Fully Collusion Resistant Traitor Tracing With Short Ciphertexts and Private Keys

Dan Boneh *
dabo@cs.stanford.edu

Amit Sahai †
sahai@cs.ucla.edu

Brent Waters
bwaters@csl.sri.com

May 16, 2006

Abstract

We construct a 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 first introduce a simpler primitive we call *private linear broadcast encryption* (PLBE) and show that any PLBE gives a tracing traitors system with the same parameters. We then show how to build a PLBE system with $O(\sqrt{N})$ size ciphertexts. Our system uses bilinear maps in groups of composite order.

1 Introduction

Traitor tracing systems, introduced by Chor, Fiat, and Naor [9], 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 public broadcasting key BK. Any certified player can decrypt using its embedded secret key K_i. Certified players, of course, could 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 decrypting encrypted DVD content.

This is where traitor tracing systems come in — when the pirate decoder is found, the distributor can run a *tracing* algorithm that interacts with the pirate decoder 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.

We give a precise description of traitor tracing systems in Appendix A. For now we give some intuition that will help explain our results. A traitor tracing system consists of four algorithms *Setup*, *Encrypt*, *Decrypt*, and *Trace*. The setup algorithm generates the broadcaster's key BK, a

*Supported by NSF and the Packard Foundation.

†This research was supported by the NSF Cybertrust and ITR Programs, an Alfred P. Sloan Research Fellowship, and a generous equipment grant from Intel.

tracing key TK, and N recipient keys $\mathrm{K}_1, \ldots, \mathrm{K}_N$. 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 algorithm that takes TK as input and interacts with a pirate decoder, treating it as a black-box oracle. It outputs the index $i \in \{1, \ldots, N\}$ of a key K_i that was used to create the pirate decoder.

Traitor tracing systems can have a number of properties informally defined below:

- **Public vs. Secret BK:** Some traitor tracing systems require that the broadcaster's key BK remain secret; Other systems [20, 3, 24, 19, 11, 21, 33, 8] support a public BK so that anyone can create broadcast data. Combinatorial systems such as [9, 23, 30, 31, 13, 14, 10, 27, 2, 29, 28, 22] 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 traitor tracing 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 [25, 26, 34, 18, 8]. In fact, our system can be made publicly traceable by making a slightly stronger complexity assumption [7]. In these settings TK is part of the public key.

- **Collusion resistance:** a traitor tracing 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 oracle. Clearly, the pirate is free to build the decoder however it wants — the pirate could obfuscate the code, use tamper resistant hardware, and try to randomize the secret keys at its disposal. Hence, it is safest to treat the pirate decoder as a black box and the tracing algorithm cannot look into the inner-workings of the decoder. More precisely, the tracer gives the decoder encrypted content and the decoder responds with valid or invalid. In real world settings, valid means the decoder plays the given encrypted content and invalid means it 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, software decoders, such as DeCSS, cannot keep any state. Pirate decoders embedded in tamper resistant hardware, such as 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. Kiayias and Yung [17] 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.

In this paper we focus on fully collusion resistant traitor tracing systems. That is, systems that remain secure no matter how many keys are at the disposal of the pirate. Existing traitor tracing systems are not designed to handle arbitrary collusions. When the collusion bound t comes close to N, most existing systems require ciphertext size linear in the number of users, which is no better than the trivial traitor tracing system.

Our results. We construct a practical fully collusion resistant traitor tracing system that has sub-linear size ciphertexts. 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 [17]. We give a precise definition of these properties in Appendix A. The system uses bilinear groups of composite order introduced in [4].

We prove security of our tracing algorithm using a tracing technique previously used in [3, 22, 17]. To formalize this technique, we introduce a new primitive called *Private Linear Broadcast Encryption*, or PLBE for short, which is conceptually a simpler primitive than traitor tracing. We show that any secure PLBE gives a (black-box) traitor tracing system. Roughly speaking, a PLBE is a broadcast encryption system [12] that can only broadcast to "linear" sets, that is sets of the form $\{i, i+1, \ldots, N\}$ for some $i = 1, \ldots, N+1$. Thus, a PLBE enables the broadcaster to create ciphertexts that can only be decrypted properly under keys $\mathrm{K}_i, \mathrm{K}_{i+1}, \ldots, \mathrm{K}_N$. A broadcast to everyone, for example, is encrypted using $i = 1$. The main security requirement is that the system should be *private* [1]: a ciphertext should reveal no non-trivial information about the recipient set. That is, a broadcast to users $\{i, \ldots, N\}$ should reveal no non-trivial information about i. We give a precise definition in the next section and show that any secure PLBE gives a secure (black-box) traitor tracing system. In the remainder of the paper we focus on constructing a secure PLBE.

Related work. Traitor tracing systems generally fall into two categories: combinatorial, as in [9, 23, 30, 31, 13, 14, 10, 27, 2, 29, 28, 22], and algebraic, as in [20, 3, 24, 19, 11, 21, 33, 8]. The broadcaster's key BK in combinatorial systems can be either secret or public. Algebraic traitor tracing use public-key techniques and are often more efficient than the public-key instantiations of combinatorial schemes. Some systems, including ours, only provide tracing capabilities. Other systems [24, 22, 16, 15, 11] 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 [19] 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 traitor tracing system and encrypting the long data by using a symmetric cipher with the message-key).

Many traitor tracing 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 [25, 26, 34, 18, 8]. Similarly, many traitor tracing systems, including ours, assume that the pirate decoder is stateless. Kiayias and Yung [17] show how to strengthen traitor tracing systems to handle stateful decoders.

Finally, we note that binary fingerprinting codes [6, 32] are closely related to traitor tracing (binary refers to the fact that the code is defined over a binary alphabet). In fact, it is known [5] that any binary fingerprinting code gives rise to a fully collusion-resistant traitor tracing system with *constant* size ciphertexts. The private key size, unfortunately, is quite large. Using [6] the private key size is $\tilde{O}(N^3)$ and using [32] it is $\tilde{O}(N^2)$.

2 Traitor Tracing and Private Linear Broadcast Encryption

In Appendix A we review the precise definition of a traitor tracing system. However, instead of directly building a traitor tracing system we build a simpler primitive called *Private Linear Broadcast Encryption* (PLBE). We first define secure PLBEs below and then briefly explain how a PLBE is used for traitor tracing. The resulting tracing algorithm makes explicit a tracing technique used in [3, 22, 17]. Then in the remainder of the paper we build a secure PLBE.

2.1 Description of Private Linear Broadcast Encryption

A PLBE is comprised of the following four algorithms:

Setup$_{\text{LBE}}(N, \lambda)$ The setup algorithm takes as input N, the number of users in the system, and the security parameter λ. The algorithm runs in polynomial time in λ and outputs a public key PK, a secret key TK, and private keys $\mathrm{K}_1, \ldots, \mathrm{K}_N$, where K_u is given to user u.

Encrypt$_{\text{LBE}}(\mathrm{PK}, M)$ Takes as input a public key PK, and a message M and outputs a ciphertext C. This algorithm is used to encrypt a message to all N users.

TrEncrypt$_{\text{LBE}}(\mathrm{TK}, i, M)$ Takes as input a secret key TK, an integer i satisfying $1 \leq i \leq N+1$, and a message M. It outputs a ciphertext C. This algorithm encrypts a message to a set $\{i, \ldots, N\}$ and is primarily used for traitor tracing. We will require below that *TrEncrypt*$_{\text{LBE}}(\mathrm{TK}, 1, M)$ outputs a distribution on ciphertexts that is indistinguishable from the distribution generated by *Encrypt*$_{\text{LBE}}(\mathrm{PK}, M)$.

Decrypt$_{\text{LBE}}(j, \mathrm{K}_j, C, \mathrm{PK})$ Takes as input a private key K_j for user j, a ciphertext C, and the public key PK. The algorithm outputs a message M or $\perp$.

The system must satisfy the following **correctness property**:
for all $i, j \in \{1, \ldots, N+1\}$, where $j \leq N$, and all messages M:

Let $\big(\mathrm{PK}, \mathrm{TK}, (\mathrm{K}_1, \ldots, \mathrm{K}_N)\big) \stackrel{\mathrm{R}}{\leftarrow} \textit{Setup}_{\text{LBE}}(N, \lambda)$

and let $C \stackrel{\mathrm{R}}{\leftarrow} \textit{TrEncrypt}_{\text{LBE}}(\mathrm{TK}, i, M)$.

If $\mathbf{j} \geq \mathbf{i}$ then $\textit{Decrypt}_{\text{LBE}}(j, \mathrm{K}_j, C, \mathrm{PK}) = M$.

Security. We define security of a PLBE system using three games. The first game just captures a consistency property which says that *TrEncrypt*$_{\text{LBE}}(\mathrm{TK}, 1, M)$ outputs a distribution on ciphertexts that is indistinguishable from the distribution generated by *Encrypt*$_{\text{LBE}}(\mathrm{PK}, M)$. The second game is a **message hiding game** and says that a ciphertext created using index $i = N+1$ is unreadable by anyone. The third game is an **index hiding game** and captures the intuition that a broadcast ciphertext created using index i reveals no non-trivial information about i. We will consider all these games for a fixed number of users N.

Game 1 – Indistinguishability. The first game says that the output of *TrEncrypt*$_{\text{LBE}}(\mathrm{TK}, 1, M)$ is indistinguishable from *Encrypt*$_{\text{LBE}}(\mathrm{PK}, M)$. The game proceeds as follows:

- **Setup** The challenger runs the $Setup_{\text{LBE}}$ algorithm and gives the adversary PK and the set of all private keys $\{K_1, \ldots, K_N\}$.
- **Challenge** The adversary gives the challenger a message M. The challenger flips a coin $\beta \in \{0,1\}$ and computes

$$c \xleftarrow{R} \begin{cases} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, 1, M) & \text{if } \beta = 0, \\ \mathit{Encrypt}_{\text{LBE}}(\text{PK}, M) & \text{if } \beta = 1. \end{cases}$$

  It gives C to the adversary.
- **Guess** The adversary returns a guess $\beta' \in \{0,1\}$ of β.

We define the advantage of adversary $\mathcal{A}$ as $\mathsf{Adv}_{\text{CG}} = |\Pr[\beta' = \beta] - 1/2|$.

Game 2 – Message Hiding. The second game says that an adversary cannot break semantic security when encrypting using index $i = N + 1$. The game proceeds as follows:

- **Setup** The challenger runs the $Setup_{\text{LBE}}$ algorithm and gives the adversary PK and all secret keys $\{K_1, \ldots, K_N\}$.
- **Challenge** The adversary outputs two equal length messages M_0, M_1. The challenger flips a coin $\beta \in \{0,1\}$ and sets $C \xleftarrow{R} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, N+1, M_\beta)$. The challenger gives C to the adversary.
- **Guess** The adversary returns a guess $\beta' \in \{0,1\}$ of β.

We define the advantage of adversary $\mathcal{A}$ as $\mathsf{Adv}_{\text{MH}} = |\Pr[\beta' = \beta] - 1/2|$.

Game 3 – Index Hiding. The third game says that an adversary cannot distinguish between an encryption to index i and one to index $i + 1$ without the key K_i. The game takes as input a parameter $i \in \{1, \ldots, N\}$ which is given to both the challenger and the adversary. The game proceeds as follows:

- **Setup** The challenger runs the $Setup_{\text{LBE}}$ algorithm and gives the adversary PK and the set of private keys $\{K_j \text{ s.t. } j \neq i\}$.
- **Challenge** The adversary outputs a message M. The challenger flips a coin $\beta \in \{0,1\}$ and computes $C \xleftarrow{R} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, i + \beta, M)$. The challenger returns C to the adversary.
- **Guess** The adversary returns a guess $\beta' \in \{0,1\}$ of β.

We define the advantage of adversary $\mathcal{A}$ as $\mathsf{Adv}_{\text{IH}}[i] = |\Pr[\beta' = \beta] - 1/2|$.

Now that the three games are established we are ready to define secure PLBE.

Definition 2.1. *We say that an N-user Private Linear Broadcast System (PLBE) is secure if for all polynomial time adversaries $\mathcal{A}$ we have that Adv_{CG}, and Adv_{MH}, and $\mathsf{Adv}_{IH}[i]$ for $i = 1, \ldots, N$, are negligible functions of λ.*

2.2 Reducing Traitor Tracing to PLBE

We briefly show that a secure PLBE gives a secure traitor tracing system. Let

$$\mathcal{E} = (Setup_{\text{LBE}},\ Encrypt_{\text{LBE}},\ TrEncrypt_{\text{LBE}},\ Decrypt_{\text{LBE}})$$

be a secure PLBE system. The derived traitor tracing system is defined as follows (we use the notation of Appendix A):

- *Setup* simply runs $Setup_{\text{LBE}}$ with the same parameters, and outputs PK as the public encryption key, TK as the secret tracing key, and the user keys identically to the PLBE scheme.
- *Encrypt* and *Decrypt* run algorithms $Encrypt_{\text{LBE}}$ and $Decrypt_{\text{LBE}}$ respectively with the same parameters.
- $Trace^{\mathcal{D}}(\text{TK},\ \epsilon)$, when called with oracle $\mathcal{D}$, and inputs TK and $\epsilon > 0$, does the following:
  1. For $i = 1$ to $N + 1$, do the following:
     (a) Let $\text{cnt} \leftarrow 0$
     (b) Repeat the following steps $W \leftarrow 8\lambda(N/\epsilon)^2$ times:
        i. Sample M from the finite message space at random
        ii. Let $C \xleftarrow{\text{R}} TrEncrypt_{\text{LBE}}(\text{TK}, i, M)$
        iii. Call oracle $\mathcal{D}$ on input C and if $\mathcal{D}(C) = M$ then $\text{cnt} \leftarrow \text{cnt} + 1$
     (c) Let $\hat{p}_i \in [0, 1]$ be the fraction of times that $\mathcal{D}$ decrypted the ciphertexts correctly. That is, $\hat{p}_i = (\text{cnt}/W)$.
  2. Let S be the set of all $i \in \{1, \dots, N\}$ for which $\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)$.
  3. Output the set S as the set of guilty colluders.

Note that the running time of *Trace* is cubic in N. It can be made essentially quadratic in N using binary search instead of a linear scan. A tighter analysis shows that the running time need only be quadratic in the number of colluders t.

Security. The following two lemmas prove that this traitor tracing scheme is secure. The first lemma proves semantic security, namely that Adv_{SS} is negligible. Note that we did not explicitly require that a PLBE be semantically secure against a chosen plaintext attack to an outsider who possess no secret keys. Nevertheless, semantic security does follow straightforwardly from the three games used to define PLBE using a hybrid argument by means of the Index Hiding game. We give the proof in Appendix B.

Lemma 2.2. *If the given PLBE is secure then for the traitor tracing system above,* Adv_{SS} *defined in Appendix A is negligible.*

Next, we argue that traceability against arbitrary collusion follows from the security of the PLBE scheme. Here we sketch the proof and then give the details in Appendix B.1.

Lemma 2.3. *If the given PLBE is secure then for the traitor tracing system above,* Adv_{TR} *defined in Appendix A is negligible.*

Proof Sketch. We show that the probability of winning the traceability game defined in Appendix A is negligible. Let $p_i = \Pr[\mathcal{D}(\mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, i, M)) = M]$ where the probability is over the random choice of M in a finite message space. We know that that $p_1 \geq \epsilon$ and p_{N+1} is negligible. The former follows from the fact that $\mathcal{D}$ is an ϵ-useful decoder. The later follows directly from the PLBE message hiding game. Then there must exist some $j \in \{1, \ldots, N\}$ such that $p_j - p_{j+1} \geq \epsilon/(2N)$. By the Chernoff bound it follows that with overwhelming probability, $\hat{p}_j - \hat{p}_{j+1} \geq \epsilon/(4N)$. Hence, the set S output by $\mathit{Trace}^{\mathcal{D}}(\text{TK}, \ \epsilon)$ is non-empty.

Using the notation of Game 2 from Appendix A, it remains to show that whenever $\hat{p}_j - \hat{p}_{j+1} > \epsilon/(4N)$ we have that $j \in T$. For such j we know, by Chernoff, that with overwhelming probability $p_j - p_{j+1} \geq \epsilon/(8N)$. Hence, $\mathcal{D}$ is able to distinguish $\mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, j, M)$ from $\mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, j+1, M)$ for random M. But since the PLBE is secure, the index hiding game implies that these two distributions are indistinguishable, unless one has K_j. It follows that the pirate who built $\mathcal{D}$ must have had K_j and therefore $j \in T$, as required. □

3 Background and complexity assumptions

Our traitor tracing system uses bilinear group of composite order. We review the definition of such groups and then state our complexity assumptions. We follow [4] in which composite order bilinear groups were first introduced.

Bilinear groups of composite order. Let $\mathcal{G}$ be an algorithm called a *group generator* that takes as input a security parameter $\lambda \in \mathbb{Z}^{>0}$ and outputs a tuple $(p, q, \mathbb{G}, \mathbb{G}_T, e)$ where p, q are two distinct primes, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order $n = pq$, and e is a function $e : \mathbb{G}^2 \to \mathbb{G}_T$ satisfying the following properties:

- (Bilinear) $\forall u, v \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$.
- (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order n in $\mathbb{G}_T$.

We assume that the group action in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all computable in polynomial time in λ. Furthermore, we assume that the description of $\mathbb{G}$ and $\mathbb{G}_T$ includes a generator of $\mathbb{G}$ and $\mathbb{G}_T$ respectively.

To summarize, $\mathcal{G}$ outputs the description of a group $\mathbb{G}$ of order $n = pq$ with an efficiently computable bilinear map. We will use the notation $\mathbb{G}_p, \mathbb{G}_q$ to denote the respective subgroups of order p and order q of $\mathbb{G}$.

3.1 Complexity assumptions

Next we review three complexity assumptions needed for proving security of our system. The first assumption is in a prime order subgroup $\mathbb{G}_p$ and the last two are over the composite group $\mathbb{G}$.

Decision 3-party Diffie-Hellman Assumption. For a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{R}} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{\text{R}} \mathbb{G}_p \\
&a, b, c \xleftarrow{\text{R}} \mathbb{Z}_p \\
&\bar{Z} \leftarrow \big((n, \mathbb{G}, \mathbb{G}_T, e), \;\; g_p, \;\; g_p^a, \;\; g_p^b, \;\; g_p^c\big) \\
&T \leftarrow g_p^{abc} \\
&\text{Output} \;\; (\bar{Z}, \; T)
\end{aligned}$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the decision 3-party Diffie-Hellman problem for $\mathcal{G}$ as:

$$\mathsf{D3DH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr[\mathcal{A}(\bar{Z}, T) = 1] - \Pr[\mathcal{A}(\bar{Z}, R) = 1] \right|$$

where $(\bar{Z}, T) \xleftarrow{\text{R}} P(\lambda)$ and $R \xleftarrow{\text{R}} \mathbb{G}_p$.

Definition 3.1. *We say that $\mathcal{G}$ satisfies the decision 3-party Diffie-Hellman assumption (D3DH) if for any polynomial time algorithm $\mathcal{A}$ we have that $\mathsf{D3DH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ.*

Subgroup Decision Assumption. The subgroup decision problem introduced in [4] states that a random element in $\mathbb{G}_q$ is indistinguishable from a random element in $\mathbb{G}$. More precisely, for a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{R}} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad \bar{Z} \leftarrow (n, \mathbb{G}, \mathbb{G}_T, e) \\
&\text{Output} \;\; \bar{Z}
\end{aligned}$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the decision 3-party Diffie-Hellman problem for $\mathcal{G}$ as:

$$\mathsf{D3DH\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr[\mathcal{A}(\bar{Z}, h, T) = 1] - \Pr[\mathcal{A}(\bar{Z}, h, R) = 1] \right|$$

where $\bar{Z} \xleftarrow{\text{R}} P(\lambda)$, $\;h, T \xleftarrow{\text{R}} \mathbb{G}_q$, and $R \xleftarrow{\text{R}} \mathbb{G}$.

Definition 3.2. *We say that $\mathcal{G}$ satisfies the subgroup decision assumption (SD) if for any polynomial time algorithm $\mathcal{A}$ we have that $\mathsf{SD\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ.*

Our statement of the SD assumption slightly deviates from the definition in [4]. In our definition the adversary is also given an element h that is known to be in $\mathbb{G}_q$. However, a simple hybrid argument shows that these two ways of stating the assumption are equivalent (within a factor of 2). Suppose that an adversary $\mathcal{A}$ has advantage $\mathsf{SD\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) = 2\epsilon$. Then either

$$\left| \Pr[\mathcal{A}(\bar{Z}, \; R, T \;) = 1 \; : \; R, T \xleftarrow{\text{R}} \mathbb{G}_q] \; - \; \Pr[\mathcal{A}(\bar{Z}, \; R, T \;) = 1 \; : \; R, T \xleftarrow{\text{R}} \mathbb{G}] \right| \geq \epsilon$$

or

$$\left| \Pr[\mathcal{A}(\bar{Z}, \; R, T \;) = 1 \; : \; R, T \xleftarrow{\text{R}} \mathbb{G}] \; - \; \Pr[\mathcal{A}(\bar{Z}, \; R, T \;) = 1 \; : \; R \xleftarrow{\text{R}} \mathbb{G}_q, \; T \xleftarrow{\text{R}} \mathbb{G}] \right| \geq \epsilon$$

However, if either of these statements holds then $\mathcal{A}$ clearly has an advantage ϵ in the subgroup decision problem of [4]. For ease of exposition, in our proofs we will use our way of stating the assumption.

Bilinear Subgroup Decision Assumption. The Bilinear Subgroup Decision (BSD) problem is states that a random order p element in $\mathbb{G}_T$ is indistiguishable from a random element in $\mathbb{G}_T$ when $g_p, g_q \in \mathbb{G}$ are given. More precisely, for a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{\text{R}} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{\text{R}} \mathbb{G}_p, \quad g_q \xleftarrow{\text{R}} \mathbb{G}_q,$$
$$\bar{Z} \leftarrow ((n, \mathbb{G}, \mathbb{G}_T, e), \;\; g_p, \; g_q)$$
$$\text{Output} \;\; \bar{Z}$$

For an algorithm $\mathcal{A}$, define $\mathcal{A}$'s advantage in solving the bilinear subgroup decision problem for $\mathcal{G}$ as:

$$\mathsf{BSD\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr[\mathcal{A}(\bar{Z}, \; e(T, g)) = 1] - \Pr[\mathcal{A}(\bar{Z}, \; e(R, g)) = 1] \right|$$

where $\bar{Z} \xleftarrow{\text{R}} P(\lambda)$, $T \xleftarrow{\text{R}} \mathbb{G}_p$, and $R \xleftarrow{\text{R}} \mathbb{G}$. Here g is an arbitrary generator of $\mathbb{G}$.

Definition 3.3. *We say that $\mathcal{G}$ satisfies the bilinear subgroup decision assumption (BSD) if for any polynomial time algorithm $\mathcal{A}$ we have that $\mathsf{BSD\,Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ.*

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 Section 2 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 ciphertexts this is easy: each user has a unique 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 will be associated with a completely different portions of the ciphertext and changing u's component has no effect on their ability to decrypt.

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 in size, we cannot let every user have a component of the ciphertext that is dedicated for them alone. Intuitively, ciphertext components must be "shared" amongst users. Therefore, we 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 can think of this as a "row-major" ordering.

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 for referring to users and encryptions will be a purely 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. *The key algebraic fact that underlies our scheme is that if g_p is any element from the order p subgroup (which we call $\mathbb{G}_p$) and g_q is any element from the order q subgroup (which we call $\mathbb{G}_q$), then we have: $e(g_p, g_q) = 1$.*

When the $\mathit{TrEncrypt}_{\text{LBE}}$ 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 ciphertext components from column y. The encryption algorithm works by creating ciphertexts in the following way.

Column Ciphertext Components. (1) Ciphertexts for columns greater than or equal to j are "well formed" in both subgroups. (2) However, for a column that is less than j, the encryption algorithm will create a ciphertext that is well formed in the $\mathbb{G}_q$ subgroup, but random in the $\mathbb{G}_p$ subgroup.

Row Ciphertext Components. (1) 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. (2) The ciphertext components for row i are well formed in both subgroups. 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 ($\mathbb{G}_p$) part of the column ciphertext will scramble the result of pairing the row and column ciphertexts together. (3) Finally, for rows greater than i the ciphertext components will be well formed elements in the $\mathbb{G}_q$ subgroup only. A user with row index greater than i will be able to decrypt no matter what his column is, because the pairing will "cancel out" the randomized ($\mathbb{G}_p$) part of any column ciphertext component with the row ciphertext component that lives in $\mathbb{G}_q$.

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. 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:

$$\left[\begin{array}{c} g, h, E = g_q^{\beta}, E_1 = g_q^{\beta r_1}, \ldots, E_m = g_q^{\beta r_m}, F_1 = h_q^{\beta r_1}, \ldots, F_m = h_q^{\beta r_m}, \\ G_1 = e(g_q, g_q)^{\beta \alpha_1}, \ldots, G_m = e(g_q, g_q)^{\beta \alpha_m}, H_1 = g^{c_1}, \ldots, H_m = g^{c_m} \end{array}\right]$$

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 K includes factors p, q along with exponents used to generate the public key.

TrEncrypt$_{\text{LBE}}(\text{K}, M, (i,j))$ The *TrEncrypt*$_{\text{LBE}}$ 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 $t \in \mathbb{Z}_n$, $w_1, \ldots, w_m$, $s_1, \ldots, s_m \in Z_n$, $z_{p,1}, \ldots, z_{p,j-1} \in \mathbb{Z}_p$, 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:

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

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

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

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, while a column $y < j$ will be well formed in the $\mathbb{G}_q$ subgroup, but not in the $\mathbb{G}_p$ subgroup.

Encrypt$_{\text{LBE}}(\text{PK}, M)$ The *Encrypt*$_{\text{LBE}}$ 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 *Encrypt*$_{\text{LBE}}$ algorithm should produce ciphertexts that are indistinguishable from *TrEncrypt*$_{\text{LBE}}$ 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:

$$R_x = E_x^{s_x} \qquad \tilde{R}_x = F_x^{s_x} \qquad A_x = E^{s_x t} \qquad B_x = M G_x^{s_x t}$$

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

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

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

$$B_x \cdot \Big(e(K_{x,y}, A_x) e(\tilde{R}_x, \tilde{C}_y) / e(R_x, C_y) \Big)^{-1}.$$

We observe that if the ciphertext was created from the tracing algorithm *TrEncrypt*$_{\text{LBE}}$with parameters (i,j) then the result is M if $x > i$ or $x = i$ and $y \geq j$. Additionally, it is easy to observe that if the ciphertext was created as *Encrypt*$_{\text{LBE}}(\text{PK}, M)$ then 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 Encapsulation 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 is used to encrypt K to for all users in row x. In practice this would be more space efficient than including $\sqrt{N}$ group elements of $\mathbb{G}_T$.

The *Encrypt*$_{\text{LBE}}$ 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 just by testing which 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

[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.

(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 Decision 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Subgroup Decision assumptions hold. Then no polynomial time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with non-negligible advantage.*

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 Decision 3-party Diffie-Hellman, assumption holds. Then no polynomial time adversary can distinguish between an encryption to (i, j) and $(i, j+1)$ in the Index Hiding game for $j < m$ with non-negligible advantage.*

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 Decision 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Subgroup Decision assumptions hold. Then no polynomial time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with non-negligible advantage.*

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 Decision 3-party Diffie-Hellman assumption holds. Then no polynomial time adversary can distinguish between experiments H_1 and H_2 with non-negligible advantage.*

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 game 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 Decision 3-party Diffie-Hellman and the Bilinear Subgroup Decision assumptions hold. Then no polynomial time adversary can distinguish between experiments H_2 and H_3 with non-negligible advantage.*

We prove this claim in Appendix C.2.

Claim 5.6. *Suppose that the Decision 3-party Diffie-Hellman assumption holds. Then no polynomial time adversary can distinguish between experiments H_3 and H_4 with non-negligible advantage.*

We prove this claim in Appendix C.3.

Claim 5.7. *Suppose that the Subgroup Decision assumption holds. Then no polynomial time adversary can distinguish between experiments H_4 and H_5 with non-negligible advantage.*

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 distinguishing between $\mathit{Encrypt}_{\text{LBE}}(\text{PK}, M)$ and $\mathit{TrEncrypt}_{\text{LBE}}(\text{K}, M, (1,1))$.

Theorem 5.8. *Suppose the Subgroup Decision assumption holds. Then for all messages M no polynomial time adversary can distinguish between a ciphertext created as $\mathit{Encrypt}_{\text{LBE}}(\text{PK}, M)$ and one created as $\mathit{TrEncrypt}_{\text{LBE}}(\text{K}, M, (1,1))$ with non-negligible advantage.*

This theorem follows by simplifying 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 message, 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.

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 agents and his 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 [17] 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. 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 decoder 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.

Acknowledgments

We would like to thank Steven Galbraith for helpful comments and suggestions.

References

[1] Adam Barth, Dan Boneh, and Brent Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography '06*, 2006.

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

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

[4] 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.

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

[6] 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.

[7] Dan Boneh and Brent Waters. A fully collusion resistant broadcast trace and revoke system with public traceability. `http://crypto.stanford.edu/dabo/pubs.html`, 2006.

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

[9] 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.

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

[11] 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.

[12] A. Fiat and M. Naor. Broadcast encryption. In *Proceedings of Crypto '93*, volume 773 of *LNCS*, pages 480–491. Springer-Verlag, 1993.

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

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.

[19] 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.

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

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

[22] 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.

[23] 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.

[24] 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.

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

[26] 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.

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

[28] 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.

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

[30] 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.

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

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

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

[34] 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

Initially, we view a pirate decoder $\mathcal{D}$ as a probabilistic circuit that takes as input a ciphertext C and outputs some message M or $\perp$. A Traitor-Tracing system, then, consists of the following four algorithms:

Setup(N, λ) The setup algorithm takes as input N, the number of users in the system, and the security parameter λ. The algorithm runs in polynomial time in λ and outputs a public key BK, a secret tracing key TK, and private keys $\mathrm{K}_1, \ldots, \mathrm{K}_N$, where K_u is given to user u.

Encrypt(BK, M) Encrypts M using the public broadcasting key BK and outputs ciphertext C.

Decrypt$(j, \mathrm{K}_j, C, \mathrm{BK})$ Decrypt C using the private key K_j of user j. The algorithm outputs a message M or $\perp$.

Trace$^{\mathcal{D}}(\mathrm{TK},\ \epsilon)$ The tracing algorithm is an oracle algorithm that is given as input the tracing key TK and a parameter ϵ, and runs in time polynomial in the security parameter λ and $1/\epsilon$. Only values of ϵ that are polynomially related to λ are considered valid inputs to *Trace*. The tracing algorithm queries the pirate decoder $\mathcal{D}$ as a black-box oracle, as defined above. It outputs a set S which is a subset of $\{1, 2, \ldots, N\}$.

The system must satisfy the following **correctness property**:
for all $j \in \{1, \ldots, N\}$ and all messages M:

Let $\big(\mathrm{BK}, \mathrm{TK}, (\mathrm{K}_1, \ldots, \mathrm{K}_N)\big) \xleftarrow{\mathrm{R}} \mathit{Setup}(N, \lambda)$ and $C \xleftarrow{\mathrm{R}} \mathit{Encrypt}(\mathrm{BK}, M)$.

Then $\mathit{Decrypt}(j, \mathrm{K}_j, C, \mathrm{BK}) = M$.

Security. We define security of the traitor tracing scheme in terms of the following two natural game.

Game 1. The first game is the standard **Semantic Security Game**. It says that the system is semantically secure to an outsider who does not possess any of the private keys.

- **Setup** The challenger runs the *Setup* algorithm and gives the adversary BK.
- **Challenge** The adversary outputs two equal length messages M_0, M_1. The challenger flips a coin $\beta \in \{0, 1\}$ and sets $C \xleftarrow{\mathrm{R}} \mathit{Encrypt}(\mathrm{BK}, M_\beta)$. The challenger gives C to the adversary.
- **Guess** The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

We define the advantage of adversary $\mathcal{A}$ in winning this game as $\mathsf{Adv}_{\mathrm{SS}} = |\Pr[\beta' = \beta] - 1/2|$.

Game 2. The second game captures the notion of **Traceability against arbitrary collusion**. For a given N, λ and ϵ (where $\epsilon = 1/f(\lambda)$ for some polynomial f), the game proceeds as follows (both challenger and adversary are given N, λ, and ϵ as input):

1. The adversary $\mathcal{A}$ outputs a set $T = \{u_1, u_2, \ldots, u_t\} \subseteq \{1, \ldots, N\}$ of colluding users.
2. The challenger runs $\mathit{Setup}(N, \lambda)$ and provides BK and $\mathrm{K}_{u_1}, \ldots, \mathrm{K}_{u_t}$ to $\mathcal{A}$. It keeps TK to itself.
3. The adversary $\mathcal{A}$ outputs a pirate decoder $\mathcal{D}$.
4. The challenger now runs $\mathit{Trace}^{\mathcal{D}}(\mathrm{TK}, \epsilon)$ to obtain a set $S \subseteq \{1, \ldots, N\}$. Note that *Trace* is only given black-box oracle access to $\mathcal{D}$.

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

- The decoder $\mathcal{D}$ is ϵ-useful. That is, for a randomly chosen M in the finite message space, 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 denote by $\mathsf{Adv}_{\mathrm{TR}}$ the probability that adversary $\mathcal{A}$ wins this game.

Definition A.1. *We say that an N-user Traitor Tracing system is secure if for all polynomial time adversaries $\mathcal{A}$ and any constant $\epsilon > 0$ we have that Adv_{SS} and Adv_{TR} are negligible functions of λ.*

We emphasize that Game 2 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. Non stateless decoders were studied in [17].

In Game 2 above the adversary requests secret keys non-adaptively. The game can be enhanced to allow the adversary to request secret keys adaptively. Our system remains secure in this stronger model. More precisely, the reduction from traitor tracing to PLBE holds even when the adversary asks for secret keys adaptively before outputting the pirate decoder.

Minimal access decoders. 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 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. To address this issue we define a 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 Game 2 above so that challenger runs $Trace^{\mathcal{P}}(\mathrm{TK},\epsilon)$ 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 argue [3] that this model accurately captures the problem of tracing a stateless tamper resistant decoder. It is not difficult to see that the tracing algorithm derived from a secure PLBE enables black-box tracing in both the full access model and the minimal access model.

B Reducing Traitor Tracing to PLBE

We prove Lemma 2.2. We first give some intuition. Our goal is to show that $Encrypt_{\text{LBE}}(\mathrm{PK},M)$ is semantically secure against an adversary who has PK but does not have any of the private keys $\mathrm{K}_1,\ldots,\mathrm{K}_N$. In other words, we want to show that $Encrypt_{\text{LBE}}(\mathrm{PK},M_0)$ is indistinguishable from $Encrypt_{\text{LBE}}(\mathrm{PK},M_1)$ where M_0, M_1 are chosen by the adversary. Informally, we do so using a hybrid argument in several steps:

1. $Encrypt_{\text{LBE}}(\mathrm{PK},M_0)$ is indistinguishable from $TrEncrypt_{\text{LBE}}(\mathrm{TK},1,M_0)$ by Game 1 of PLBE security.
2. We know that $TrEncrypt_{\text{LBE}}(\mathrm{TK},i,M_0)$ is indistinguishable from $TrEncrypt_{\text{LBE}}(\mathrm{TK},i{+}1,M_0)$ for $i = 1,\ldots,N$ by Game 3 of PLBE security.
3. $TrEncrypt_{\text{LBE}}(\mathrm{TK},N+1,M_0)$ is indistinguishable from $TrEncrypt_{\text{LBE}}(\mathrm{TK},N+1,M_1)$ by Game 2 of PLBE security.
4. Again, for $i = N+1,\ldots,2$ we know that $TrEncrypt_{\text{LBE}}(\mathrm{TK},i,M_1)$ is indistinguishable from $TrEncrypt_{\text{LBE}}(\mathrm{TK},i-1,M_1)$ by Game 3 of PLBE security.
5. Finally, $TrEncrypt_{\text{LBE}}(\mathrm{TK},1,M_1)$ is indistinguishable from $Encrypt_{\text{LBE}}(\mathrm{PK},M_1)$ by Game 1 of PLBE security.

Overall, this shows that $Encrypt_{\text{LBE}}(\mathrm{PK},M_0)$ is indistinguishable from $Encrypt_{\text{LBE}}(\mathrm{PK},M_1)$ as required. We now give the full proof.

Proof. To make the argument precise we recast the PLBE security games of Section 2 in terms of the following experiments:

- Indistinguishability experiment (PLBE Game 1), denoted $\mathsf{EXP}^{(b)}_{\text{IND}}$ for $b \in \{0,1\}$. The challenger runs the $\mathit{Setup}_{\text{LBE}}$ algorithm and gives the adversary PK and $\{\text{K}_1, \ldots, \text{K}_N\}$. The adversary $\mathcal{A}$ outputs a message M. If $b = 1$ the challenger returns $C \stackrel{\text{R}}{\leftarrow} \mathit{Encrypt}_{\text{LBE}}(\text{PK}, M)$; otherwise the challenger returns $C \stackrel{\text{R}}{\leftarrow} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, 1, M)$. The adversary outputs a guess $b' \in \{0,1\}$. We let $\mathsf{EXP}^{(b)}_{\text{IND}}[\mathcal{A}]$ denote the probability that $b' = 1$.

- Index hiding experiment (PLBE Game 3), denoted $\mathsf{EXP}^{(b)}_{\text{IH}}[i]$ for $b \in \{0,1\}$ and $0 \le i \le N+1$. The challenger runs the $\mathit{Setup}_{\text{LBE}}$ algorithm and gives the adversary i, PK and $\{\text{K}_j \; , \; j \neq i\}$. The adversary $\mathcal{A}$ outputs a message M. The challenger returns $C \stackrel{\text{R}}{\leftarrow} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, i + b, M)$. The adversary outputs a guess $b' \in \{0,1\}$. We let $\mathsf{EXP}^{(b)}_{\text{IH}}[\mathcal{A}, i]$ denote the probability that $b' = 1$.

- Message hiding experiment (PLBE Game 2), denoted $\mathsf{EXP}^{(b)}_{\text{MH}}$ for $b \in \{0,1\}$. The challenger runs the $\mathit{Setup}_{\text{LBE}}$ algorithm and gives the adversary PK and $\{\text{K}_1, \ldots, \text{K}_N\}$. The adversary $\mathcal{A}$ outputs two equal length messages M_0, M_1. The challenger returns $C \stackrel{\text{R}}{\leftarrow} \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, N + 1, M)$. The adversary outputs a guess $b' \in \{0,1\}$. We let $\mathsf{EXP}^{(b)}_{\text{MH}}[\mathcal{A}]$ denote the probability that $b' = 1$.

- Semantic security experiment, denoted $\mathsf{EXP}^{(b)}_{\text{SS}}$ for $b \in \{0,1\}$. The challenger runs the $\mathit{Setup}_{\text{LBE}}$ algorithm and gives the adversary PK. The adversary $\mathcal{A}$ outputs two equal length messages M_0, M_1. The challenger returns $C \stackrel{\text{R}}{\leftarrow} \mathit{Encrypt}_{\text{LBE}}(\text{PK}, M_b)$. The adversary outputs a guess $b' \in \{0,1\}$. We let $\mathsf{EXP}^{(b)}_{\text{SS}}[\mathcal{A}]$ denote the probability that $b' = 1$.

Since the PLBE is secure we know that for all polynomial time adversaries $\mathcal{A}$ and all $i = 1, \ldots, N$ the quantities

$$
\begin{aligned}
2 \cdot \mathsf{Adv}_{\text{IND}} &= \left| \mathsf{EXP}^{(0)}_{\text{IND}}[\mathcal{A}] - \mathsf{EXP}^{(1)}_{\text{IND}}[\mathcal{A}] \right| \;, \\
2 \cdot \mathsf{Adv}_{\text{IH}}[i] &= \left| \mathsf{EXP}^{(0)}_{\text{IH}}[\mathcal{A}, i] - \mathsf{EXP}^{(1)}_{\text{IH}}[\mathcal{A}, i] \right| \;, \\
2 \cdot \mathsf{Adv}_{\text{MH}} &= \left| \mathsf{EXP}^{(0)}_{\text{MH}}[\mathcal{A}] - \mathsf{EXP}^{(1)}_{\text{MH}}[\mathcal{A}] \right|
\end{aligned}
$$

are negligible. Our goal is to show that $|\mathsf{EXP}^{(0)}_{\text{SS}}[\mathcal{A}] - \mathsf{EXP}^{(1)}_{\text{SS}}[\mathcal{A}]|$ is negligible for any polynomial time adversary $\mathcal{A}$. Given a semantic security adversary $\mathcal{A}$ we define three additional adversaries $\mathcal{B}, \mathcal{B}_0, \mathcal{B}_1$ as follows:

- Adversary $\mathcal{B}$ receives PK and $\{\text{K}_1, \ldots, \text{K}_N\}$ from its challenger. It then runs $\mathcal{A}$ giving it PK and obtaining M_0, M_1 in return. It passes these to the challenger and obtains a challenge ciphertext. It gives the challenge to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

- For $\beta = 0, 1$ adversary $\mathcal{B}_\beta$ works as follows. It receives PK and some subset of private keys. It runs $\mathcal{A}$ giving it PK and obtaining M_0, M_1 in return. It passes M_β to the challenger and obtains a challenge ciphertext. It gives the challenge to $\mathcal{A}$ and outputs whatever $\mathcal{A}$ outputs.

Note that $\mathcal{B}$ is a valid adversary for the message hiding experiment. Similarly, $\mathcal{B}_0$ and $\mathcal{B}_1$ are adversaries for the index hiding experiment and indistinguishability experiment. Now, by the triangle inequality we have:

$$\left|\mathsf{EXP}^{(0)}_{\mathrm{SS}}[\mathcal{A}] - \mathsf{EXP}^{(1)}_{\mathrm{SS}}[\mathcal{A}]\right| \leq$$

$$\left|\mathsf{EXP}^{(0)}_{\mathrm{SS}}[\mathcal{A}] - \mathsf{EXP}^{(0)}_{\mathrm{IND}}[\mathcal{B}_0]\right| \tag{1}$$

$$+ \quad \left|\mathsf{EXP}^{(0)}_{\mathrm{IND}}[\mathcal{B}_0] - \mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_0, 1]\right| \tag{2}$$

$$+ \quad \sum_{i=1}^{N} \left|\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_0, i] - \mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_0, i+1]\right| \tag{3}$$

$$+ \quad \left|\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_0, N+1] - \mathsf{EXP}^{(0)}_{\mathrm{MH}}[\mathcal{B}]\right| \tag{4}$$

$$+ \quad \left|\mathsf{EXP}^{(0)}_{\mathrm{MH}}[\mathcal{B}] - \mathsf{EXP}^{(1)}_{\mathrm{MH}}[\mathcal{B}]\right| \tag{5}$$

$$+ \quad \left|\mathsf{EXP}^{(1)}_{\mathrm{MH}}[\mathcal{B}] - \mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_1, N+1]\right| \tag{6}$$

$$+ \quad \sum_{i=1}^{N} \left|\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_1, i+1] - \mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_1, i]\right| \tag{7}$$

$$+ \quad \left|\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_1, 1] - \mathsf{EXP}^{(0)}_{\mathrm{IND}}[\mathcal{B}_1]\right| \tag{8}$$

$$+ \quad \left|\mathsf{EXP}^{(0)}_{\mathrm{IND}}[\mathcal{B}_1] - \mathsf{EXP}^{(1)}_{\mathrm{SS}}[\mathcal{A}]\right| \tag{9}$$

We argue that all the terms (1)-(9) are negligible, which will prove the lemma. We first state a few simple equalities. For $\gamma = 0, 1$ we have:

$$\mathsf{EXP}^{(\gamma)}_{\mathrm{SS}}[\mathcal{A}] = \mathsf{EXP}^{(1)}_{\mathrm{IND}}[\mathcal{B}_\gamma] \tag{10}$$

$$\mathsf{EXP}^{(0)}_{\mathrm{IND}}[\mathcal{B}_\gamma] = \mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_\gamma, 1] \tag{11}$$

$$\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_\gamma, i+1] = \mathsf{EXP}^{(1)}_{\mathrm{IH}}[\mathcal{B}_\gamma, i] \quad \text{for } i = 0, \ldots, N \tag{12}$$

$$\mathsf{EXP}^{(0)}_{\mathrm{IH}}[\mathcal{B}_\gamma, N+1] = \mathsf{EXP}^{(0)}_{\mathrm{MH}}[\mathcal{B}] \tag{13}$$

We now obtain the following:

- The terms (1) and (9) are negligible by equality (10) and the fact that $\mathsf{Adv}_{\mathrm{IND}}$ is negligible.
- The terms (2) and (8) are zero by equality (11).
- Each of the terms in (3) and (7) are negligible by equality (12) and the fact that $\mathsf{Adv}_{\mathrm{IH}}[i]$ is negligible.
- The terms (4) and (6) are zero by equality (13).
- The term (5) is negligible since $\mathsf{Adv}_{\mathrm{MH}}$ is negligible.

This completes the proof of the lemma. □

B.1 Security of Tracing

We prove Lemma 2.3. We sketched the proof in Section 2.2. Here we give more details.

Proof. We show that the probability of winning the traceability game defined in Appendix A is negligible. For a pirate decoder $\mathcal{D}$ and $i = 1, \dots, N+1$ define:

$$p_i = \Pr[\mathcal{D}(\mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, i, M)) = M]$$
$$p = \Pr[\mathcal{D}(\mathit{Encrypt}_{\text{LBE}}(\text{PK}, M)) = M]$$

where M is sampled at random from the finite message space. Let ϵ be some fixed constant. We distinguish between three types of pirates:

- Type 1: A pirate that produces an ϵ-useful pirate decoder $\mathcal{D}$ for which $|p-p_1|$ is non-neligible.
- Type 2: A pirate that produces an ϵ-useful pirate decoder $\mathcal{D}$ for which $|p-p_1|$ is neligible, but the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\text{TK},\ \epsilon)$ outputs an empty set.
- Type 3: A pirate that produces an ϵ-useful pirate decoder $\mathcal{D}$ for which $|p-p_1|$ is neligible, but the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\text{TK},\ \epsilon)$ outputs a set that is not contained in the set of colluders T.

It is straight forward to show that any pirate that produces a Type 1 decoder with non-negligible probability can be used to win the PLBE indistiguishability game. Similarly, any pirate that produces a Type 2 decoder with non-negligible probability can be used to win the PLBE message hiding game.

We briefly show that a pirate that produces a Type 3 decoder with non-negligible probability can be used to win the PLBE index hiding game. Recall that the tracing algorithm defined in Section 2.2 computes an approximataion $\hat{p}_i$ of p_i using $W = 8\lambda(N/\epsilon)^2$ samples. The Chernoff bound then implies that for all $i = 1, \dots, N$ we have:

$$\Pr[|p_i - \hat{p}_i| > \epsilon/(16N)] < 2e^{-\lambda/64}$$

which is negligible in the security parameter λ. Therefore, we may assume from here on that $|p_i - \hat{p}_i| \le \epsilon/(16N)$ for all $i = 1, \dots, N$.

Now, suppose a pirate P produces a Type 3 decoder with probability δ. Then there must be some user $u \in \{1, \dots, N\}$ that is falsely accused with probability at least δ/N. We build an adversary $\mathcal{A}$ for which $\mathsf{Adv}_{\text{IH}}[u]$ is non-negligible. This adversary works as follows:

1. $\mathcal{A}$ obtains from its challenger PK and keys $\{\text{K}_i\,,\, i \neq u\}$.
2. $\mathcal{A}$ runs P and obtains a set $T = \{u_1, u_2, \dots, u_t\} \subseteq \{1, \dots, N\}$ of colluding users. If $u \in T$ then $\mathcal{A}$ outputs `fail` and aborts.
3. Otherwise, $\mathcal{A}$ gives P the public key PK and the secret keys $\{\text{K}_{u_1}, \text{K}_{u_2}, \dots, \text{K}_{u_t}\}$. The pirate P outputs a useful pirate decoder $\mathcal{D}$.
4. $\mathcal{A}$ picks a random message M in the finite message space and gives it to its challenger. The challenger responds with $C \leftarrow \mathit{TrEncrypt}_{\text{LBE}}(\text{TK}, i+\beta, M)$ for $\beta \xleftarrow{\text{R}} \{0,1\}$.

5. If $\mathcal{D}(C) = M$ then $\mathcal{A}$ outputs $\beta' = 0$; otherwise it outputs $\beta' = 1$.

Let $\mathcal{E}$ be the event that P outputs a Type 3 decoder that falsely accuses user u. We know $\mathcal{E} \geq \delta/N$. When event $\mathcal{E}$ happens then $\hat{p}_u - \hat{p}_{u+1} > \epsilon/(4N)$ and therefore $p_u - p_{u+1} > \epsilon/(8N)$ with overwhelming probability. We also know that for $b = 0, 1$ we have $p_{u+b} = \Pr[\beta' = 0 | \beta = b]$. Hence, when event $\mathcal{E}$ happens

$$|\Pr[\beta = \beta'] - 1/2| > \epsilon/(4N)$$

Since $\mathcal{E}$ happens with non-negligible probabilty we see that $\beta = \beta'$ happens with non-negligible advantage. This contradicts the fact that $\mathsf{Adv}_{\text{IH}}[u]$ is negligible.

□

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 know the factorization of n, the order of the group. For this game simulator will run the core part of the simulation in the $\mathbb{G}_p$ subgroup and choose all values in the $\mathbb{G}_q$ 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 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 receives 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 $\beta, r_{q,1}, \ldots, r_{q,m}, c_{q,1}, \ldots, c_{q,m} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{p,1}, \ldots, r'_{p,m}, c'_{p,1}, \ldots, c'_{p,m} \in \mathbb{Z}_p$.

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:

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

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

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

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 $s_{q,i}, \ldots, s_{q,m} \in \mathbb{Z}_q$ and $t_q \in \mathbb{Z}_q$. Additionally, it chooses random $s'_p \in \mathbb{Z}_p$, $z_{p,1}, \ldots, z_{p,j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$.

It then creates the ciphertext as:

$$\begin{array}{llll}
\text{if } x > i: & R_x = g_q^{s_{q,x} r_{q,x}} & \tilde{R}_x = h_q^{s_{q,x} r_{q,x}} \\
& A_x = g_q^{s_{q,x} t_q} & B_x = M e(g_q, g_q)^{\alpha_x s_{q,x} t_q} \\
\text{if } x = i: & R_x = g_q^{s_{q,x} r_{q,x}} g_p^{s'_p r'_{p,x}} & \tilde{R}_x = h_q^{s_{q,x} r_{q,x}} B^{s'_p r'_{p,x}} \\
& A_x = g^{s_{q,x} t_q} A^{s'_p} & B_x = M e(g_q, g_q)^{\alpha_x s_{q,x} t_{q,x}} e(g_p, A)^{\alpha_x s'_p} \\
\text{if } x < i: & R_x = g^{v_{x,1}} & \tilde{R}_x = h^{v_{x,1}} \\
& A_x = g^{v_{x,2}} & B_x = e(g, g)^{v_{x,3}} \\
\text{if } y > j: & C_y = g_q^{c_{q,y} t_q} h^{w'_y} & \tilde{C}_y = A^{-c'_{p,y}} g^{w'_y} \\
\text{if } y = j: & C_y = g_q^{c_{q,y} t_q} T h^{w'_y} & \tilde{C}_y = g^{w'_y} \\
\text{if } y < j: & C_y = g_q^{c_{q,y} t_q} g_p^{z_{p,y}} h^{w'_y} & \tilde{C}_y = g^{w'_y}
\end{array}$$

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. It follows from our assumption that the adversary has a negligible advantage. □

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 by a random element $e(g, g)^z$.

Distinguishing between H_2 and H_{2a} We first show that if the Decision 3-party Diffie-Hellman assumption holds then no polynomial time adversary can distinguish between experiments H_2 and H_{2a}. We note that if no polynomial time adversary can break the decision 3-party Diffie-Hellman with non-negligible advantage then no polynomial time adversary has non-negligible advantage 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 DBDH 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 $g_p^{c_{p,i}} = A^{-1} g^{c'_{p,i}}$ where $c'_{p,1}, \ldots, c'_{p,m}$ 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 $g = g_p g_q, h = h_p h_q$. Additionally, it chooses $\beta, c_{q,1}, \ldots, c_{q,m}, r_{q,i} \in \mathbb{Z}_q, r_1, \ldots r_{i-1}, , r_{i+1}, \ldots, r_m \in \mathbb{Z}_n$, $\alpha_{q,i} \in \mathbb{Z}_q$, and $\alpha_1, \ldots \alpha_{i-1}, , \alpha_{i+1}, \ldots, \alpha_m \in \mathbb{Z}_n$.

The public parameters are published as:

$$g, h, E = g_q^{\beta}, E_1 = g_q^{\beta r_1}, \ldots, E_i = g_q^{\beta r_{q,i}}, \ldots, E_m = g_q^{\beta r_m},$$
$$F_1 = g_q^{d\beta r_1}, \ldots, F_i = g_q^{d\beta r_{q,i}}, \ldots, F_m = g_q^{d\beta r_m},$$
$$G_1 = e(g_q, g)^{\beta\alpha_1}, \ldots, G_i = e(g_q, g)^{\beta\alpha_{q,i}}, \ldots, G_m = e(g_q, g)^{\beta\alpha_m},$$
$$H_1 = g^{c_q} A^{-1} g^{c'_{p,1}}, \ldots, H_m = g^{c_q} A^{-1} g^{c'_{p,m}}.$$

The keys are created as:

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

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 randomly 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:

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

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

$$g, h, E = k_q^{\beta}, E_x = \begin{cases} k_q^{\beta r'_{q,x}} : x = i \\ A^{\beta r'_{q,x}} : x \neq i \end{cases} \qquad F_x = \begin{cases} k_q^{d\beta r'_{q,x}} : x = i \\ A^{d\beta r'_{q,x}} : x \neq i \end{cases}$$

$$G_1 = e(A,A)^{\beta\alpha_1}, \ldots, G_m = e(A,A)^{\beta\alpha_m}, H_1 = k_q^{c'_{q,1}} g_p^{c_{p,1}}, \ldots, H_m = k_q^{c'_{q,x}} g_p^{c_{p,m}}.$$

Keys are generated as

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

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

It creates the challenge ciphertext as:

$$\begin{array}{lllll} \text{if } x > i: & R_x = k_q^{s'_{q,x}r_{q,x}} & \tilde{R}_x = k_q^{ds'_{q,x}r_{q,x}} & A_x = B^{s'_{q,x}} & B_x = Me(A,B)^{\alpha_x s'_{q,x}} \\ \text{if } x = i: & R_x = C^{s'_{q,x}} g_p^{s_p r_{q,x}} & \tilde{R}_x = C^{ds'_{q,x}} g_p^{ds_p r_{q,x}} & A_x = T g_p^{\delta} & B_x = e(g,g)^{\gamma} \\ \text{if } x < i: & R_x = g^{v_{x,1}} & \tilde{R}_x = h^{v_{x,1}} & A_x = g^{v_{x,2}} & B_x = e(g,g)^{v_{x,3}} \\ \forall y & C_y = B^{c'_{q,y}} g_p^{z'_y} h^{w'_y} & \tilde{C}_y = g^{w'_y} & & \end{array}$$

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 From our assumptions and the reductions above we can now bound the Adversary's advantage to be negligible. □

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

$\beta, r_{q,1}, \ldots, r_{q,m}, c_{q,1}, \ldots, c_{q,m} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{p,1}, \ldots, r'_{p,m}, c'_{p,1}, \ldots, c'_{p,m} \in \mathbb{Z}_p$.

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:

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

Next, it creates the private keys as:

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

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 $s_{q,i+1}, \ldots, s_{q,m} \in \mathbb{Z}_q$ and $t_q \in \mathbb{Z}_q$. Next, it chooses random $z_{p,1}, \ldots, z_{p,j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$.

It then creates the ciphertext as:

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

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 negligible. □

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:

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

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

□