BK is public, but the tracer's key TK must be kept secret, (2) the system is black-box traceable, and (3) is designed for stateless pirate decoders. The system uses bilinear groups of composite order introduced in [BGN05].

We prove security of our tracing algorithm using a standard tracing technique previously used in [BF99, NNL01, KY01]. To formalize this technique, we introduce a new primitive called *Private*

Linear Broadcast Encryption, or PLBE for short. For any $i \in \{0, \ldots, n\}$ a PLBE enables the sender to create broadcast ciphertexts that can only be decrypted properly under keys $K_i, \ldots, K_N$. A broadcast to everyone, for example, is encrypted using $i = 1$. The system should be private meaning that a ciphertext for users $\{i, \ldots, N\}$ should reveal no non-trivial information about i. We show that any secure PLBE gives a secure (black-box) tracing traitors system. We then construct a secure PLBE with the properties above. Future work on tracing traitors can adopt the same technique and focus on constructing PLBEs which is conceptually much simpler than tracing traitors.

Related work. Tracing traitors systems generally fall into two categories: combinatorial, as in [CFN94], and algebraic, as in [KD98, BF99]. The broadcaster's key BK in combinatorial systems can be either secret or public. Algebraic tracing traitors use public-key techniques and are usually more efficient than the public-key instantiations of combinatorial schemes. Some systems (including ours) only provide tracing capabilities. Other systems [NP01, NNL01, HS02, GST04, DF03] combine tracing with broadcast encryption to obtain trace-and-revoke features — after tracing, the distributor can revoke the pirate's keys without affecting any other legitimate decoder.

Kiayias and Yung [KY02b] describe a black-box tracing system that achieves constant rate for long messages, where rate is measured as the ratio of ciphertext length to plaintext length. For full collusion resistance, however, the ciphertext size is linear in the number of users n. For comparison, our new system generates ciphertexts of size $O(\sqrt{N})$ and achieves constant rate (rate = 1) for long messages by using hybrid encryption (i.e. encrypting a short message-key using the tracing-traitors system and encrypting the long data by using a symmetric cipher with the message-key).

Finally, we note that binary fingerprinting codes [BS98, Tar03] are closely related to tracing traitors (binary refers to the fact that the code is defined over a binary alphabet). In fact, it is known [BN02] that any binary fingerprinting code gives rise to a fully collusion-resistant tracing traitors system with *constant* size ciphertexts. The private key size, unfortunately, is quite large. Using [BS98] the private key size is $\tilde{O}(N^3)$ and using [Tar03] it is $\tilde{O}(N^2)$.

2 Traitor Tracing and Private Linear Broadcast Encryption

In Appendix A we formally define tracing traitors systems along with the games used to define security. However, instead of directly building a tracing traitors system we build a simpler primitive called *Private Linear Broadcast Encryption* (PLBE). We define secure PLBE's below and then show in Appendix B that a secure PLBE directly gives a tracing traitors system. This conversion makes precise a standard tracing technique used in [BF99, NNL01, KY01]. Then in the next section we build a PLBE.

2.1 Description of Private Linear Broadcast Encryption

A PLBE is composed of the following four algorithms:

Setup$_{\text{LBE}}(N, 1^\kappa)$ The setup algorithm takes as input N, the number of users in the system, and the security parameter κ. It outputs public parameters P, a secret key SK, a public key PK, and N keys $K_1, \ldots, K_N$, where K_u is given to user u.

Encrypt$_{\text{LBE}}(\mathbf{SK}, m, u)$ The encryption algorithm will take as input the secret key SK, a message m, and an encryption index $1 \leq u \leq N+1$. We assume that m comes from some samplable set M (defined by P). The algorithm outputs a ciphertext c so that only users with index greater than or equal to u will be able to decrypt c.

Broadcast$_{\text{LBE}}(\mathbf{PK}, m)$ A public key encryption algorithm that takes a public key and message as input and outputs a ciphertext c. We require that $\mathit{Broadcast}_{\text{LBE}}(\text{PK}, m)$ is identically distributed with $\mathit{Encrypt}_{\text{LBE}}(\text{SK}, m, 1)$ so that any user can decrypt.

Decrypt$_{\text{LBE}}(u', K_{u'}, c)$ User u' attempts to decrypt ciphertext c using key $K_{u'}$. The algorithm outputs a plaintext m or fail. The user can successfully decrypt if his index u' is greater than or equal to u, the index used to create the ciphertext.

Security. We define security of PLBE using two games. We first define the **Index Hiding Game**, which captures the intuition that a broadcast ciphertext to users $u, \ldots, N$ reveals no non-trivial information about u. More precisely, an adversary is unable to distinguish between an encryption to index u and one to $u+1$ without the key K_u. We will consider the game for a fixed number of users, N.

- **Init** The adversary gives the challenger an index u in $\{1, \ldots, N\}$. The challenger runs the setup algorithm and gives the adversary P and PK along with all keys K_i such that $i \neq u$.
- **Challenge** The adversary gives the challenger a message m. The challenger flips a coin $\beta \in \{0, 1\}$ and computes $c = \mathit{Encrypt}_{\text{LBE}}(\text{SK}, m, u + \beta)$. The challenger returns c to the adversary.
- **Guess** The adversary returns a guess β' of β.

We define the adversary's advantage as $\mathsf{Adv}_{IH} = |\Pr[\beta' = \beta] - 1/2|$. We require that the adversary's advantage is negligible in κ.

Next, we describe the **Message Hiding Game** which is a semantic security game for index $u = N + 1$ (i.e. for ciphertexts that cannot be decrypted by any of the participants). The game proceeds in three phases:

- **Init** The challenger runs the setup algorithm and gives the adversary P and PK along with all keys $K_1, \ldots K_N$.
- **Challenge** The adversary gives the challenger two equal length messages m_0, m_1. The challenger flips a coin $\beta \in \{0, 1\}$ and computes $c = \mathit{Encrypt}_{\text{LBE}}(\text{SK}, m_\beta, N + 1)$. The challenger gives c to the adversary.
- **Guess** The adversary returns a guess β' of β.

We define the adversary's advantage as $\mathsf{Adv}_{MH} = |\Pr[\beta' = \beta] - 1/2|$. We require that the adversary's advantage is negligible in κ.

Definition 2.1. *A* PLBE *system is secure if both* Adv_{IH} *and* Adv_{MH} *are negligible functions of* κ.

3 Background and complexity assumptions

3.1 Bilinear maps

We review some general notions about bilinear maps and groups, with an emphasis on groups of *composite order* which will be used in most of our constructions. We follow [BGN05] in which composite order bilinear groups were first introduced.

Consider two finite cyclic groups $\mathbb{G}$ and $\mathbb{G}_T$ of same order $n = pq$, where p and q are distinct primes, and in which the respective group operation is efficiently computable and denoted multiplicatively. Assume the existence of an efficiently computable function $e : \mathbb{G} \times \mathbb{G} \to G_T$, with the following properties:

- (Bilinear) $\forall u, v \in G$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$, where the product in the exponent is defined modulo n;
- (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order n in G_T. In other words, $e(g, g)$ is a generator of G_T, whereas g generates G.

We will use the notation $\mathbb{G}_p, \mathbb{G}_q$ to denote the respective subgroups of order p and order q of $\mathbb{G}$.

We now review three assumptions we will use for proving our security. The first two assumptions are in prime order subgroups and the last two are over a composite group $\mathbb{G}$.

3.2 Decision 3-party Diffie-Hellman Assumption

The decision 3-party Diffie-Hellman problem is stated as follows. Given a group $\mathbb{G}_p$ of prime order p and random elements $g_p, A = g_p^a, B = g_p^b, C = g_p^c$ of $\mathbb{G}$ distinguish between $T = g_p^{abc}$ and $T = g_p^z$, where z is random in $\mathbb{Z}_p$.

We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the problem if

$$\left| \Pr[\mathcal{A}(g_p, g_p^a, g_p^b, g_p^c, g_p^{abc}) = 1] - \Pr[\mathcal{A}(g_p, g_p^a, g_p^b, g_p^c, g_p^z) = 1] \right| \geq \epsilon$$

The (t, ϵ)-decision 3-party Diffie-Hellman assumption (D3DH) is that no t-time adversary has advantage more than ϵ. Note that the decision 3-party Diffie-Hellman assumption implies the decision Bilinear Diffie-Hellman assumption. It also implies the standard linear assumption defined in [BBS04].

3.3 Subgroup Decision Problem

The Subgroup Decision (SD) problem is stated as follows. Given a group $\mathbb{G}$ of composite order $n = pq$, where p, q are distinct (unknown) primes, and generators $g_p \in \mathbb{G}_p$ and $g \in \mathbb{G}$, distinguish between whether an element T is a random member of the subgroup $\mathbb{G}_p$ or a random element of the full group $\mathbb{G}$. That is distinguish whether T is a random element of $\mathbb{G}_p$ or $\mathbb{G}$.

We say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the Subgroup Decision Problem if

$$\left| \Pr[\mathcal{A}(n, g_p, g, T) = 1 \ : \ T \stackrel{\mathrm{R}}{\leftarrow} \mathbb{G}_p] - \Pr[\mathcal{A}(n, g_p, g, T) = 1 \ : \ T \stackrel{\mathrm{R}}{\leftarrow} \mathbb{G}] \right| \geq \epsilon.$$

The (t, ϵ)-subgroup decision assumption is that no t-time adversary has advantage more than ϵ.

3.4 Bilinear Subgroup Decision Problem

The Bilinear Subgroup Decision (BSD) problem is stated as follows. Given a group $\mathbb{G}$ of composite order $n = pq$, where p, q are distinct (unknown) primes, and generators $g_p \in \mathbb{G}_p$ and $g_q \in \mathbb{G}_q$, distinguish a random order p element in the group $\mathbb{G}_T$ from a uniform element in the group $\mathbb{G}_T$. More precisely, we say that an algorithm $\mathcal{A}$ has advantage ϵ in solving the problem if

$$\left| \Pr[\mathcal{A}(n, g, g_p, g_q,\ e(T, g)) = 1 \ :\ T \xleftarrow{\mathrm{R}} \mathbb{G}_p] - \Pr[\mathcal{A}(n, g, g_p, g_q,\ e(T, g)) = 1 \ :\ T \xleftarrow{\mathrm{R}} \mathbb{G}] \right| \geq \epsilon.$$

The (t, ϵ)-bilinear subgroup decision assumption is that no t-time adversary has advantage more than ϵ.

4 A $\sqrt{N}$ size Private Linear Broadcast Encryption System

In this section we show how to construct a Private Linear Broadcast Encryption (PLBE) system with $O(\sqrt{N})$ size ciphertext. We can then apply the results of the previous section and use this to build a traitor tracing scheme with $O(\sqrt{N})$ size ciphertexts.

Before we describe our construction we give some intuition as to why constructing PLBE systems with sublinear ciphertext size is difficult and describe the framework for which we will construct our PLBE system.

PLBE with Sublinear Ciphertext Size The primary difficulty in constructing a PLBE system is to provide the Index Hiding property. Using linear size ciphertext this is easy: each user has a designated portion of the ciphertext assigned to them, which is used to encrypt the message (or session key) to just that user. If an encryptor replaces the ciphertext component of user u with a random encryption, only user u can tell the difference. All other users use different portion of the ciphertext for decrypting and changing u's component has no effect on them.

To construct a PLBE system with sublinear size ciphertexts we must use a fundamentally different approach than the one above. Since, the ciphertexts are sublinear we cannot let every user have a component of the ciphertext that is dedicated for them and is not needed by other users. Intuitively, ciphertext components must be "shared" amongst users. We therefore cannot use the simple strategy of completely randomizing a portion of the ciphertext to prevent a particular user u from decrypting, since this will inherently effect the ability of other users to decrypt.

Our Framework We now give a framework for our PLBE system. We assume that the number of users, N in the system equals m^2 for some m. If the number of real users is not a square we can add "dummy" users to pad out to the next square. We arrange the users in an $m \times m$ matrix. Each user is assigned and identified by an unique tuple (x, y) where $1 \leq x, y \leq m$.

Since we will be constructing a Private Linear Broadcast Encryption system we must have a linear ordering of the users that we can traverse. The first user in the system will be the user at matrix position $(1, 1)$ and from there we will order the users by traversing one row at a time. More precisely, the user at matrix position (x, y) will have the index $u = (x - 1)m + y$ in our ordering.

We observe that we can now refer to our Private Linear Broadcast Encryption scheme in terms of positions on the matrix. An encryption to position (i, j) means that a user at position (x, y) will be able to decrypt the message if either $x > i$ or both $x = i$ and $y \geq j$. With this notation, the Index Hiding game property states that:

- For $j < m$ it is difficult to distinguish between an encryption of a message to (i, j) from $(i, j + 1)$ without the key of user $(x = i, y = j)$.
- For $j = m$ it is difficult to distinguish an encryption of a message to position $(i, j = m)$ to that of one to $(i + 1, j = 1)$ without the key of user $(i, j = m)$.

The use of pairwise notation (i, j) is purely a notational convenience for describing our system.

4.1 Our Construction

Our construction makes use of bilinear maps of composite order n, where $n = pq$ and p and q are primes. In describing our scheme we will often use p or q in a subscript to denote if a group element is in the subgroup of order p or order q.

When the $Encrypt_{\text{LBE}}$ algorithm encrypts to an index (i, j) it creates ciphertext components for every column and every row. The keys of user (x, y) are structured in such a way that in order to decrypt he must pair the ciphertext components from row x, with the column ciphertext components from column y.

The encryption algorithm works by creating ciphertexts in the following way. Ciphertexts for columns greater than or equal to j are well formed in both subgroups. However, for a column that is less than j, the encryption algorithm will create a ciphertext which is well formed in the $\mathbb{G}_q$ subgroup, but random in the $\mathbb{G}_p$ subgroup. Ciphertexts for rows less than i are completely random. Therefore, any user whose row index is less than x will not be able to decrypt. The ciphertext for row i is composed of well-formed elements in the group $\mathbb{G}$. A user with row index i will be able to decrypt if his column index is greater than or equal to j. If it is less than j, the randomized part of the column ciphertext will scramble the result of pairing the row and column ciphertexts together. Finally, for rows greater than i the ciphertext components will be well formed elements in the $\mathbb{G}_q$ subgroup. A user with row index greater than i will be able to decrypt no matter what his column value is since the row ciphertext will be orthogonal to the randomized parts of any column ciphertexts.

The decryption algorithm for a user (x, y) will attempt to decrypt a ciphertext in the same manner no matter what the target index (i, j) is. Intuitively, the structure of the ciphertext will restrict decryption to only be successful for a user (x, y) if $x > i$ or $x = i$ and $y \geq j$. Additionally, since, the attempted decryption procedure is independent of (i, j) a user can only learn whether his decryption was successful or not and the system will be private.

We describe the four algorithms that compose our PLBE system:

Setup$_{\text{LBE}}(N = m^2, 1^\kappa)$ The setup algorithm takes as input the number of users N and a security parameter κ. It first generates an integer $n = pq$ where p,q are random primes (whose size is determined by the security parameter). The algorithm creates a bilinear group $\mathbb{G}$ of composite order n. It next creates random generators $g_p, h_p \in \mathbb{G}_p$ and $g_q, h_q \in \mathbb{G}_q$ and sets $g = g_p g_q, h = h_p h_q \in \mathbb{G}$. Next it chooses random exponents $r_1, \ldots, r_m, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$ and $\beta \in \mathbb{Z}_q$.

The public key PK includes the description of the group and the following elements:

$$g, h, E = g^\beta, E_1 = g_q^{\beta r_1}, \ldots, E_m = g_q^{\beta r_m}, F_1 = h_q^{\beta r_1}, \ldots, F_m = h_q^{\beta r_m},$$

$$G_1 = e(g_q, g_q)^{\beta \alpha_1}, \ldots, G_m = e(g_q, g_q)^{\beta \alpha_m}$$

$$H_1 = g^{c_1}, \ldots, H_m = g^{c_m}$$

The private key for user (x, y) is generated as $K_{x,y} = g^{\alpha_x} g^{r_x c_y}$. Finally, the authority's secret key SK includes factors p, q along with exponents used to generate the public key.

Encrypt$_{\text{LBE}}(\mathbf{SK}, M, (i, j))$ The *Encrypt*$_{\text{LBE}}$ algorithm is a secret key algorithm used by the tracing authority. The algorithm encrypts a message M to the subset of receivers that have row values greater than i or both row value equal to i and column values greater than or equal to j.

The encryption algorithm will take as input the secret key, a message $M \in \mathbb{G}_T$ and an index i, j. The encryption algorithm first chooses random $t \in \mathbb{Z}_n$, $w_1, \ldots, w_m$, $s_1, \ldots, s_m \in Z_n$, $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, and $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n^{(3)}$.

For each row x we create four ciphertext components $(R_x, \tilde{R}_x, A_x, B_x)$ as follows:

$$\begin{array}{llllll} \text{if } x > i\text{:} & R_x = g_q^{s_x r_x} & \tilde{R}_x = h_q^{s_x r_x} & A_x = g_q^{s_x t} & B_x = Me(g_q, g)^{\alpha_x s_x t} \\ \text{if } x = i\text{:} & R_x = g^{s_x r_x} & \tilde{R}_x = h^{s_x r_x} & A_x = g^{s_x t} & B_x = Me(g, g)^{\alpha_x s_x t} \\ \text{if } x < i\text{:} & R_x = g^{v_{x,1}} & \tilde{R}_x = h^{v_{x,1}} & A_x = g^{v_{x,2}} & B_x = e(g, g)^{v_{x,3}} \end{array}$$

For each column y the algorithm creates values $C_y, \tilde{C}_y$ as:

$$\begin{array}{lll} \text{if } y \geq j\text{:} & C_y = g^{c_y t} h^{w_y} & \tilde{C}_y = g^{w_y} \\ \text{if } y < j\text{:} & C_y = g^{c_y t} g_p^{z_y} h^{w_y} & \tilde{C}_y = g^{w_y} \end{array}$$

Note that the ciphertext contains $5\sqrt{N}$ elements in $\mathbb{G}$ and $\sqrt{N}$ elements of $\mathbb{G}_T$.

In the above description there are three classes of rows. A row $x > i$ will have all its elements in the $\mathbb{G}_q$ subgroup, while the "target" row i will have its components in the full group $\mathbb{G}$. A row $x < i$ will essentially have its group elements randomly chosen. A column $y \geq j$ will be well formed, which a column $y < j$ will be well formed in the $\mathbb{G}_q$ subgroup, but not in the $\mathbb{G}_p$ subgroup.

Broadcast$_{\text{LBE}}(\mathbf{PK}, M)$ The *Broadcast*$_{\text{LBE}}$ algorithm is used by an encryptor to encrypt a message such that all the recipients can receive it. This algorithm is used during normal (non-tracing) operation to distribute content to all the receivers. The *Broadcast*$_{\text{LBE}}$ algorithm should produce ciphertexts that are indistinguishable from *Encrypt*$_{\text{LBE}}$ algorithm to the index $(1, 1)$ for the same message.

The encryption algorithm first chooses random $t \in \mathbb{Z}_n$, $w_1, \ldots, w_m$, $s_1, \ldots, s_m \in \mathbb{Z}_n$,

For each row x the algorithm creates the four ciphertext components $(R_x, \tilde{R}_x, A_x, B_x)$ as follows:

$R_x = E_x^{s_x} \quad \tilde{R}_x = F_x^{s_x} \quad A_x = E^{s_x t} \quad B_x = MG_x^{s_x t}$

For each column j the algorithm creates $C_y, \tilde{C}_y$ as:

$C_y = H_y^t h^{w_y} \quad \tilde{C}_y = g^{w_y}$

Decrypt$_{\text{LBE}}((x, y), K_{x,y}, C)$ User (x, y) uses key $K_{x,y}$ to decrypt by computing:

$$B_x / \left(e(K_{x,y}, A_x) e(\tilde{R}_x, \tilde{C}_y) / e(R_x, C_y) \right).$$

We first observe that if the ciphertext was created from the *Encrypt*$_{\text{LBE}}(\text{SK}, M, (i, j))$ then if $x > i$ or $x = i$ and $y \geq j$ then the result is M. If the row value of the decryptor $x > i$ then the row components of the ciphertext will be in the $\mathbb{G}_q$ subgroup. When these row components are paired with the corresponding column components it will not matter if the columns are well formed in the

$\mathbb{G}_p$ subgroup since the row component will be orthogonal to that subgroup. (This follows from the fact that pairing an element of $\mathbb{G}_p$ with an element of $\mathbb{G}_q$ gives the identity element in the target group.) If the row value of the decryptor $x = i$ then the row components of the ciphertext are in the full group $\mathbb{G}_p$. Therefore, decryption will only work if $y > j$, since the column values C_y must be well formed in both subgroups. Additionally, it is easy to observe that if the ciphertext was created as $\mathit{Broadcast}_{\mathrm{LBE}}(\mathrm{PK}, M)$ all parties can decrypt and receive M.

4.2 Discussion

Roughly, the size of the ciphertext is $5\sqrt{N}$ elements in $\mathbb{G}$ and $\sqrt{N}$ elements of $\mathbb{G}_T$. In practice, a message will be encrypted with a symmetric key cipher under a key K and our system will be used to transmit the key K to each user. We note that we can actually save in ciphertext size by converting our encryption system into a Key-Extraction Mechanism (KEM). To do this we do not include the B_x values in the ciphertext, but instead user (x, y) can extract a key $K_x = e(K_{x,y}, A_x)e(\tilde{R}_x, \tilde{C}_y)/e(R_x, C_y)$. The extraction mechanism will actually derive $\sqrt{N}$ different keys $K_1, \ldots K_m$, so key K_x would still need to be used to encrypt K to for all users in row x. However, in practice this would be more space efficient than including $\sqrt{N}$ group elements of $\mathbb{G}_T$.

The $\mathit{Broadcast}_{\mathrm{LBE}}$ algorithm requires $6\sqrt{N}$ exponentiations. The decryption algorithm is surprisingly efficient and simple requiring only three pairing computations. Thus, decryption time is independent of the number of users in the system.

We constructed a (limited)[1] broadcast encryption system in which decryptors are oblivious as to which set of users the broadcast is targeted for. A set of colluding users will of course be able to learn some information about the target from their positions in the matrix and whether each one of them was able to decrypt. However, they should not learn anything more than what can naturally be inferred. The key to keeping the broadcast set private is that the decryption algorithm performs the same steps to attempt decryption no matter what the broadcast set is. In the next section we prove this intuition to be correct by showing that our scheme is secure in the Index Hiding game.

5 Security Proof

In this section we prove our Private Linear Broadcast Encryption system secure in the Index Hiding and the Message Hiding games. The Index Hiding proof is the most interesting and requires us to consider two cases. The first is when an adversary tries to distinguish between an encryption to (i, j) and an encryption to $(i, j+1)$ for $j < m$ and second for when an adversary tries to distinguish between an encryption (i, m) and one to $(i + 1, 1)$.

In the first case we show that the difficulty of this game can be reduced to the 3-party Diffie-Hellman assumption, while the second case is more complicated since the structure of the row ciphertexts are changed. We handle the second case by constructing a sequence of hybrid experiments. Due to space requirements we will defer several of the proofs of various lemmas and claims to the appendix.

Theorem 5.1. *Suppose that the (t, ϵ_{DBDHL})-decision 3-party Diffie-Hellman, (t, ϵ_{BSD})-Bilinear Subgroup Decision, and (t, ϵ_{SD})-Subgroup Decision assumptions hold. Then no $\tilde{t}$-time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with advantage greater than $2\epsilon_{D3DH} + \epsilon_{BSD} + \epsilon_{SD}$, where $\tilde{t} \approx t$.*

[1] A Private Linear Broadcast Encryption system is restricted in the sets of users it can encrypt to — it can only encrypt to sets $\{i, \ldots, N\}$ for any i.

We first consider the case where an adversary $\mathcal{A}$ attempts to distinguish between an encryption to $(i+j)$ and $(i, j+1)$ where $j < m$. This is the case when the distinguishing game does not cross rows. We prove the following lemma.

Lemma 5.2. *Suppose that the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman, assumption holds. Then no t-time adversary can distinguish between an encryption to (i, j) and $(i, j+1)$ in the Index Hiding game for $j < m$ with advantage $> \epsilon_{D3DH}$.*

We prove this lemma in Appendix C.1.

We now turn to the more difficult case of when the adversary $\mathcal{A}$ chooses to distinguish between an encryption to (i, m) and one to $(i+1, 1)$ for some $1 \leq i < m$. This case becomes more complicated because the form of ciphertext rows will change. In our proofs we will refer to the rows with ciphertexts in the $\mathbb{G}_q$ subgroup as "greater than" rows and the the row with well formed ciphertexts in $\mathbb{G}$ as a "target" row. Additionally, when we say we "encrypt to column j" this means that we create ciphertexts for which C_y is well formed in the $\mathbb{G}_p$ subgroup for all $y \geq j$. We state our lemma and then prove it.

Lemma 5.3. *Suppose that the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman, (t, ϵ_{BSD})-Bilinear Subgroup Decision, and (t, ϵ_{SD})-Subgroup Decision assumptions hold. Then no $\tilde{t}$-time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with advantage greater than $2\epsilon_{D3DH} + \epsilon_{BSD} + \epsilon_{SD}$, where $\tilde{t} \approxeq t$.*

We first define a sequence of hybrid experiments as follows:

- H_1: Encrypt to column m, row i is target row, i+1 is a "greater than" row.
- H_2: Encrypt to column $m+1$, row i is target row, i+1 is a "greater than" row.
- H_3: Encrypt to column $m+1$, row i is less than row, i+1 is a "greater than" row (no target row exists).
- H_4: Encrypt to column 1, row i is less than row, i+1 is "greater than" row (no target row exists).
- H_5: Encrypt to column 1, row i is less than row, i+1 is target row.

We prove our lemma by giving reductions for each consecutive pair of hybrid experiments.

Claim 5.4. *Suppose that the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman assumption holds. Then no t-time adversary can distinguish between experiments H_1 and H_2 with advantage greater than ϵ_{D3DH} .*

In both experiments we encrypt with row i as the target row and all C_y for $y < m$ random in the $\mathbb{G}_p$ subgroup. The experiment is whether an adversary can tell if the $\mathbb{G}_p$ component of C_m is well-formed without key $K_{i,m}$. This experiment is exactly the same as the one we proved above and thus we apply the result of Lemma 5.2. □

Claim 5.5. *Suppose that the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman and the (t, ϵ_{BSD})-Bilinear Subgroup Decision assumptions hold. Then no t-time adversary can distinguish between experiments H_2 and H_3 with advantage greater than $2\epsilon_{D3DH} + \epsilon_{BSD}$.*

We prove this claim in Appendix C.2.

Claim 5.6. *Suppose that the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman assumption holds. Then no t-time adversary can distinguish between experiments H_3 and H_4 with advantage greater than $m \cdot \epsilon_{D3DH}$.*

We prove this claim in Appendix C.3.

Claim 5.7. *Suppose that the (t, ϵ_{SD})-Subgroup Decision assumption holds. Then no t-time adversary can distinguish between experiments H_4 and H_5 with advantage greater than ϵ_{SD} .*

We prove this claim in Appendix C.4.

Lemma 5.3 follows by summing the maximum adversarial advantages across the hybrid experiments and Theorem 5.1 follows by observing that the bound of Lemma 5.2 is included in Lemma 5.3. □

We now state our theorem about distingishing between $Broadcast_{\text{LBE}}(\text{PK}, M)$ and $Encrypt_{\text{LBE}}(\text{SK}, M, (1, 1))$.

Theorem 5.8. *Suppose the (t, ϵ_{SD}) Subgroup Decision assumption holds. Then for all messages M no t-time adversary can distinguish between a ciphertext created as $Broadcast_{\text{LBE}}(PK, M)$ and one created as $Encrypt_{\text{LBE}}(SK, M, (1, 1))$ with advantage greater than ϵ_{SD}.*

This theorem follows by simplying applying the same techniques as in our proof of Claim 5.7, so we omit the details. □

Finally, we state the theorem from our Message Hiding game.

Theorem 5.9. *All adversaries have advantage 0 in playing the Message Hiding game.*

The message hiding theorem is concerned with the adversaries advantage in winning the game when we encrypt to $(m + 1, 1)$. However, this means that all rows will be completely random and independent of the messge, thus an adversary has 0 advantage. Essentially, the inability of the adversary to learn the message when he does not have any of the right keys is actually captured in our Index Hiding experiments. This final theorem shows that at the end the adversary learns now information about the ciphertext. □

6 Discussion

Our traitor tracing system has a number of possible interesting extensions for future work. In this section we discuss a few of these.

t-collusion resistant variant One of the most important properties of our traitor tracing system is that it remains secure no matter how many users collude. However, there may be instances where we would like to relax our requirements to only be resistant to t colluders in return for a gain in efficiency. We leave for future work the problem of seeing if any such tradeoffs exist that make use of our techniques.

Public Traceability In our current system the tracing key, TK, is kept secret and only the authority is able to trace pirate boxes. In practice, it might be useful to have a system where the tracing key is public. For example, in a large content distribution system the capturing and tracing of pirate boxes or software will likely be done by different several agents each of which will need

the tracing key. We would like our system to remain secure even if one of these agent's and their tracing key is compromised.

In our $\sqrt{N}$ PLBE system the tracing algorithm would be public if a user was able to encrypt a message to an arbitrary set of indices (i, j). Then the user could simply run the tracing algorithm in the same way as the authority. In order to this we would need to give the user the capability to form C_y column ciphertext components that were well formed in its $\mathbb{G}_q$ subgroup, but not in the $\mathbb{G}_p$ subgroup. If we simply include an element of $\mathbb{G}_p$ in the public key our scheme will become insecure as an attacker could use this to determine which row index i a broadcast was intended for. Achieving public traceability would seem to require a more complex technique and possibly the use of a stronger assumption.

Stateful Receivers Like most other tracing traitor solutions our solution solves the tracing traitors problem in the stateless model, where the tracer is allowed to reset the pirate algorithm after each tracing query. However, there are some applications where we would like to consider a stronger model where a pirate box can retain state between each broadcast. In practice, a hardware pirate box might keep state and shut down if it detects that it is being traced.

Kiayias and Yung [KY01] showed a method which can handle stateful receivers if it were possible to embed watermarks in the distributed content and for a tracer to be able to observe these watermarks when interacting with a pirate algorithm. Their method essentially works as follows. During *non-tracing* operation the broadcaster encrypts two copies of digital content, each of which has a different watermark embedded in, to a random (and hidden) index u. The encryption is such that all users with index less than u can decrypt the first ciphertext and all users with index greater than u can decrypt the second ciphertext. The decryption algorithm simply tries to decrypt both ciphertexts and uses whichever one results in a well-formed plaintext. The tracing algorithm will create ciphertexts in an *identical* manner to the regular encryption algorithm. The tracer will simply observe which watermarks are embedded in every probing ciphertext and use this information to identify the traitor. Since, the regular broadcast and tracing algorithms are identical a pirate box is unable to leverage its ability to maintain state.

In our current construction, our PLBE scheme is only secure if the pirate constructing the pirate algorithm has not seen encryptions to arbitrary indices. However, if we were able to find a new PBLE algorithm that was secure under chosen-plaintext queries to arbitrary indicies then we could implement the techniques of Kiayias and Yung. We would simply set up two PLBE systems in which the users were given the opposite indices in each system. The user with index u in the first system has index $N + 1 - u$ in the second system.

7 Conclusions and Open Problems

We constructed the first fully collusion resistant traitor tracing system with sublinear size ciphertexts and constant size private keys. In particular, our system has ciphertexts of size $O(\sqrt{N})$ where N is the number of users in the system and the time for decryption is independent of N. We achieve our traitor tracing system by first introducing a simpler primitive we call private linear broadcast encryption (PLBE) that we show can give a traitor tracing system. Then, we built an efficient PLBE system by making novel use of bilinear groups of composite order.

One interesting open problem is to create a version of our traitor system that allows for public traceability. This would allow both for the tracer to be untrusted and could be used to give a

solution that is secure against stateful receivers. Additionally, it is an open problem to see if one can get smaller than $\sqrt{N}$ size ciphertexts with small private keys.

References

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Proceedings of Crypto 2004*, LNCS. Springer-Verlag, August 2004. To appear.

[BF99] 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.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In Joe Kilian, editor, *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342. Springer, 2005.

[BN02] Dan Boneh and Moni Naor. Tracing traitors with constant size ciphertext using binary fingerprinting codes. Unpublished, 2002.

[BPS00] O. Berkman, M. Parnas, and J. Sgall. Efficient dynamic traitor tracing. In *Proceedings of SODA '00*, 2000.

[BS98] Dan Boneh and James Shaw. Collusion secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998. Extended abstract in Crypto '95.

[CFN94] 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.

[CFNP00] Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

[CPP05] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT*, pages 542–558, 2005.

[DF03] Yevgeniy Dodis and Nelly Fazio. Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In *Public Key Cryptography - PKC 2003*, volume 2567 of *LNCS*, pages 100–115, 2003.

[FT99] Amos Fiat and T. Tassa. Dynamic traitor tracing. In *Proceedings of Crypto '99*, volume 1666 of *LNCS*, pages 354–371, 1999.

[GST04] M. T. Goodrich, J. Z. Sun, , and R. Tamassia. Efficient tree-based revocation in groups of low-state devices. In *Proceedings of Crypto '04*, volume 2204 of *LNCS*, 2004.

[GSY99] Eli Gafni, Jessica Staddon, and Yiqun Lisa Yin. Efficient methods for integrating traceability and broadcast encryption. In *CRYPTO '99: Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pages 372–387, London, UK, 1999. Springer-Verlag.

[HS02] D. Halevy and A. Shamir. The lsd broadcast encryption scheme. In *Proceedings of Crypto '02*, volume 2442 of *LNCS*, pages 47–60, 2002.

[KD98] K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Proceedings of Eurocrypt '98*, pages 145–157, 1998.

[KY01] Aggelos Kiayias and Moti Yung. On crafty pirates and foxy tracers. In *ACM Workshop in Digital Rights Management – DRM 2001*, pages 22–39, London, UK, 2001. Springer-Verlag.

[KY02a] Aggelos Kiayias and Moti Yung. Breaking and repairing asymmetric public-key traitor tracing. In Joan Feigenbaum, editor, *ACM Workshop in Digital Rights Management – DRM 2002*, volume 2696 of *Lecture Notes in Computer Science*, pages pp. 32–50. Springer, 2002.

[KY02b] Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 450–465, London, UK, 2002. Springer-Verlag.

[MSK02] Shigeo Mitsunari, Ryuichi Sakai, and Masao Kasahara. A new traitor tracing. *IEICE Trans. Fundamentals*, E85-A(2):481–484, 2002.

[NNL01] 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.

[NP98] Moni Naor and Benny Pinkas. Threshold traitor tracing. In *CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology*, pages 502–517, London, UK, 1998. Springer-Verlag.

[NP01] 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.

[Pfi96] B. Pfitzmann. Trials of traced traitors. In *Proceedings of Information Hiding Workshop*, pages 49–64, 1996.

[PW97] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *Proceedings of the ACM Conference on Computer and Communication Security*, pages 151–160, 1997.

[SNW00] Reihaneh Safavi-Naini and Yejing Wang. Sequential traitor tracing. In *Proceedings of Crypto '00*, volume 1880 of *LNCS*, pages 316–332, 2000.

[SSW00] Jessica N. Staddon, Douglas R. Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. Cryptology ePrint 2000/004, 2000.

[SSW01] Alice Silverberg, Jessica Staddon, and Judy L. Walker. Efficient traitor tracing algorithms using list decoding. In *Proceedings of ASIACRYPT '01*, volume 2248 of *LNCS*, pages 175–192, 2001.

[SW98a] D. Stinson and R. Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM Journal on Discrete Math*, 11(1):41–53, 1998.

[SW98b] D. Stinson and R. Wei. Key preassigned traceability schemes for broadcast encryption. In *Proceedings of SAC '98*, volume 1556 of *LNCS*, 1998.

[Tar03] Gabor Tardos. Optimal probabilistic fingerprint codes. In *Proceedings of STOC '03*, pages 116–125, 2003.

[TSNZ03] V. To, R. Safavi-Naini, and F. Zhang. New traitor tracing schemes using bilinear map. In *Proceedings of 2003 DRM Workshop*, 2003.

[WHI01] Yuji Watanabe, Goichiro Hanaoka, and Hideki Imai. Efficient asymmetric public-key traitor tracing without trusted agents. In *Proceedings CT-RSA '01*, volume 2020 of *LNCS*, pages 392–407, 2001.

A Definition of Tracing Traitors

A Traitor-Tracing Scheme consists of the following four algorithms:

Setup$(N,\ 1^\kappa)$**:** The setup algorithm takes as input N, the number of users in the system, and the security parameter k. All algorithms should run in time polynomial in k, unless otherwise specified. It is assumed that N is polynomial in k. The algorithm outputs public parameters P, a broadcast key BK, a tracing key TK, and N decryption keys $K_1, K_2, \ldots K_N$ for the users. We assume for notational simplicity that each of these keys contain the public parameters.

Encrypt$(\mathbf{BK},\ m)$**:** The broadcast algorithm takes as input the broadcast key BK, and a message m and outputs a ciphertext c. We assume that m comes from some samplable set M and c is in some recognizable set C (both M and C are defined by P).

Decrypt$(i,\ K_i,\ c)$**:** The decryption algorithm takes as input a user number i, the user's decryption key K_i, and a ciphertext c, and outputs the decryption of c into a message.

Trace$^{\mathcal{D}}(\mathbf{TK},\ \epsilon)$ The tracing algorithm is an oracle algorithm that is given as input the tracing key TK and a parameter ϵ, and it runs in time polynomial in the security parameter κ and $1/\epsilon$. Only values of ϵ that are polynomially related to k are considered valid inputs to *Trace*. The tracing algorithm outputs a set S which is a subset of $\{1, 2, \ldots, N\}$.

Security. We define security of the traitor tracing scheme in terms of the following natural requirements:

- **Correctness.** After setup, for all messages m, we have

$$\Pr[\mathit{Decrypt}(i, K_i, \mathit{Encrypt}(\mathrm{BK}, m)) = m] = 1$$

- **Secrecy.** We require that the encryption scheme defined implicitly in the traitor tracing scheme be semantically secure against chosen-plaintext attack. (As this is a standard notion, we don't detail it here.)

- **Traceability against arbitrary collusion.** For a given κ and ϵ (where $\epsilon = 1/f(\kappa)$ for some polynomial f), define the following game between a challenger and an adversary $\mathcal{A}$ (both are given κ and ϵ as input):

  1. The adversary outputs a number of users N, and a set $T = \{u_1, u_2, \ldots, u_t\} \subseteq \{1, \ldots, N\}$ of colluding users.
  2. The challenger runs $\mathit{Setup}(N, 1^\kappa)$ and provides BK, TK, and $K_{u_1}, \ldots, K_{u_t}$ to $\mathcal{A}$.
  3. The adversary $\mathcal{A}$ outputs a pirate decoder $\mathcal{D}$ which is a probabilistic circuit that takes as input ciphertexts in C and outputs some message m.
  4. The challenger now runs $\mathit{Trace}^{\mathcal{D}}(\mathrm{TK}, \epsilon)$ to obtain a set $S \subseteq \{1, \ldots, N\}$.

  We say that the adversary $\mathcal{A}$ wins the game if the following two conditions hold:

  – For a randomly chosen m in M, we have that $\Pr[\mathcal{D}(\mathit{Encrypt}(\mathrm{BK}, m)) = m] \geq \epsilon$.

  – The set S is either empty, or is not a subset of T.

  We require that for all polynomials f the probability of the adversary winning the game above is negligible in κ.

We note that the game above places no limit on the size of the coalition under the control of the adversary. Furthermore, the pirate decoder need not be perfect. It only needs to play valid content with probability ϵ. Finally, note that we are modeling a stateless (resettable) pirate decoder — the decoder is just an oracle and maintains no state between activations.

The black-box tracing model described above is often called the *full access model* — the tracer is given the decryptions output by $\mathcal{D}$. When the decoder $\mathcal{D}$ is a tamper resistant box, such as a a music player, the tracer does not get direct access to decryptions; it only sees whether a given ciphertext results in music being played or not. Thus, the full access model does not accurately capture tamper resistant decoders. To resolve this problem we define more restricted black-box tracing model called *minimal access tracing*. This model is similar to the game above with the exception that the challenger presents the tracing algorithm with a more restricted oracle $\mathcal{P}(\cdot, \cdot)$ which takes a message-ciphertext pair as input and outputs:

$$\mathcal{P}(m, c) = \begin{cases} 1 & \text{if } \mathcal{D}(c) = m \\ 0 & \text{otherwise} \end{cases}$$

We then modify Step 4 of the game above so that challenger runs $\mathit{Trace}^{\mathcal{P}}(\mathrm{TK}, \epsilon)$ to obtain a set $S \subseteq \{1, \ldots, N\}$. Consequently, in the minimal access game the tracing algorithm is given far more restricted access to $\mathcal{D}$. One can show [BF99] that this model accurately captures the problem of tracing a tamper resistant decoder.

Fortunately, PLBE enables black-box tracing in both the full access model and the minimal access model.

B Reducing Traitor Tracing to PLBE

We now show how to build a tracing traitors system out of a PLBE. We first show how to build the build the individual algorithms for Traitor Tracing from PLBE:

- *Setup* simply runs $Setup_{\text{LBE}}$ with the same parameters, and outputs BK = PK, TK = SK, and the user keys identically to the PLBE scheme.
- *Encrypt* executes algorithm $Broadcast_{\text{LBE}}$ to produce a ciphertext.
- *Decrypt* executes algorithm $Decrypt_{\text{LBE}}$ with the same parameters.
- *Trace*, when called with oracle $\mathcal{D}$, with tracing key TK = SK, and parameter ϵ, does the following:

  1. Initialize set S to the empty set.
  2. For $i = 1$ to $N + 1$, do the following:
     (a) The algorithm repeats the following $8nk/\epsilon$ times:
        i. Sample m from the message space M.
        ii. Let $c = Encrypt_{\text{LBE}}(\text{SK}, m, i)$.
        iii. Call oracle $\mathcal{D}$ on input c, and compare the output of $\mathcal{D}$ to m.
     (b) Based on the number of times $\mathcal{D}$ succeeds in decrypting the ciphertexts successfully, calculate p'_i as the observed probability of successful decryption of a random message above.
     (c) If $i > 1$ and if $|p'_i - p'_{i-1}| \geq \epsilon/4n$, then add $i - 1$ to set S.
  3. Output the set S.

Note that the running time of *Trace* is quadratic in n, but can be made $O(n \log n)$ using binary search instead of a linear scan.

Security. We prove that this traitor tracing scheme is secure. Correctness is immediate. Secrecy follows straightforwardly from the Message Hiding game, and a hybrid argument by means of the Index Hiding game.

We now show that traceability against arbitrary collusion also follows from the security of the PLBE scheme.

Suppose that the adversary wins the traceability game with non-negligible probability δ, for some parameter ϵ that is polynomially related to k (and therefore ϵ is a non-negligible function of k). From now on, we condition on the event that the adversary wins when it reaches Step 5 of the traceability game.

We first define p_i for $i = 1, \ldots, N + 1$ to be the actual probability that $\mathcal{D}$ decrypts random messages encrypted under index i.

We first observe that, in the running of the Trace algorithm, $p'_1 \geq \epsilon/2$ with probability $1-2^{-\Omega(k)}$. We know that, because of the win condition for the adversary, that the probability of $\mathcal{D}$ successfully decrypts a random message (which is p_1) is at least ϵ. Therefore this observation follows from a direct application of a Chernoff bound. We therefore condition on the event that $p'_1 \geq \epsilon/2$.

We next observe that $p'_{N+1} \leq \epsilon/4$, with probability $1 - 2^{-\Omega(k)}$. This follows directly from the message hiding game, which shows that $\mathcal{D}$ cannot decrypt messages encrypted with ID $N+1$ with non-negligible probability. Therefore p_{N+1} is negligible. The observation then follows from an application of a Chernoff bound. We therefore condition on the event that $p'_{N+1} \leq \epsilon/4$.

Since $p'_1 \geq \epsilon/2$ and $p'_{N+1} \leq \epsilon/4$, it follows that there exists at least one index i such that $p'_i - p'_{i+1} \geq \epsilon/4n$. Therefore Trace outputs a non-empty set S.

Finally, we must confirm that every element of S is from the set T corresponding to the secret keys given to the adversary. Suppose this is not the case, and there is some index $i \in S$ such that $i \notin T$. (In other words, the Trace algorithm has implicated an innocent user.)

This happens only if $p'_i - p'_{i+1} \geq \epsilon/4n$. By a pair of Chernoff bounds, we know that with probability $1 - 2^{-\Omega(k)}$, this implies that $p_i - p_{i+1} \geq \epsilon/8n$. We condition on this event.

However, this immediately contradicts the security of the PLBE scheme with respect to index hiding, as follows: We consider the index hiding game with respect to i, and construct the following adversary A'. Our adversary A' gives i to the challenger, and receives all user keys except for K_i. We then run A_1, and provide it the keys it requires (which does not include i by assumption). We thus obtain the circuit $\mathcal{D}$. We sample a random message m and provide it to the challenger. On input the ciphertext c, we use $\mathcal{D}$ to decrypt it. If $\mathcal{D}$ returns m, then we output a guess of 0, otherwise we return 1. By construction, and the probability bounds we have already shown, we have that the adversary's advantage is at least $\delta \cdot (\epsilon/8n) - 2^{-\Omega(k)}$, a contradiction.

Therefore the Trace algorithm cannot implicate any innocent users, and must implicate at least one member of the adversary's coalition. This completes the reduction.

C Proofs

C.1 Proof of Lemma 5.2

For this distinguishing experiment we will show that distinguishing between whether an encryption is to position (i, j) or $(i, j+1)$ is as hard as the 3-party Diffie-Hellman assumption. Since, the assumption is in a prime order group the simulator can now the factorization of n, the order of the group. The simulator will essentially run the core part of the simulation in the $\mathbb{G}_p$ subgroup and basically choose all values outside the subgroup for itself. Our formal proof follows.

Suppose there exists a t-time adversary $\mathcal{A}$ that breaks the Index Hiding game with advantage ϵ. Then we build a simulator as follows. The simulator receives the 3-party Diffie-Hellman challenge from the simulator as:

$$g_p, A = g_p^a, B = g_p^b, C = g_p^c, T.$$

The challenge will be given in a the subgroup of prime order p of a composite order group $n = pq$. The simulator is given the factors p, q.

Next, the simulator runs the Init phase and receieves the index (i, j) from $\mathcal{A}$. Since the game will be played in the subgroup $\mathbb{G}_p$, the simulator can choose for itself everything in the $\mathbb{G}_q$ subgroup. It chooses random generators $g_q, h_q \in \mathbb{G}_q$ and random exponents $\beta, r_{1,q}, \ldots, r_{m,q}, c_{1,q}, \ldots, c_{m,q} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{1,p}, \ldots, r'_{m,p}, c'_{1,p}, \ldots, c'_{m,p} \in \mathbb{Z}_p$.

The simulator is now able to create the public and secret keys as follows. It first publishes

$g = g_q g_p$ and $h = h_q B$. It creates the public keys:

$$E = g_q^{\beta} \qquad E_x = g_q^{\beta r_{x,q}} \qquad F_x = h_q^{\beta r_{x,q}} \qquad G_x = e(g_q, g_q)^{\beta \alpha_x} \qquad H_y = \begin{cases} g_q^{c_{y,q}} g_p^{c'_{y,p}} & : y \neq j \\ g_q^{c_{y,q}} C^{c'_{y,p}} & : y = j \end{cases}$$

Next, it creates the private keys for all users except (i, j) as:

$$K_{x,y} = \begin{cases} g^{\alpha_x} g_q^{r_{x,q} c_{y,q}} g_p^{r'_{x,p} c'_{y,p}} & : x \neq i, y \neq j \\ g^{\alpha_x} g_q^{r_{x,q} c_{y,q}} B^{r'_{x,p} c'_{y,p}} & : x = i, y \neq j \\ g^{\alpha_x} g_q^{r_{x,q} c_{y,q}} C^{r'_{x,p} c'_{y,p}} & : x \neq i, y = j \end{cases}$$

We note that all the simulator creates public and private with the same distribution as the real scheme.

In the challenge phase the adversary first gives the simulator a message $M \in \mathbb{G}_T$. The simulator then chooses exponents $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n^{(3)}$, and exponents $s_{i,q}, \ldots s_{m,q} \in \mathbb{Z}_q$ and $t_q \in \mathbb{Z}_q$. Additionally, it chooses random $s'_p \in \mathbb{Z}_p$, $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$.

It then creates the ciphertext as:

if $x > i$ $\quad R_x = g_q^{s_{x,q} r_{x,q}} \quad \tilde{R}_x = h_q^{s_{x,q} r_{x,q}} \quad A_x = g_q^{s_{x,q} t_q} \quad B_x = M e(g_q, g_q)^{\alpha_x s_{x,q} t_q}$

if $x = i$ $\quad R_x = g_q^{s_{x,q} r_{x,q}} g_p^{s'_p r'_{x,p}} \quad \tilde{R}_x = h_q^{s_{x,q} r_{x,q}} B^{s'_p r'_{x,p}} \quad A_x = g^{s_{x,q} t_q} A^{s'_p}$

$$B_x = M e(g_q, g_q)^{\alpha_x s_{x,q} t_{x,q}} e(g_p, A)^{\alpha_x s'_p}$$

if $x < i$ $\quad R_x = g^{v_{x,1}} \quad \tilde{R}_x = h^{v_{x,1}} \quad A_x = g^{v_{x,2}} \quad B_x = e(g, g)^{v_{x,3}}$

if $y > j$ $\quad C_y = g_q^{c_{y,q} t_q} h^{w'_y} \quad \tilde{C}_y = A^{-c'_{y,p}} g^{w'_y}$

if $y = j$ $\quad C_y = g_q^{c_{y,q} t_q} T h^{w'_y} \quad \tilde{C}_y = g^{w'_y}$

if $y < j$ $\quad C_y = g_q^{c_{y,q} t_q} g_p^{z_y} h^{w'_y} \quad \tilde{C}_y = g^{w'_y}$

If T forms a 3-party Diffie-Hellman tuple then the ciphertext is a well-formed encryption to the indices (i, j), otherwise if T is randomly chosen it is a encryption to $(i, j+1)$. The simulator will receive a guess γ from $\mathcal{A}$ and it will simply repeat this guess as its answer to the 3-party Diffie-Hellman game. The simulator's advantage in the Index Hiding game will be exactly equal to $\mathcal{A}$'s advantage. $\square$

C.2 Proof of Claim 5.5

In order to prove this claim we further refine our hybrid experiments by defining two more hybrid experiments.

- H_{2a}: Same as H_2 except B_i is multiplied by a random element $e(g_p, g)^z$.
- H_{2b}: Same as H_2 except B_i is multiplied by a random element $e(g, g)^z$.

Distinguishing between H_2 and H_{2a} We first show that if the (t, ϵ_{D3DH})-decision 3-party Diffie-Hellman assumption holds then no t-time adversary can distinguish between experiments H_2 and H_{2a}. We first note that if no t-time adversary can break the decision 3-party Diffie-Hellman with advantage greater than ϵ_{D3DH} then no t-time advsersary has greater advantage than ϵ_{D3DH} in the decisional Bilinear-Diffie Hellman (DBDH) assumption where the target, T is in $\mathbb{G}_T$.

Consider an adversary $\mathcal{A}$ that distinguishes between the two experiments with probability ϵ. We construct a simulator that plays the decisional DBDH game with advantage ϵ.

The simulator first takes in a DBHD challenge $g_p, A = g_p^a, B = g_p^b, C = g_p^c, T$. Again, the assumption is in a subgroup of order p and the simulator is given the factors p, q of n. The main idea of the simulation is that it will let $g_p^{r_i} = B$, $g_p^{\alpha_x} = g_p^{ab}$, $g_p^{t_p} = C$, and $g_p^{c_{i,p}} = A^{-1} g^{c'_{i,p}}$ where $c'_{1,p}, \ldots, c'_{m,p}$ are chosen by the simulation. The simulator will then be able to generate all keys, but still uses T as a challenge since g_p^c only appears in the term A_i.

The simulator chooses $g_q \in \mathbb{G}_q$, $d \in \mathbb{Z}_n$ and sets $h_q = g_q^d, h_p = g_p^d$ and lets $g = g_p g_q, h = g_p h_q$. Additionally, it chooses $\beta, c_{1,q}, \ldots, c_{m,q} r_{i,q} \in \mathbb{Z}_q$,$r_1, \ldots r_{i-1}, , r_{i+1}, \ldots, r_m \in \mathbb{Z}_n$, $\alpha_{i,q} \in \mathbb{Z}_q$, and $\alpha_1, \ldots \alpha_{i-1}, , \alpha_{i+1}, \ldots, \alpha_m \in \mathbb{Z}_p$.

The public parameters are published as:

$$g, h, E = g_q^{\beta}, E_1 = g_q^{\beta r_1}, \ldots, E_i = g_q^{\beta r_{i,q}}, \ldots, E_m = g_q^{\beta r_m}, F_1 = g_q^{d\beta r_1}, \ldots, F_i = g_q^{d\beta r_{i,q}}, \ldots, F_m = g_q^{d\beta r_m},$$

$$G_1 = e(g_q, g)^{\beta\alpha_1}, \ldots, G_i = e(g_q, g)^{\beta\alpha_{i,q}}, \ldots, G_m = e(g_q, g)^{\beta\alpha_m}, H_1 = g^{c_q} A^{-1} g^{c'_{1,p}}, \ldots, H_m = g^{c_q} A^{-1} g^{c'_{m,p}}.$$

The keys are created as:

$$K_{x,y} = \begin{cases} g^{\alpha_x} (g_q^{c_{y,q}} A^{-1} g_p^{c'_{y,p}})^{r_x} & : x \neq i \\ (g^{\alpha_{i,q}} g_q^{r_{i,q} c_{y,q}}) B^{c'_{y,p}} & : x = i \end{cases}$$

The simulator then receives a challenge message,M, from $\mathcal{A}$. It chooses $t_q, s_i, \ldots, s_m, w_1, \ldots, w_m$, $(v_{x,1}, v_{x,2}, v_{x,3}), \ldots, (v_{x,1}, v_{x,2}, v_{x,3})$ for itself. The simulator can now create all $C_y, \tilde{C}_y$ values in a straightforward manner since the $\mathbb{G}_p$ subgroup components are random. Similarly, all $R_x, \tilde{R}_x, A_x, B_X$ values for $x < i$ are just created radomly and all $R_x, \tilde{R}_x, A_x, B_x$ values for $x > i$ can be created by the simulators knowledge since they only draw from the $\mathbb{G}_q$ subgroup components which it knows.

Finally, it creates the target row ciphertext as:

$$R_x = (g^{r_{i,q}} B)^{s_i}, \tilde{R}_x = (g^{r_{i,q}} B)^{d s_i}, A_x = (g^{t_q} C)^{s_i}, B_X = M \left(e(g_q, g)^{\alpha + i, q t_q} T \right)^{s_i}$$

If T is a tuple $e(g, g)^{abc}$ then we are in experiment H_2, otherwise if it is random we are in experiment $H_{2,a}$. The simulator can then repeate the adversary's guess and play the DBDH game with advantage ϵ.

Distinguishing between H_{2a} and H_{2b} We now show that if there is an adversary that can distinguish between experiments H_{2a} and H_{2b} with advantage ϵ then we can build a simulator that can play the Bilinear Subgroup Decision game with advantage ϵ.

The simulator will first receive a Bilinear Subgroup Decision challenge g_p, g_q, T from the challenger. Using g_p, g_q it is able to set up all the system parameters just as the real setup algorithm does.

Next, it receives a challenge message, M. It first chooses all encryption variables and creates an encryption as in the H_2 experiment with one exception. For the B_i component it multiplies in the value T. If $T \in \mathbb{G}_{T,p}$ then the we are in hybrid experiment H_{2a}. Otherwise if $T \in \mathbb{G}_T$ then we are in hybrid experiment H_{2b}.

Therefore the simulator can use the adversary's guess to play the Bilinear Subgroup Decision game with advantage ϵ.

Distinguishing between H_{2b} and H_3 We now show that if there is an adversary that can distinguish between experiments H_{2b} and H_3 with advantage ϵ then we can build a simulator that can play the 3-Party Diffie-Hellman game with advantage ϵ.

The simulator first receives a 3-Party Diffie-Hellman challenge, $k_q, A = k_q^a, B = k_q^b, C = k_q^c, T \in \mathbb{G}_q^4$ (we rename the generator int the challenge to k for ease of exposition).

The simulator first chooses $d, \in \mathbb{Z}_n$,$\beta \in \mathbb{G}_q$, and $g_p \in \mathbb{G}_p$. It then sets $g = g_pA$,$h = (g_pA)^d$. Next, it chooses the secrets for the $\mathbb{G}_p$ subgroup: $r_{1,p}, \ldots, r_{m,p}, c_{1,p}, \ldots, c_{m,p}, \alpha_{1,p}, \alpha_{m,p} \in \mathbb{Z}_p$. Then, it chooses $r'_{1,q}, \ldots, r'_{m,q}, c'_{1,q}, \ldots, c'_{m,q}, \alpha'_{1,q}, \alpha'_{m,q} \in \mathbb{Z}_q$. The simulator can now publish the parameters as:

$$g, h, E = k_q^{\beta}, E_i = k_q^{\beta r'_{i,q}}, \text{ For } x \neq i : E_x = A^{\beta r'_{m,q}},$$

$$F_i = k_q^{d\beta r'_{i,q}}, \text{ For } x \neq i : E_x = A^{d\beta r'_{m,q}},$$

$$G_1 = e(A, A)^{\beta\alpha_1}, \ldots, G_m = e(A, A)^{\beta\alpha_m},$$

$$H_1 = k_q^{c'_{1,q}} g_p^{c_{1,p}}, \ldots, H_m = k_q^{c'_{x,q}} g_p^{c_{m,p}}.$$

Keys are generated as

$$K_{x,y} = \begin{cases} (Ag_p)^{\alpha_x} g_p^{r_{x,p}c_{y,p}} k^{r'_{x,q}c'_{y,p}} & : x = i \\ (Ag_p)^{\alpha_x} g_p^{r_{x,p}c_{y,p}} A^{r'_{x,q}c'_{y,p}} & : x \neq i \end{cases}$$

The simulator next receieves a challenge message M. It then chooses random $t_p \in Z_p$, $(v_{1,1}, v_{1,2}, v_{1,3})$,...,$(v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n$, $w_1, \ldots, w_m \in \mathbb{Z}_n$,$z'_1, \ldots, z'_m \in \mathbb{Z}_n$, and $s'_{i+1,q}, \ldots, s'_{m,q} \in \mathbb{Z}_q$.

It creates the challenge ciphertext as:

if $x > i$	$R_x = k_q^{s'_{x,q}r_{x,q}}$	$\tilde{R}_x = k_q^{ds'_{x,q}r_{x,q}}$	$A_x = B^{s'_{x,q}}$	$B_x = Me(A, B)^{\alpha_x s'_{x,q}}$
if $x = i$	$R_x = C^{s'_{x,q}} g_p^{s_p r_{x,q}}$	$\tilde{R}_x = C^{ds'_{x,q}} g_p^{ds_p r_{x,q}}$	$A_x = Tg_p^{\delta}$	$B_x = e(g, g)^{\gamma}$
if $x < i$	$R_x = g^{v_{x,1}}$	$\tilde{R}_x = h^{v_{x,1}}$	$A_x = g^{v_{x,2}}$	$B_x = e(g, g)^{v_{x,3}}$
$\forall y$	$C_y = B^{c'_{y,q}} g_p^{z'_y} h^{w'_y}$	$\tilde{C}_y = g^{w'_y}$		

If $T = k^{abc}$ then we are in experiment H_{2b}, otherwise if T is a random element of $\mathbb{G}_q$ then we are in experiment H_3. Therefore, our simulator can use the adversary's response to get an ϵ advantage in the 3-party Diffie-Hellman game.

Putting it together By the assumptions above we can now bound the Adversary's advantage in distinguishing between experiments H_2 and H_3 by $2\epsilon_{D3DH} + \epsilon_{BSD}$, proving our claim. □

C.3 Proof of Claim 5.6

To prove the claim we consider a sequence of hybrid experiments $H_{3,m+1}, \ldots, H_{3,1}$, where in experiment $H_{3,j}$ all ciphertext C_y values are well formed in the $\mathbb{G}_p$ subgroup for $y \geq j$ and random in the subgroup for $y < j$, and the rest of the experiment is built as the ciphertext from experiment H_3. We observe that experiments H_3 and $H_{3,m+1}$ are equivalent and that experiments H_4 and $H_{3,1}$ are equivalent. Therefore we can bound an adversary's advantage in distinguishing between H_3 and H_4 as m times his advantage in distinguishing between any two sub-experiments.

Suppose there exits an adversary $\mathcal{A}$ that for some j distinguishes between $H_{3,j}$ and $H_{3,j+1}$ that succeeds with advantage ϵ. We can bound its advantage with a proof similar to that of Lemma 5.2, however, this will be even simpler since there is no target row in the hybrid experiment.

We construct a simulator that play the 3-party Diffie-Hellman game. The simulator first receives the 3-party Diffie-Hellman challenge from the simulator as:

$$g_p, A = g_p^a, B = g_p^b, C = g_p^c, T.$$

Since the game will be played in the subgroup $\mathbb{G}_p$, the simulator can choose for itself everything in the $\mathbb{G}_q$ subgroup. It chooses random generators $g_q, h_q \in \mathbb{G}_q$ and random exponents $\beta, r_{1,q}, \ldots, r_{m,q}, c_{1,q}, \ldots, c_{m,q} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{1,p}, \ldots, r'_{m,p}, c'_{1,p}, \ldots, c'_{m,p} \in \mathbb{Z}_p$.

The simulator is now able to create the public and secret keys as follows. It first publishes $g = g_q g_p$ and $h = h_q B$. It creates the public keys:

$$E = g_q^\beta \qquad E_x = g_q^{\beta, r_{x,q}} \qquad F_x = h_q^{\beta, r_{x,q}} \qquad G_x = e(g_q, g_q)^{\beta \alpha_x} \qquad H_y = \begin{cases} g_q^{c_{y,q}} g_p^{c'_{y,p}} & : y \neq j \\ g_q^{c_{y,q}} C^{c'_{y,p}} & : y = j \end{cases}$$

Next, it creates the private keys as:

$$K_{x,y} = \begin{cases} g^{\alpha_x} g_q^{r_{x,q} c_{y,q}} g_q^{r'_{x,p} c'_{y,q}} & : y \neq j \\ g^{\alpha_x} g_q^{r_{x,q} c_{y,q}} C^{r'_{x,p} c'_{y,q}} & : y = j \end{cases}$$

In the challenge phase the adversary first gives the simulator a message $M \in \mathbb{G}_T$. The simulator first chooses exponents $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{i,1}, v_{i,2}, v_{i,3}) \in \mathbb{Z}_n$, and exponents $s_{i+1,q}, \ldots, s_{m,q} \in \mathbb{Z}_q$ and $t_q \in \mathbb{Z}_q$. Next, it chooses random $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$.

It then creates the ciphertext as:

if $x > i+1$	$R_x = g_q^{s_{x,q} r_{x,q}}$	$\tilde{R}_x = h_q^{s_{x,q} r_{x,q}}$	$A_x = g_q^{s_{x,q} t_q}$	$B_x = M e(g_q, g_q)^{\alpha_x s_{x,q} t_q}$
if $x \leq i$	$R_x = g^{v_{x,1}}$	$\tilde{R}_x = h^{v_{x,1}}$	$A_x = g^{v_{x,2}}$	$B_x = e(g,g)^{v_{x,3}}$
if $y > j$	$C_y = g_q^{c_{y,q} t_q} h^{w'_y}$	$\tilde{C}_y = A^{-c'_{y,p}} g^{w'_y}$		
if $y = j$	$C_y = g_q^{c_{y,q} t_q} T h^{w'_y}$	$\tilde{C}_y = g^{w'_y}$		
if $y < j$	$C_y = g_q^{c_{y,q} t_q} g_p^{z_y} h^{w'_y}$	$\tilde{C}_y = g^{w'_y}$		

If T forms a 3-party Diffie-Hellman tuple then we simulated $H_{3,j}$, otherwise if T is randomly chosen we simulated $H_{3,j+1}$. The simulator will receive a guess from $\mathcal{A}$ and it will simply repeat this guess as its answer to the 3-party Diffie-Hellman game. The simulator's advantage will be exactly equal to $\mathcal{A}$'s advantage.

Therefore, we can bound an adversary's advantage of distinguishing between H_3 and H_4 as $m \cdot \epsilon_{D3DH}$. □

C.4 Proof of Claim 5.7

Suppose there exists an adversary $\mathcal{A}$ that distinguishes between H_4 and H_5 with advantage ϵ. Then we build a simulator that plays the Subgroup Decision game. In this game the simulator does not know the factors of n.

The simulator first takes in the challenge g, g'_q, T. We note that $g = g_p g_q$ for some $g_p \in \mathbb{G}_p$ and $g_q \in \mathbb{G}_q$ and that $g'_q = g_q^\beta$ for some $\beta \in Z_q$. Next, it chooses $d \in Z_n$ and lets $h = g^d$. Then it chooses random exponents $r_1, \ldots, r_m, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$.

The public key includes the description of the group and the following elements:

$$g, h, E = g'_q E_1 = g_q'^{r_1}, \ldots, E_m = g_q'^{r_m}, F_1 = g_q'^{r_1}, \ldots, F_m = g_q'^{d r_m}, G_1 = e(g'_q, g)^{\alpha_1}, \ldots, G_m = e(g'_q, g)^{\alpha_m}.$$

$$H_1 = g^{c_1}, \ldots, H_m = g^{c_m}.$$

The private key for user x, y is generated as $K_{x,y} = g^{\alpha_x} g^{r_x c_y}$. All public and private keys are created with the same distribution as in the real simulation.

The simulator receives a message M in the challenge phase and chooses random $t \in \mathbb{Z}_n$, $w_1, \ldots, w_m$, $s_1, \ldots, s_m \in Z_n$, and $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n$.

The challenge ciphertext for all $C_y, \tilde{C}_y$ and $R_x, \tilde{R}_x, A_x, B_x$ for $x \leq i$ are encrypted created as in the real encryption. This is easy since $j = 1$ and all rows less than or equal to i are just randomly created. For $x > i + 1$ $R_x, \tilde{R}_x, A_x, B_x$ are created in a straightforward manner using g'_q.

Finally, the encryption values for row $i + 1$ are created as:

$$R_{i+1} = T^{s_{i+1} r_{i+1}}, \tilde{R}_{i+1} = T^{d s_{i+1} r_{i+1}}, A_{i+1} = T^{s_{i+1} t}, B_{i+1} = Me(T, g)^{\alpha_{i+1} s_{i+1} t}.$$

If T is a random element of $\mathbb{G}_q$ then we simulated experiment H_4, otherwise we simulated experiment H_5.

□