A Fully Collusion Resistant Broadcast, Trace and Revoke System

Dan Boneh Brent Waters

Abstract

We introduce a simple primitive called *Augmented Broadcast Encryption* (ABE) that is sufficient for constructing broadcast encryption, traitor-tracing, and trace-and-revoke systems. These ABE-based constructions are resistant to an arbitrary number of colluders and are secure against *adaptive adversaries*. Furthermore, traitor tracing requires no secrets and can be done by anyone. These broadcast systems are designed for broadcasting to arbitrary sets of users. We then construct a secure ABE system for which the resulting concrete trace-and-revoke system has ciphertexts and private keys of size $\sqrt{N}$ where N is the total number of users in the system. In particular, this is the first example of a fully collusion resistant broadcast system with sub-linear size ciphertexts and private keys that is secure against adaptive adversaries. The system is publicly traceable.

1 Introduction

A **broadcast encryption** system [15] enables a broadcaster to encrypt a message for an arbitrary subset $S \subseteq \{1, \ldots, N\}$ of users who are listening on a broadcast channel. Any user in S can decrypt the broadcast using his private key. Moreover, even if *all* users outside of S collude they obtain no information about the contents of the broadcast. Such systems are said to be collusion resistant. **Traitor tracing** [10] is an orthogonal problem. Here a broadcaster encrypts messages so that *all* N users can decrypt the resulting ciphertexts. Suppose a coalition of users $T \subseteq \{1, \ldots, N\}$ get together and build a pirate decoder $\mathcal{D}$. Then there is a tracing algorithm *Trace* that takes the public key PK as input and interacts with $\mathcal{D}$ as a black-box oracle. The algorithm outputs the identity of at least one of the users who created $\mathcal{D}$. That is, $\emptyset \neq \mathit{Trace}^{\mathcal{D}}(\mathrm{PK}) \subseteq T$. Note, however, that there is no way to revoke the traitor — broadcasts can always be decrypted by all users. The tracing algorithm, as described above, needs no secrets and can be run by anyone. Such traitor tracing systems are said to be publicly traceable.

Trace and Revoke [25, 24] systems provide both broadcast encryption and traitor tracing. They are motivated by content protection on various platforms such as PCs, DVD players, and general content viewers. When the system is first rolled out, broadcasts are encrypted for some subset of users $S \subseteq \{1, \ldots, N\}$ authorized to receive them. The goal is to then revoke users when their keys are compromised. Suppose a pirate builds a pirate decoder $\mathcal{D}$ using the private keys of users $T \subseteq \{1, \ldots, N\}$. The tracing algorithm then interacts with $\mathcal{D}$ and identifies one of the active keys in the pirate's possession, namely a key of user $t \in T \cap S$. We write $\mathit{Trace}^{\mathcal{D}}(\mathrm{PK}, S) \subseteq T \cap S$. The broadcaster revokes user t by encrypting future broadcasts to the set $S' \leftarrow S \setminus \{t\}$. If the pirate decoder $\mathcal{D}$ can still decrypt these broadcasts, we run the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(\mathrm{PK}, S')$ again and obtain another pirate key $t' \in T \cap S'$. Again, t' is revoked by setting $S'' \leftarrow S' \setminus \{t'\}$ and so on. Roughly speaking, the trace and revoke system is secure if this process eventually disables

$\mathcal{D}$ without revoking any innocent party. We give precise definitions later in the paper. Note that the broadcaster can add or remove recipients from S at will.

Our Contribution In this paper we focus on constructing public-key trace and revoke systems that are fully collusion resistant and have short ciphertexts and private keys. The system is publicly traceable in the sense that anyone can run the tracing algorithm — no additional secrets are needed. For message privacy we only consider chosen plaintext attacks. Rather than directly build a trace and revoke system, we instead construct a simpler primitive we call **Augmented Broadcast Encryption** or ABE for short. We then show that ABE implies a trace and revoke system.

An ABE contains the same algorithms as a public-key broadcast encryption system, namely $(\mathit{Setup}_{\mathrm{ABE}}, \mathit{Encrypt}_{\mathrm{ABE}}, \mathit{Decrypt}_{\mathrm{ABE}})$. The encryption algorithm $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i, M)$, however, takes one additional parameter i. Here PK is the public key, M is a message, S is a subset of $\{1, \ldots, N\}$, and i is an additional special input $1 \leq i \leq N+1$. The encryption algorithm outputs a ciphertext that can be decrypted by any user in $S \cap \{i, \ldots, N\}$. We only require that

- The output of $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, N+1, M)$ contains no information about M.
- For $i \in S$ the distribution generated by $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i, M)$ is indistinguishable from the distribution generated by $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i+1, M)$ for any attacker that does not possess the secret key of user i. When $i \notin S$ the two distributions are indistinguishable to anyone.

We give precise definitions in the next section. We show that an ABE system directly gives a secure and fully collusion resistant broadcast encryption system. To encrypt message M to set S we run $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, 1, M)$, namely setting $i = 1$. Values of i greater than 1 are only used in the proof of security and for tracing. The resulting broadcast encryption system is secure against *adaptive adversaries* — adversaries that choose adaptively the subset of users to attack. We then show that this broadcast system is publicly traceable (and hence is a trace and revoke system). The tracing algorithm is based on a standard tracing technique that was previously used in [3, 24, 21] and was recently made explicit in [6]. The tracing system of [6], however, required a secret tracing key held by a trusted party. Here, tracing requires no secrets so that there is no need for a trusted party.

In summary, the two simple ABE security properties are sufficient for obtaining a trace and revoke system that is fully collusion resistant, is secure against adaptive adversaries, and is publicly traceable. We view this as the preamble leading to our main results. The main part of the paper builds a secure ABE system where the size of private keys and ciphertexts is $\sqrt{N}$. We thus obtain a trace and revoke system with $\sqrt{N}$ size ciphertext and private keys that is fully collusion resistant and is secure against adaptive adversaries. Some previous fully collusion resistant broadcast systems [4, 7] for arbitrary sets were only secure against static adversaries and were not traceable.

1.1 Related work.

Broadcast encryption systems are often designed for the case when the pirate has fewer than some t private keys [15, 35, 36, 1, 37, 25, 13, 18]. Several elegant constructions [24, 12, 20, 19], primarily designed for broadcasting to sets where a small number of users are revoked, resist arbitrary collusion, but the size of the ciphertext grows linearly with the number of revoked users. A recent system based on pairings [4] resists arbitrary collusion and has constant size ciphertext and private keys, but does not support traitor tracing. The system is only proven secure for static adversaries, namely adversaries that commit to the set they wish to attack before seeing the public key. Broadcast encryption secure against adaptive attacks was defined in [13], but the resulting system had linear size ciphertexts when broadcasting to an arbitrary set S. The system in this paper provides adaptive security with sub-linear ciphertexts and private keys.

Similarly, traitor tracing systems are often designed for the case when the pirate has fewer than t private keys [10, 34, 32, 33, 23, 26, 3, 16, 11, 29, 2, 31, 30, 37, 21, 22]. A recent system based on pairings [6] resists arbitrary collusion and has constant size private keys and $\sqrt{N}$ size ciphertexts. That system is the basis of our tracing mechanism. Many tracing traitors systems, including [6], assume the tracer is a trusted party and require a secret tracing key. The system in this paper is publicly traceable meaning that tracing requires no secrets. Other publicly traceable systems are [27, 28, 38, 22, 9].

Several trace and revoke systems are available [25, 17, 24, 37, 13, 14, 20, 19] that are designed for broadcasting to large sets. In summary, the following sub-linear size fully collusion resistant systems are available:

	system type	ciphertext size	private key size	public key size	comment
[4]	Broadcast Encryption	$O(1)$	$O(1)$	$O(N)$	Static attacker
[6]	Traitor Tracing	$O(\sqrt{N})$	$O(1)$	$O(\sqrt{N})$	Private tracing
This paper	Trace and Revoke	$O(\sqrt{N})$	$O(\sqrt{N})$	$O(\sqrt{N})$	adaptive and public tracing

where N is the total number of users in the system. As usual, all these expressions are multiplied by the security parameter.

2 Augmented broadcast encryption

Our goal is to build a fully collusion resistant trace and revoke system secure against adaptive adversaries. In particular, this gives a broadcast encryption system secure against adaptive adversaries. However, instead of directly building a trace and revoke system we build a simpler primitive we call *Augmented Broadcast Encryption* or ABE for short. We begin by defining what ABE is and then explain how it gives a trace and revoke system. Then in the next section we build an efficient ABE.

2.1 Augmented Broadcast Encryption: Definitions

An ABE is a public-key broadcast system comprising of the following algorithms:

Setup$_{\mathbf{ABE}}(N, \lambda)$ A probabilistic algorithm that takes as input N, the number of users in the system, and a security parameter λ. The algorithm runs in polynomial time in λ and outputs a public key PK and private keys $\mathrm{SK}_1, \ldots, \mathrm{SK}_N$, where SK_u is given to user u.

Encrypt$_{\mathbf{ABE}}(S, \mathbf{PK}, i, M)$ Takes as input a subset $S \subseteq \{1, \ldots, N\}$, a public key PK, 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 $S \cap \{i, \ldots, N\}$.

Decrypt$_{\textbf{ABE}}(S, j, \mathbf{SK}_j, C, \mathbf{PK})$ Takes as input a subset $S \subseteq \{1, \ldots, N\}$, the private key SK_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 subsets $S \subseteq \{1, \ldots, N\}$, all $i, j \in \{1, \ldots, N+1\}$ (where $j \leq N$), and all messages M:

Let $(\mathrm{PK}, (\mathrm{SK}_1, \ldots, \mathrm{SK}_N)) \overset{R}{\leftarrow} \mathit{Setup}_{\mathrm{ABE}}(N, \lambda)$ and $C \overset{R}{\leftarrow} \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i, M)$.

If $\mathbf{j \in S}$ **and** $\mathbf{j \geq i}$ then $\mathit{Decrypt}_{\mathrm{ABE}}(S, j, \mathrm{SK}_j, C, \mathrm{PK}) = M$.

Security. We define security of an ABE system using two games. The first game is a **message hiding game** and says that a ciphertext created using index $i = N+1$ is unreadable by anyone. The second 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. The first game, called the **Message Hiding 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 $\mathit{Setup}_{\mathrm{ABE}}(N, \lambda)$ and gives the adversary PK and all secret keys $\{\mathrm{SK}_1, \ldots, \mathrm{SK}_N\}$.
- **Challenge** The adversary outputs a set $S \subseteq \{1, \ldots, N\}$ and two equal length messages M_0, M_1. The challenger flips a coin $\beta \in \{0,1\}$ and sends $C \overset{R}{\leftarrow} \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, N+1, M_\beta)$ 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 the game as $\mathsf{MH\,Adv}_{\mathcal{A}} = |\Pr[\beta' = \beta] - 1/2|$.

Game 2. The second game, called the **Index Hiding Game** says that an adversary cannot distinguish between an encryption to index i and one to index $i+1$ without the key SK_i. Additionally, it says that an adversary cannot distinguish between an encryption to index i and one to index $i+1$ when i is not in the target set S even with the key SK_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 $\mathit{Setup}_{\mathrm{ABE}}(N, \lambda)$ and gives the adversary PK and the set of private keys $\{\mathrm{SK}_j \text{ s.t. } j \neq i\}$.
- **Query** The adversary outputs a bit $\tilde{s} \in \{0,1\}$. If $\tilde{s} = 1$ the challenger sends SK_i to the adversary. Otherwise the challenger does nothing.
- **Challenge** The adversary gives the challenger a set $S \subseteq \{1, \ldots, N\}$ and a message M. The only restriction is that if $\tilde{s} = 1$ then $i \notin S$. The challenger flips a coin $\beta \in \{0,1\}$ and sends $C \overset{R}{\leftarrow} \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i+\beta, M)$ to the adversary.

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

We define the advantage of adversary $\mathcal{A}$ as the quantity $\mathsf{IH\,Adv}_{\mathcal{A}}[i] = |\Pr[\beta' = \beta] - 1/2|$. In words, the game captures two properties. The case $\tilde{s} = 0$ captures the fact that even if all users other than i collude they cannot distinguish whether i or $i+1$ was used to create a ciphertext C. The case $\tilde{s} = 1$ captures the fact that when $i \notin S$ then even if everyone colludes they cannot distinguish whether i or $i+1$ was used to create C. Indeed, when $i \notin S$ the key SK_i gives little additional information.

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

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

2.2 Using Augmented Broadcast Encryption

We first show that a secure ABE is a broadcast encryption system secure against adaptive attackers. We then show that this system is traceable, thus obtaining a trace and revoke system. From here on, whenever we refer to an adversary we mean an adversary whose running time is polynomial in the security parameter λ.

2.2.1 Broadcast encryption secure against adaptive attacks

Let $\mathcal{E} = (\mathit{Setup}_{\mathrm{ABE}}, \mathit{Encrypt}_{\mathrm{ABE}}, \mathit{Decrypt}_{\mathrm{ABE}})$ be a secure ABE system. Define

$$\mathit{Encrypt}(S, \mathrm{PK}, M) = \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, 1, M)$$

We show that $\mathcal{E}_{\mathrm{BE}} = (\mathit{Setup}_{\mathrm{ABE}}, \mathit{Encrypt}, \mathit{Decrypt}_{\mathrm{ABE}})$ is a fully collusion resistant broadcast encryption system secure against adaptive attackers.

First we need a slightly more elaborate message hiding game. In addition to N, λ, the game takes as input a parameter $i \in \{1, \ldots, N+1\}$ which is only given to the challenger. The game proceeds as follows:

- **Setup** The challenger runs $\mathit{Setup}_{\mathrm{ABE}}(N, \lambda)$ and gives the adversary PK.

- **Query** The adversary issues *adaptive* private key queries: it repeatedly sends values $j \in \{1, \ldots, N\}$ to the challenger and the challenger responds with SK_j. Let $S_0 \subseteq \{1, \ldots, N\}$ denote the entire set of private keys requested by the adversary during the query phase. Let $\overline{S_0} = \{1, \ldots, N\} \smallsetminus S_0$.

- **Challenge** The adversary outputs a set $S \subseteq \overline{S_0}$ and two equal length messages M_0, M_1. The challenger flips a coin $\beta \in \{0,1\}$ and send $C \overset{R}{\leftarrow} \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i, M_\beta)$ to the adversary. This is the only place where i is used in this game.

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

We define the advantage of $\mathcal{A}$ in winning the game as $\mathsf{MH\,Adv}_{\mathcal{A}}[i] = \big|\Pr[\beta' = \beta] - 1/2\big|$.

The main point is that $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ is the same quantity used to define broadcast encryption security against adaptive attackers [13, 4] for $\mathcal{E}_{\mathrm{B}E}$. Hence, if we prove that $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ is negligible then $\mathcal{E}_{\mathrm{B}E}$ is a broadcast system that is fully collusion resistant and secure against adaptive adversaries.

Theorem 2.2. *If $\mathcal{E}$ is a secure ABE then $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ is a negligible function of λ for any adversary $\mathcal{A}$.*

Proof Sketch. Suppose $\mathsf{MH\,Adv}_{\mathcal{A}}[1] > \epsilon$ for some adversary $\mathcal{A}$ and non-negligible ϵ. Since $\mathcal{E}$ is a secure ABE we know that $\mathsf{MH\,Adv}_{\mathcal{A}}$ (defined in Game 1) is negligible. It follows that $\mathsf{MH\,Adv}_{\mathcal{A}}[N+1]$ is negligible. For simplicity, say $\mathsf{MH\,Adv}_{\mathcal{A}}[N+1] = 0$. Then, by the standard hybrid argument there exists a $j \in \{1, \ldots, N\}$ such that $\left|\mathsf{MH\,Adv}_{\mathcal{A}}[j] - \mathsf{MH\,Adv}_{\mathcal{A}}[j+1]\right| > \epsilon/N$. In other words, this $\mathcal{A}$ is somehow able to distinguish $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, j, M)$ from $\mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, j+1, M)$ for some M and S. But then $\mathcal{A}$ can be directly used to win the ABE index hiding game.

More precisely, we show in Appendix B that for all adversaries $\mathcal{A}$ there exists an adversary $\mathcal{B}$ such that for all $i = 1, \ldots, N$ we have

$$\left|\mathsf{MH\,Adv}_{\mathcal{A}}[i] - \mathsf{MH\,Adv}_{\mathcal{A}}[i+1]\right| \leq 2 \cdot \mathsf{IH\,Adv}_{\mathcal{B}}[i] \tag{1}$$

Then

$$\left|\mathsf{MH\,Adv}_{\mathcal{A}}[1] - \mathsf{MH\,Adv}_{\mathcal{A}}[N+1]\right| \leq \sum_{i=1}^{n} \left|\mathsf{MH\,Adv}_{\mathcal{A}}[i] - \mathsf{MH\,Adv}_{\mathcal{A}}[i+1]\right| \leq 2 \sum_{i=1}^{n} \mathsf{IH\,Adv}_{B}[i]$$

But since $\mathcal{E}$ is a secure ABE we know that $\mathsf{MH\,Adv}_{\mathcal{A}}[N+1]$ and $\mathsf{IH\,Adv}_{B}[i]$ for $i = 1, \ldots, N$ are negligible for any $\mathcal{A}$. Therefore, $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ is negligible, as required. $\square$

2.2.2 Trace and Revoke

A trace and revoke system is a broadcast system with a tracing algorithm. We formally define trace and revoke systems in Appendix A along with the games used to define security. We show here that a secure ABE directly gives a trace and revoke system. In particular, we show a tracing algorithm for the broadcast system $\mathcal{E}_{\mathrm{B}E}$ above. The tracing algorithm uses a general tracing method, previously used in [3, 24, 21, 6]. For a given $\epsilon > 0$ and a set $S_{\mathcal{D}}$ the tracing algorithm $\mathit{Trace}^{\mathcal{D}}(S_{\mathcal{D}}, \mathrm{PK}, \epsilon)$ works as follows.

1. Initialize set T to the empty set.
2. For $i = 1$ to N, do the following:
   (a) The algorithm repeats the following $8\lambda(N/\epsilon)^2$ times:
      i. Sample M from the finite message space at random.
      ii. Let $C \xleftarrow{R} \mathit{Encrypt}_{\mathrm{ABE}}(S_{\mathcal{D}}, \mathrm{PK}, i, M)$.
      iii. Call oracle $\mathcal{D}$ on input C, and compare the output of $\mathcal{D}$ to M.
   (b) Let $\hat{p}_i$ be the fraction of times that $\mathcal{D}$ decrypted the ciphertexts correctly.
   (c) If $\hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4N)$, then add i to set T.
3. Output the set T.

Note that the running time of *Trace* is cubic in N. It can be made (almost) quadratic using binary search instead of a linear scan.

Let $\mathcal{E}_{\mathrm{T}R} = (\mathit{Setup}_{\mathrm{ABE}}, \mathit{Encrypt}, \mathit{Decrypt}_{\mathrm{ABE}}, \mathit{Trace})$ be the resulting trace and revoke system. Note $\mathcal{E}_{\mathrm{T}R}$ is just the broadcast system $\mathcal{E}_{\mathrm{B}E}$ with the tracing algorithm *Trace*. We show that $\mathcal{E}_{\mathrm{T}R}$ is secure in the sense of Definition A.1, namely fully collusion resistant against an adaptive adversary.

Theorem 2.3. *If $\mathcal{E}$ is a secure ABE then for $\mathcal{E}_{TR}$ the quantity $\mathsf{TR\,Adv}_{\mathcal{A}}$ defined in Appendix A is negligible.*

Proof Sketch. Let $p_i = \Pr[\mathcal{D}(\mathit{Encrypt}_{\mathrm{ABE}}(S_{\mathcal{D}}, \mathrm{PK}, i, M)) = M]$. We know that that $p_1 \geq \epsilon$ and p_{N+1} is negligible. The former follows from the fact that $\mathcal{D}$ is a useful decoder. The later follows directly from the ABE 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 T output by $\mathit{Trace}^{\mathcal{D}}(S_{\mathcal{D}}, \mathrm{PK},\ \epsilon)$ is non-empty.

It remains to show that whenever $\hat{p}_j - \hat{p}_{j+1} > \epsilon/(4N)$ we have that $j \in S_{\mathcal{D}} \cap U$. For such j we know, by Chernoff, that with overwhelming probability $p_j - p_{j+1} \geq \epsilon/(8N)$. We can now show that the ABE index hiding game implies $j \in U$ and $j \in S_{\mathcal{D}}$. Clearly $j \in S_{\mathcal{D}}$ since otherwise, even given all the secret keys, there is no hope of distinguishing p_j from p_{j+1}. But if $j \in S_{\mathcal{D}}$ and $j \notin U$ then $\mathcal{D}$ must distinguish p_j from p_{j+1} without the key SK_j. Again, such a $\mathcal{D}$ can be directly used to win the ABE index hiding game. Hence, $j \in S_{\mathcal{D}} \cap U$. We give the full proof details in the full version of the paper. $\square$

3 Background and complexity assumptions

Our traitor tracing system uses bilinear groups of composite order. We review the definition of such groups and then state our complexity assumptions. We follow [5] in which composite order bilinear groups were first introduced.

The assumptions we make are a little stronger than the ones in [6]. These stronger assumptions are needed to make the system publicly traceable, namely not requiring a secret tracing key. Without public traceability a direct variant of the system in this paper can be proven secure under the exact same assumptions used in [6].

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

$$\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G}(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{R} \mathbb{G}_p \\
&a, b, c \xleftarrow{R} \mathbb{Z}_p \\
&\bar{Z} \leftarrow \Big((n, \mathbb{G}, \mathbb{G}_T, e),\ \ g_p,\ \ g_p^a,\ \ g_p^b,\ \ g_p^c,\ \ g_p^{(b^2)}\Big) \\
&T \leftarrow g_p^{abc} \\
&\text{Output} \ \ (\bar{Z},\ T)
\end{aligned}$$

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

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

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

Definition 3.1. *We say that $\mathcal{G}$ satisfies the decision (modified) 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.*

The assumption is a little stronger than the corresponding assumption in [6] since we also give $g_p^{(b^2)}$ to the adversary.

Diffie-Hellman Subgroup Decision Assumption. The Diffie-Hellman subgroup decision assumption states that a random element in $\mathbb{G}_q$ is indistinguishable from a random element in $\mathbb{G}$, even when an ElGamal encryption of a $\mathbb{G}_p$ element is provided. More precisely, for a given group generator $\mathcal{G}$ define the following distribution $P(\lambda)$:

$$\begin{aligned}
&(p, q, \mathbb{G}, \mathbb{G}_T, e) \xleftarrow{R} \mathcal{G}(\lambda), \quad n \leftarrow pq \\
&g, h \xleftarrow{R} \mathbb{G}, \quad v_p \xleftarrow{R} \mathbb{G}_p \\
&a \xleftarrow{R} \mathbb{Z}_q, \quad b \xleftarrow{R} \mathbb{Z}_n, \\
&\bar{Z} \leftarrow \big((n, \mathbb{G}, \mathbb{G}_T, e),\ g,\ h,\ g^b v_p,\ h^b,\ g^{pa},\ h^{pa}\big) \\
&\text{Output} \ \ \bar{Z}
\end{aligned}$$

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

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

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

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

The Diffie-Hellman subgroup decision assumption is stronger than the subgroup decision assumption introduced in [5] and also used in [6]. In our definition the adversary is also given an ElGamal encryption of an element v_p that is known to be in $\mathbb{G}_p$. Furthermore, the adversary is given $g^{pa},\ h^{pa} \in \mathbb{G}_q$.

Bilinear Subgroup Decision Assumption. The Bilinear Subgroup Decision (BSD) assumption 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 $P(\lambda)$:

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

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

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

where $\bar{Z} \xleftarrow{R} P(\lambda)$, $T \xleftarrow{R} \mathbb{G}_p$, and $R \xleftarrow{R} \mathbb{G}$. 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 $\mathsf{BSD}\,\mathsf{Adv}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ.*

4 An Efficient Augmented Broadcast Encryption System

In this section we will give our construction of an Augmented Broadcast Encryption (ABE) system that has ciphertexts and private keys of size $O(\sqrt{N})$.

We begin by offering some intuition into the design and technical novelty of our scheme. An Augmented Broadcast Encryption system must have both broadcast and tracing properties. To achieve this we will use some techniques from the broadcast encryption system of [4] and the traitor tracing system of [6],

A trace and revoke system, however, cannot simply be constructed from combining a broadcast encryption system and a tracing system in a naive manner. Consider the following (misguided) approach. Suppose we created both a broadcast encryption and traitor tracing system each for N users, where each user has the same index in both systems. To encrypt a message M, an algorithm splits the message randomly into two pieces M_b, M_t such that $M_b \cdot M_t = M$ and then encrypts M_b under the broadcast system and M_t under the tracing system. In order to decrypt a message a single user will need to be able to decrypt under both systems. However, is two users, Alice and Bob collude to make a pirate decoder they can break this construction. They will simply use Alice's key to decrypt the ciphertext from the broadcast system and Bob's key to decrypt the ciphertext from the tracing system. The tracing algorithm will identify Bob as a traitor. However, after Bob is revoked (from the broadcast system) the decoder will still be functional and moreover will continue to identify Bob as the traitor even though he was already revoked!

The principle behind resisting this type of attack is construct user's private keys in such a way that they must be simultaneously used for both the broadcast and tracing portions of a trace and revoke system. We are able to achieve a secure ABE system by preventing colluding users from decomposing the two systems — the two sub-systems are essentially intertwined. In order to achieve this we multiply keys of the two portions together. Additionally, unlike [4] and [6] private keys in our system are randomized to prevent such attacks.

The other contribution of our scheme is that it allows for public traceability. This comes from the fact that there exists a public key algorithm for encrypting to arbitrary index. In [6] the public encryption algorithm could only be used to broadcast a message to everyone and a secret tracing key was required for encrypting to arbitrary indices. The reason behind this was that "column" ciphertexts needed to be randomized in the $\mathbb{G}_p$ subgroup, while kept well-formed in the $\mathbb{G}_q$ subgroup. The most natural way to do this is to give an element of $\mathbb{G}_p$ as part of the public parameters. However, giving our such an element allows an attacker to break our scheme. In our construction we construct the parameters in a novel way that allows for this type of public encryption without giving out an element of $\mathbb{G}_p$.

Notation We will express our ABE system using the same two index notation as the tracing traitors system of [6]. We assume that the number of users, N in the system equals m^2 for some m. If the number of 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 $1 \leq x, y \leq m$.

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. Additionally, 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 $(i, j = m)$.

We emphasize that the use of pairwise notation (i, j) is purely a notational convenience for describing our system.

4.1 Construction

We will assume there are $N = m^2$ users in the system and we will address each user will be assigned a unique pair of indexes (x, y) where $1 \leq x, y \leq m$. A detailed description of the algorithms follows:

Setup$_{\mathbf{ABE}}(N = m^2,\ \lambda)$ The setup algorithm takes as input the number of users N and a security parameter λ. 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 $g = g_p g_q, h = h_p h_q \in \mathbb{G}$. Additionally it chooses random elements $u_{p,1}, \ldots, u_{p,m} \in \mathbb{G}_p$,$u_{q,1}, \ldots, u_{q,m} \in \mathbb{G}_q$, and defines $u_i = u_{p,i} u_{q,i}$ for $i = 1, \ldots, m$. Next it chooses random exponents

$$\delta, r_1, \ldots, r_m, \quad c_1, \ldots, c_m, \quad \alpha_1, \ldots, \alpha_m \ \in \ \mathbb{Z}_n \quad \text{and} \quad \beta \in \mathbb{Z}_q \quad \text{and} \quad \gamma \in \mathbb{Z}_p$$

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

$$\left[\begin{array}{c} g, h, V = g^{\delta} g_p^{\gamma}, \tilde{V} = h^{\delta} \\ E_q = g_q^{\beta}, E_1 = g^{\beta r_1}, \ldots, E_m = g^{r_m}, E_{q,1} = g_q^{\beta r_1}, \ldots, E_{q,m} = g_q^{\beta r_m}, \\ F_1 = h^{r_1}, \ldots, F_m = h^{r_m}, F_{q,1} = h_q^{\beta r_1}, \ldots, F_{q,m} = h_q^{\beta r_m} \\ G_1 = e(g,g)^{\alpha_1}, \ldots, G_m = e(g,g)^{\alpha_m}, G_{q,1} = e(g_q, g_q)^{\beta \alpha_1}, \ldots, G_{q,m} = e(g_q, g_q)^{\beta \alpha_m} \\ H_1 = g^{c_1}, \ldots, H_m = g^{c_m} \\ U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^{\beta}, \ldots, U_{q,m} = u_{q,m}^{\beta} \end{array}\right]$$

The authority creates the private key for user (x, y) by first choosing a random exponent $\sigma_{x,y} \in \mathbb{Z}_n$ then generates it as:

$$\begin{aligned} \mathrm{SK}_{x,y} &= \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, , d_{y+1}, \ldots, d_m\right) \\ &= \left(g^{\alpha_x} g^{r_x c_y} \cdot u_y^{\sigma_{x,y}}, g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}\right) \end{aligned}$$

The public parameters $u_{q,1}^{\beta}, \ldots, u_{q,m}^{\beta}$ are related to the broadcast potion of the system, while the other parameters are related to the traitor tracing portion of the system. The secret key component $d'_{x,y}$ contains the secret key g^{α_x} blinded by $g^{r_x c_y}$, which is related to traitor tracing and $u_y^{\sigma_{x,y}}$, which is related the broadcast encryption system. An important technical novelty is that since $d'_{x,y}$ contains both pieces multiplied together, and an attacker will be unable to separate these pieces out and decrypt the tracing and broadcast portions of the system separately. Thus, for a key to be useful for decrypting a ciphertext it must be both in the broadcast set of the ciphertext and have an index greater than or equal to the encrypted index.

***Encrypt*$_{\mathbf{ABE}}$$(S, \mathbf{PK}, (i,j), M)$** The *Encrypt*$_{\mathrm{ABE}}$ algorithm is primarily used for tracing. It encrypts a message M to the subset of receivers that are in S and that have row values greater than i or both row value equal to i and column values greater than j. The algorithm encrypts messages $M \in \mathbb{G}_T$. It first chooses random

$$t,\ w_1, \ldots, w_m,\ s_1, \ldots, s_m \in \mathbb{Z}_n$$

$$b_1, \ldots, b_{j-1} \in \mathbb{Z}_n$$

$$(\nu_{1,1}, \nu_{1,2}, \nu_{1,3}), \ldots, (\nu_{i-1,1}, \nu_{i-1,2}, \nu_{i-1,3}) \in \mathbb{Z}_n^{(3)}$$

Let S_x denote the set of all values y such that the user (x, y) is in the set S. For each row x we create five ciphertext components $(R_x, \tilde{R}_x, T_x, A_x, B_x)$ as follows:

$$\begin{array}{llll}
\text{if } x > i: & R_x = E_{q,x}^{s_x} & \tilde{R}_x = F_{q,x}^{s_x} & A_x = E_q^{s_x t} \\
& T_x = (\prod_{k \in S_x} U_{q,k})^{s_x t} & B_x = M G_{q,x}^{s_x t} & \\
& & & \\
\text{if } x = i: & R_x = E_x^{s_x} & \tilde{R}_x = F_x^{s_x} & A_x = g^{s_x t} \\
& T_x = (\prod_{k \in S_x} U_k)^{s_x t} & B_x = M G_x^{s_x t} & \\
& & & \\
\text{if } x < i: & R_x = g^{\nu_{x,1}} & \tilde{R}_x = h^{\nu_{x,1}} & A_x = g^{\nu_{x,2}} \\
& T_x = (\prod_{k \in S_x} U_k)^{\nu_{x,2}} & B_x = e(g,g)^{\nu_{x,3}} &
\end{array}$$

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

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

We note that for $y < j$ the $\mathbb{G}_p$ subgroup will be completely random in C_y.

The final ciphertext, containing $O(\sqrt{N} = m)$ group elements, consists of

$$\left((R_x, \tilde{R}_x, T_x, A_x, B_x)_{x=1}^m, \quad (C_y, \tilde{C}_y)_{y=1}^m \right)$$

The T_x values can be viewed a broadcast encryption to all members of the row x that are in the sub-target set S_x. We can also see how the parameters allow for public encryption (which in turn gives public traceability) to an arbitrary index (i,j). The public parameters that are from the G_q subgroup are used for the encryption to rows greater than i. The public parameters values $V, \tilde{V}$ are used to make column components that are well formed in the $\mathbb{G}_q$ subgroup and random in the $\mathbb{G}_p$ subgroup. By forming the parameters in this way we can accomplish this without giving out a group element from $\mathbb{G}_p$, which would break the difficulty of subgroup hiding.

***Decrypt*$_{\mathbf{ABE}}$**$(S, (x,y), \mathbf{SK}_{x,y}, C, \mathbf{PK})$ If user $(x,y) \in S$ it can attempt to decrypt by first computing a temporary key

$$K'_{x,y} = d'_{x,y} \prod_{\substack{k \in S_x \\ k \neq y}} d_{x,y,k}$$

Then it computes:

$$B_x / \left(e(K'_{x,y}, A_x) e(\tilde{R}_x, \tilde{C}_y) / \left(e(R_x, C_y) e(T_x, d''_{x,y}) \right) \right).$$

Suppose that the ciphertext was encrypted to index (i,j) and that $x > i$ then in decryption pairing $e(K'_{x,y}, A_x)$ gives the value $e(g, g_q)^{\alpha_x s_x t} e(g, \prod_{k \in S_x} u_{q,k})^{s_x t \theta_{x,y}} e(g, g_q)^{s_x t r_x c_y}$. The other pairings are used to divide out $e(g, \prod_{k \in S_x} u_{q,k})^{s_x t \theta_{x,y}} e(g, g_q)^{s_x t r_x c_y}$ and get the blinding factor $e(g, g_q)^{\alpha_x s_x t}$. If $x = i$ and $y \geq j$ then decryption can be explained in a similar way except the target groups are in $\mathbb{G}_T$ instead of the subgroup $\mathbb{G}_{T,q}$,

5 Security

In this section we prove the security of our Augmented Broadcast Encryption system by showing that it is secure under both games defined in Section 2.

The structure of our proofs is similar to that of [6]. There are two cases where there are important differences between what we prove here and what was given in [6]. The first is to show that it is difficult to distinguish between an encryption to indices (i, j) and indices $(i, j + 1)$ even when the attacker has key $K_{i,j}$ if user (i, j) is revoked. This is shown in the proof of Lemma 5.2.

Secondly, we need to prove that the public parameters given out to allow for public encryption to arbitrary indices do not break the security of our scheme. This is reflected in our proof of Claim 5.7, which shows that deciding subgroups is still hard even if the adversary has access to our public parameters. The other portions of the proofs are conceptually similar to [6], however, we include them for completeness.

5.1 Proof of Security for Game 1 (Message Hiding)

The argument for security of the Message Hiding game is very straightforward since an encryption to index $(m + 1, 1)$ contains no information about the ciphertext. The simulator simply runs the actual Setup algorithm and encrypts message M_β to set S and index $N + 1$. Since, all B_x values contain no information about the ciphertext the bit β is perfectly hidden and the adversary's advantage is 0.

5.2 Proof of Security for Game 2 (Index-Hiding)

For clarity we present our Index-Hiding proofs in a structure similar to that of [6]. We state our main theorem and prove its security from a series our claims and lemmas whose proofs we defer to the appendix.

Theorem 5.1. *Suppose that the (Modified) 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Diffie-Hellman Subgroup Decision assumptions hold. Then no polynomial time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with non-negligible advantage.*

First we consider the case where the adversary $\mathcal{A}$ chooses to distinguish between an encryption to indices (i, j) and $(i, j + 1)$ where $j < m$. We state the following lemma:

Lemma 5.2. *Suppose that the Decision (Modified) 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 with non-negligible advantage.*

In this game we build a simulator that will guess whether the bit $\tilde{s}$. If $\tilde{s} = 0$ and $(i, j) \in S$ then the proof is very similar to that from the traitor tracing system of [6]. However, if $\tilde{s} = 1$ and $(i, j) \notin S$ then our simulator will need to generate the key $K_{i,j}$ for the adversary and still simulate the challenge ciphertext. The proof of this case captures the security that is gained by our particular method of composing a broadcast and traitor tracing system to make an Augmented Broadcast Encryption system.

We prove this lemma in Appendix C.1.

We now consider the case when the adversary $\mathcal{A}$ attempts to distinguish between an encryption to (i, m) and one to $(i + 1, 1)$ for some $1 \le i < m$. We 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 $y \ge j$. We state our lemma and then prove it.

Lemma 5.3. *Suppose that the Decision (Modified) 3-party Diffie-Hellman, Bilinear Subgroup Decision, and Diffie-Hellman Subgroup Decision assumptions hold. Then no polynomial time adversary $\mathcal{A}$ can succeed in the Index-Hiding game with non negligible advantage.*

We define a sequence of hybrid experiments:

- 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.

Claim 5.4. *Suppose that the Decision (Modified) 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 this game the adversary attempts to distinguish whether the ciphertext component C_m is well formed in the $\mathbb{G}_p$ subgroup. This game is exactly the same as the one proved in Lemma 5.2.

Claim 5.5. *Suppose that the Decision (Modified) 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 Diffie-Hellman 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. □

6 Conclusion

We constructed a fully collusion resistant trace and revoke system for arbitrary sets S where the size of ciphertexts and private keys is $O(\sqrt{N})$. The system is publicly traceable and secure against adaptive adversaries which is unusual for algebraic constructions. Instead of directly building a trace and revoke system we constructed a simpler primitive called Augmented Broadcast Encryption

(ABE) with $O(\sqrt{N})$-size ciphertexts and private keys. We showed that ABE is sufficient for both broadcast encryption and tracing traitors. While we proved our broadcast secure under plaintext attacks, it is not difficult to modify it slightly and apply the methods of Canetti, Halevi, and Katz [8] for security against CCA attacks. We hope that future research will lead to ABEs with even shorter ciphertexts and private keys.

References

[1] J. Anzai, N. Matsuzaki, and T. Matsumoto. A quick key distribution scheme with entity revocation. In *Proc. of Asiacrypt '99*, pages 333–347. Springer-Verlag, 1999.

[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, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO '05*, pages 258–275, 2005.

[5] 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.

[6] Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Eurocrypt '06*, 2006.

[7] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2003.

[8] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Proceedings of Eurocrypt 2004*, LNCS, pages 207–222, 2004.

[9] Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT '05*, pages 542–558, 2005.

[10] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *Proceedings of Crypto '94*, volume 839 of *LNCS*, pages 257–270, 1994.

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

[12] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Proceedings of the Digital Rights Management Workshop 2002*, volume 2696 of *LNCS*, pages 61–80. Springer, 2002.

[13] 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.

[14] Yevgeniy Dodis, Nelly Fazio, Aggelos Kiayias, and Moti Yung. Scalable public-key tracing and revoking. *Distributed Computing*, 17(4):323–347, 2005. Extended abstract in PODC '03.

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

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

[17] 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.

[18] Juan A. Garay, Jessica Staddon, and Avishai Wool. Long-lived broadcast encryption. In *CRYPTO '00: Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, pages 333–352. Springer-Verlag, 2000.

[19] 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.

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

[21] 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.

[22] 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.

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

[24] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. In *Proceedings of Crypto '01*, volume 2139 of *LNCS*, pages 41–62, 2001.

[25] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial cryptography 2000*, volume 1962 of *LNCS*, pages 1–20. Springer, 2000.

[26] 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.

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

[28] 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.

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

[30] 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.

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

[32] 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.

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

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

[35] Doug R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Des. Codes Cryptography*, 12(3):215–243, 1997.

[36] Doug R. Stinson and Tran Van Trung. Some new results on key distribution patterns and broadcast encryption. *Des. Codes Cryptography*, 14(3):261–279, 1998.

[37] W. Tzeng and Z. Tzeng. A public-key traitor tracing scheme with revocation using dynamic shares. In *Proc. of PKC 2001*, pages 207–224. Springer-Verlag, 2001.

[38] 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 Trace and revoke systems

A Trace and Revoke (TR) system is a broadcast encryption system with an additional tracing algorithm. We describe TR systems where encryption is public-key, but tracing requires a secret tracing key. A TR system thus consists of four algorithms:

Setup(N, λ) A probabilistic algorithm that takes as input N, the number of users in the system, and a security parameter λ. The algorithm runs in polynomial time in λ and outputs a public key PK and private keys $\mathrm{SK}_1, \dots, \mathrm{SK}_N$, where SK_u is for user u.

Encrypt$(S, \mathbf{PK}, M)$ Takes as input a subset $S \subseteq \{1, \dots, N\}$, a public key PK, and a message M. It outputs a ciphertext C intended for a specific recipient set S.

Decrypt$(S, j, \mathbf{SK}_j, C, \mathbf{PK})$ Takes as input a subset $S \subseteq \{1, \dots, N\}$, the private key SK_j for user j, a ciphertext C, and the public key PK. The algorithm outputs a message M or $\perp$.

Trace$^{\mathcal{D}}(S, \mathbf{PK}, \ \epsilon)$ The tracing algorithm is an oracle algorithm that is given as input a set $S \subseteq \{1, \ldots, N\}$, the public key PK 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 outputs a set $T \subseteq \{1, \ldots, N\}$.

The system must satisfy the same **correctness property** as for broadcast encryption, namely: for all subsets $S \subseteq \{1, \ldots, N\}$, all $j \in \{1, \ldots, N\}$, and all messages M:

Let $(\mathrm{PK}, (\mathrm{SK}_1, \ldots, \mathrm{SK}_N)) \overset{R}{\leftarrow} \mathit{Setup}_{\mathrm{ABE}}(N, \Lambda)$ and $c \overset{R}{\leftarrow} \mathit{Encrypt}(S, \mathrm{PK}, M)$.

If $j \in S$ then $\mathit{Decrypt}(S, j, \mathrm{SK}_j, c, \mathrm{PK}) = M$.

Security. We define security of a trace and revoke system using two natural games. The **Message Hiding Game** is the same as for a broadcast encryption system secure against an adaptive attacker [13, 4]. We described the game in Section 2.2.1. We let $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ denote the advantage of $\mathcal{A}$ in winning the game.

The **Tracing Game** ensures that the tracing algorithm successfully traces any pirate decoder, no matter how many secret keys were used to create the decoder. The adversary's goal is to build a pirate decoder $\mathcal{D}$ that will decrypt all ciphertexts encrypted for a certain set $S_{\mathcal{D}}$. The tracing algorithm's goal is extract from $\mathcal{D}$ at least one of the keys $u \in S_{\mathcal{D}}$ that were used to construct $\mathcal{D}$. The broadcaster will encrypt all future messages to the set $S' = S_{\mathcal{D}} \setminus \{u\}$. If the decoder $\mathcal{D}$ can decode ciphertexts encrypted for S' then we run the tracing algorithm again, this time giving it the set S', to extract another of the pirate's keys in $S_{\mathcal{D}}$. This process continues, iteratively shrinking S', until $\mathcal{D}$ stops functioning.

The following game ensures that this process will eventually disable $\mathcal{D}$ without disabling any innocent parties. The game is defined between a challenger and an adversary $\mathcal{A}$ (both are given N, λ and ϵ as input):

1. The challenger runs $\mathit{Setup}(N, \lambda)$ to obtain PK and $\mathrm{SK}_1, \ldots, \mathrm{SK}_N$. It provides PK to $\mathcal{A}$.

2. The adversary issues *adaptive* private key queries. It repeatedly sends values $j \in \{1, \ldots, N\}$ to the challenger and the challenger responds with SK_j. Let $U \subseteq \{1, \ldots, N\}$ be the total set of keys obtained by the adversary.

3. Finally, the adversary $\mathcal{A}$ outputs a set $S_{\mathcal{D}} \subseteq \{1, \ldots, N\}$ and 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 $\mathit{Trace}^{\mathcal{D}}(S_{\mathcal{D}}, \mathrm{PK}, \epsilon)$ to obtain a set $T \subseteq \{1, \ldots, N\}$.

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

- For a randomly chosen M in the finite message space, we have that

$$\Pr[\mathcal{D}(\mathit{Encrypt}(S_{\mathcal{D}}, \mathrm{PK}, M)) = M] \geq \epsilon$$

- The set T is either empty, or is not a subset of $U \cap S_{\mathcal{D}}$.

We denote by $\mathsf{TR\,Adv}_{\mathcal{A}}$ the probability that adversary $\mathcal{A}$ wins this game.

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 need only decrypt 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 [21].

Definition A.1. *We say that an N-user Trace-and-Revoke system is secure if for all polynomial time adversaries $\mathcal{A}$ we have that* $\mathsf{MH\,Adv}_{\mathcal{A}}[1]$ *and* $\mathsf{TR\,Adv}_{\mathcal{A}}$ *are negligible functions of λ.*

B ABE implies Broadcast Encryption

To prove that the broadcast encryption system in Section 2.2.1 is secure it remains to prove that Equation (1) holds for all $i = 1, \ldots, N$. Consider a specific $i \in \{1, \ldots, N\}$. Adversary $\mathcal{B}$ plays the index hiding game with input i and works as follows:

- **Setup** $\mathcal{B}$ receives PK and the set of private keys $\big\{\mathrm{SK}_j \text{ s.t. } j \neq i\big\}$ from its challenger. $\mathcal{B}$ runs adversary $\mathcal{A}$ and gives it PK.
- **Query** $\mathcal{A}$ issues adaptive private key queries. To respond to a query for SK_j adversary $\mathcal{B}$ does:
  - If $j \neq i$ then $\mathcal{B}$ simply gives SK_j to $\mathcal{B}$.
  - Otherwise, $j = i$. Then $\mathcal{B}$ sends the bit $\tilde{s} = 1$ to its challenger and receives SK_i in return. It gives SK_i to $\mathcal{A}$.

  Let $S_0 \subseteq \{1, \ldots, N\}$ denote the set of private keys requested by $\mathcal{A}$. Define the complement as $\overline{S_0} = \{1, \ldots, N\} \smallsetminus S_0$.
- **Challenge** Finally, $\mathcal{A}$ outputs a set $S \subseteq \overline{S_0}$ and two equal length messages M_0, M_1. Then $\mathcal{B}$ flips a coin $\gamma \in \{0, 1\}$ and gives S and M_γ to its challenger. Note that if $\mathcal{B}$ sent $\tilde{s} = 1$ to its challenger then $i \notin S$, as required.

  $\mathcal{B}$ receives back $C \overset{R}{\leftarrow} \mathit{Encrypt}_{\mathrm{ABE}}(S, \mathrm{PK}, i + \beta, M_\gamma)$ for some random $\beta \in \{0, 1\}$. It gives C to $\mathcal{A}$.
- **Guess** Finally, $\mathcal{A}$ outputs a guess γ' for γ. If $\gamma' = \gamma$ then $\mathcal{B}$ outputs 0. Otherwise, $\mathcal{B}$ outputs 1.

Now, observe that when $\beta = 0$ then $\mathcal{B}$ is emulating perfectly an $\mathsf{MH\,Adv}_{\mathcal{A}}[i]$ challenger. When $\beta = 1$ then $\mathcal{B}$ is emulating perfectly an $\mathsf{MH\,Adv}_{\mathcal{A}}[i+1]$ challenger. A standard argument now shows that $|\mathsf{MH\,Adv}_{\mathcal{A}}[i] - \mathsf{MH\,Adv}_{\mathcal{A}}[i+1]| \leq 2 \cdot \mathsf{IH\,Adv}_{\mathcal{B}}[i]$ as required.

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 Decision (Modified) 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. The simulator runs the core part of the simulation in the $\mathbb{G}_p$ subgroup and chooses all values outside the subgroup for itself. Our formal proof follows.

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

$$g_p, A = g_p^a, B = g_p^b, C = g_p^c, D = g^{b^2}, T.$$

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.

The simulator receives the target indices (i, j) from the adversary. The adversary will eventually behave in one of two different ways.

Case I The adversary will give a bit $\tilde{s} = 0$ specify a set S where $(i, j) \in S$ and the simulator needs to generate all keys, except key $K_{i,j}$.

Case II The adversary will specify a bit $\tilde{s} = 1$ and a target set S such that $(i, j) \notin S$ and the simulator needs to generate all keys.

At this point the simulator does not know how the adversary will behave so it will need to guess which case it will be in. Since the simulator's output will be independent of which case it guesses the simulator will be able to continue the simulation with probability 1/2.

We describe how the simulator will behave in each case:

Case I Since the game is played in the subgroup $\mathbb{G}_p$, the simulator chooses 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 $r'_{p,1}, \ldots, r'_{p,m}, c'_{p,1}, \ldots, c'_{p,m} \in \mathbb{Z}_p$.

Finally, it chooses $\delta \in \mathbb{Z}_n, \gamma, \mu_{p,1}, \ldots, \mu_{p,m} \in \mathbb{Z}_p$ at random and chooses random $u_{q,1}, \ldots, u_{q,m} \in \mathbb{G}_q$. It then sets $u_{p,1} = g_p^{\mu_{p,1}}, \ldots, u_{p,m} = g_p^{\mu_{p,m}}$ and it assigns $u_i = u_{q,i} u_{p,i}$ for $i = 1, \ldots, m$.

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

$$\left[\begin{array}{c}
g = g_q g_p, h = h_q B, V = g^\delta g_p^\gamma, \tilde{V} = h^\delta \\
E_q = g_q^\beta, E_x = g_q^{r_{q,x}} g_p^{r'_{p,x}} : x \neq i, \quad E_i = g_q^{r_{q,i}} B^{r'_{p,i}} \\
E_{q,1} = g_q^{\beta r_{q,1}}, \ldots, E_{q,m} = g_q^{\beta r_{q,m}} \\
F_x = g_q^{r_{q,x}} B^{r'_{p,x}} : x \neq i, \quad F_i = g_q^{r_{q,i}} D^{r'_{p,i}} \\
F_{q,1} = h_q^{\beta r_{q,1}}, \ldots, F_{q,m} = h_q^{\beta r_{q,m}} \\
G_1 = e(g,g)^{\alpha_1}, \ldots, G_m = e(g,g)^{\alpha_m}, G_{q,1} = e(g_q, g_q)^{\beta \alpha_1}, \ldots, G_{q,m} = e(g_q, g_q)^{\beta \alpha_m} \\
H_y = g_q^{c_{q,y}} g_p^{c'_{p,y}} : y \neq j, \quad H_j = g_q^{c_{q,j}} C^{c'_{p,j}} \\
U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^\beta, \ldots, U_{q,m} = u_{q,m}^\beta
\end{array}\right]$$

Next, it chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all $(x, y) \neq (i, j)$ and creates private keys for all users except (i, j) as:

$$K_{x,y} = \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, , d_{y+1}, \ldots, d_m\right)$$
$$= \begin{cases} (g^{\alpha_x} g_q^{r_{q,x} c_{q,y}} g_p^{r'_{p,x} c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x \neq i, y \neq j \\ (g^{\alpha_x} g_q^{r_{q,x} c_{q,y}} B^{r'_{p,x} c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x = i, y \neq j \\ (g^{\alpha_x} g_q^{r_{q,x} c_{q,y}} C^{r'_{p,x} c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x \neq i, y = j \end{cases}$$

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

The simulator gives a bit $\tilde{s}$ as the query. If $\tilde{s} = 1$ the simulation aborts. In the challenge phase the adversary first gives the simulator a target set S and 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 $s'_p \in \mathbb{Z}_p$, $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$. We again let S_x denote the set of all values y such that the user (x, y) is in the set S. The simulator then creates the ciphertext as:

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

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 (Modified) 3-party Diffie-Hellman game. The simulator's advantage in the Index Hiding game will be exactly equal to $\mathcal{A}$'s advantage.

Case II The simulator chooses random generators $g_q, h_q \in \mathbb{G}_q$ and random exponents β, $r_{q,1}, \ldots, r_{q,m}, c_{q,1}, \ldots, c_{q,m} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\delta, \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{p,1}, \ldots, r'_{p,m}, c'_{p,1}, \ldots, c'_{p,m} \in \mathbb{Z}_p$.

Finally, it chooses $\gamma, \mu_{p,1}, \ldots, \mu_{p,m} \in \mathbb{Z}_p$ at random and chooses random $u_{q,1}, \ldots, u_{q,m} \in \mathbb{G}_q$. It then sets $u_{p,1} = g_p^{\mu_{p,1}}, u_{p,j-1} = g_p^{\mu_{p,j-1}}, , u_{p,j+1} = g_p^{\mu_{p,j+1}}, \ldots, u_{p,m} = g_p^{\mu_{p,m}}$. It then sets $u_{p,j} = B^{r'_{p,i}} g_p^{\mu_{p,j}}$. It finally assigns $u_i = u_{q,i} u_{p,i}$ for $i = 1, \ldots, m$.

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

$$\left[\begin{array}{c}
g = g_q g_p, h = h_q B, V = g^{\delta} g_p^{\gamma}, \tilde{V} = h^{\delta} \\
E_q = g_q^{\beta}, E_x = g_q^{r_{q,x}} g_p^{r'_{p,x}} : x \neq i, \quad E_i = g_q^{r_{q,i}} B^{r'_{p,i}} \\
E_{q,1} = g_q^{\beta r_{q,1}}, \ldots, E_{q,m} = g_q^{\beta r_{q,m}} \\
F_x = g_q^{r_{q,x}} B^{r'_{p,x}} : x \neq i, \quad F_i = g_q^{r_{q,i}} D^{r'_{p,i}} \\
F_{q,1} = h_q^{\beta r_{q,1}}, \ldots, F_{q,m} = h_q^{\beta r_{q,m}} \\
G_1 = e(g,g)^{\alpha_1}, \ldots, G_m = e(g,g)^{\alpha_m}, G_{q,1} = e(g_q,g_q)^{\beta\alpha_1}, \ldots, G_{q,m} = e(g_q,g_q)^{\beta\alpha_m} \\
H_y = g_q^{c_{q,y}} g_p^{c'_{p,y}} : y \neq j, \quad H_j = g_q^{c_{q,j}} C^{c'_{p,j}} \\
U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^{\beta}, \ldots, U_{q,m} = u_{q,m}^{\beta}
\end{array}\right]$$

Next, it chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all $(x,y) \neq (i,j)$, values $\sigma'_{q,i,j} \in \mathbb{Z}_q$ and $\sigma'_{p,i,j} \in \mathbb{Z}_p$. It creates private keys for all users as:

$$\begin{aligned}
&K_{x,y} = \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, , d_{y+1}, \ldots, d_m\right) \\
&= \begin{cases}
(g^{\alpha_x} g_q^{r_{q,x}c_{q,y}} g_p^{r'_{p,x}c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x \neq i, y \neq j \\
(g^{\alpha_x} g_q^{r_{q,x}c_{q,y}} B^{r'_{p,x}c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x = i, y \neq j \\
(g^{\alpha_x} g_q^{r_{q,x}c_{q,y}} C^{r'_{p,x}c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x \neq i, y = j \\
(g^{\alpha_x} g_q^{r_{q,x}c_{q,y}} u_{q,y}^{\sigma'_{q,x,y}} C^{-c'_{p,j}\mu_{p,y}} (B^{r'_{p,x}} g_p^{\mu_{p,y}})^{\sigma_{p,x,y}}, & \\
\quad g_q^{\sigma'_{q,x,y}} C^{-c'_{p,j}} g_p^{\sigma'_{p,x,y}}, u_1^{\sigma'_{p,x,y}}, \ldots, u_{y-1}^{\sigma'_{p,x,y}}, , u_{y+1}^{\sigma_{p,x,y'}}, \ldots, u_m^{\sigma_{p,x,y'}}) & : x = i, y = j
\end{cases}
\end{aligned}$$

For the computation of key $K_{i,j}$ the simulator implicitly uses a value of $\sigma_{p,i,j} = -cc'_{p,i,j} + \sigma'_{p,i,j}$. This assignment allows for the cancellation of the hard to compute part of key i,j.

The adversary gets all the keys generated except key $K_{i,j}$ We note that all the simulator creates public and private with the same distribution as the real scheme.

The adversary specifies the query bit $\tilde{s}$, if $\tilde{s} = 0$ the simulator aborts, otherwise it gives the key $K_{i,j}$ to the adversary. Then, in the challenge phase the adversary gives the simulator the target set S such that $(i,j) \notin S$ and 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 $s'_p \in \mathbb{Z}_p$, $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$. We again let S_x denote the set of all values y such that the user (x,y) is in the set S. The simulator then creates the ciphertext as:

if $x > i$ $\quad R_x = g_q^{s_{q,x} r_{q,x}}$ $\quad \tilde{R}_x = h_q^{s_{q,x} r_{q,x}}$ $\quad A_x = g_q^{s_{q,x} t_q}$

$\qquad T_x = (\prod_{k \in S_x} u_{q,k})^{s_{q,x} t_q}$ $\quad B_x = Me(g_q, g_q)^{\alpha_x s_{q,x} t_q}$

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

$\qquad T_x = (\prod_{k \in S_x} u_{q,k})^{s_{q,x} t_q} A^{s'_p \sum_{k \in S_x} \mu_{p,k}}$

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

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

$\qquad T_x = (\prod_{k \in S_x} u_k)^{v_{x,2}}$ $\quad B_x = e(g, g)^{v_{x,3}}$

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

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

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

We point out that since $j \notin S_i$ that the formulation T_i is correct. In the Case II simulation the simulator is able to choose u_j in a special way such that it can compute the private key $K_{i,j}$ and it is still able to simulate the ciphertext since it knows user (i, j) is not in the target set S.

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 Decision (Modified) 3-party Diffie-Hellman game. The simulator's advantage in the Index Hiding game will be exactly equal to $\mathcal{A}$'s advantage, which by our assumption must be negligible. $\square$

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 Decision (Modified) 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 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 $g_p^{r_i} = B$, $g_p^{\alpha_i} = g_p^{ab}$, $g_p^{t_p} = C$, 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 $g = g_p g_q, h = g_p h_q$. 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 $\alpha_1, \ldots \alpha_{i-1},, \alpha_{i+1}, \ldots, \alpha_m \in \mathbb{Z}_n$. Finally, it chooses $\mu_{p,1}, \ldots, \mu_{p,m} \in \mathbb{Z}_p$ at random and chooses random $u_{q,1}, \ldots, u_{q,m} \in \mathbb{G}_q$. It then sets $u_{p,1} = g_p^{\mu_{p,1}}, \ldots, u_{p,m} = g_p^{\mu_{p,m}}$. It finally assigns $u_i = u_{q,i} u_{p,i}$ for $i = 1, \ldots, m$ and chooses random $\delta \in \mathbb{Z}_n$ and $\gamma \in \mathbb{Z}_p$.

The public parameters are published as:

$$\left[\begin{array}{c}
g, h, V = g^{\delta} g^{\gamma}, \tilde{V} = h^{\delta} \\[1ex]
E_q = g_q^{\beta}, E_x = g^{r_x} : x \neq i, \quad E_i = g_q^{r_{q,i}} B \\[1ex]
E_{q,x} = g_q^{\beta r_x} : x \neq i, \quad E_{q,i} = g_q^{\beta r_{q,i}} \\[1ex]
F_x = g^{d r_x} : x \neq i, \quad F_i = g_q^{d r_{q,i}} B \\[1ex]
F_{q,x} = g_q^{d \beta r_x} : x \neq i, \quad F_{q,i} = g_q^{d \beta r_{q,i}} \\[1ex]
G_x = e(g,g)^{\alpha_x} : x \neq i, \quad G_i = e(g_q, g_q)^{\alpha_{q,i}} e(A, B) \\[1ex]
G_{q,x} = e(g_q, g_q)^{\beta \alpha_x} : x \neq i. \quad G_{q,i} = e(g_q, g_q)^{\beta \alpha_{q,i}} \\[1ex]
H_1 = g^{c_q} A^{-1} g^{c'_{p,1}}, \ldots, H_m = g^{c_q} A^{-1} g^{c'_{p,m}} \\[1ex]
U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^{\beta}, \ldots, U_{q,m} = u_{q,m}^{\beta}
\end{array}\right]$$

The simulator chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all (x, y) and the private keys are created as:

$$\begin{aligned}
K_{x,y} &= \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1},, d_{y+1}, \ldots, d_m\right) \\
&= \begin{cases} g^{\alpha_x} (g_q^{c_{q,y}} A^{-1} g_p^{c'_{p,y}})^{r_x} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}},, u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : x \neq i \\ (g^{\alpha_{q,i}} g_q^{r_{q,i} c_{q,y}}) B^{c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}},, u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : x = i \end{cases}
\end{aligned}$$

The simulator gives out all keys except $K_{i,j}$. Then it receives the query bit $\tilde{s}$ and gives out $K_{i,j}$ if $\tilde{s} = 1$. Next, the adversary specifies a target set S and a challenge message,$M \in \mathbb{G}_T$. The simulator 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 $R_x, \tilde{R}_x, A_x, T_x, B_x$ values for $x < i$ are just created randomly and all $R_x, \tilde{R}_x, A_x, T_x, B_x$ 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:

$$\begin{array}{lll}
R_x = (g^{r_{q,i}} B)^{s_i} & \tilde{R}_x = (g^{r_{q,i}} B)^{d s_i} & A_x = (g^{t_q} C)^{s_i}, \\
T_x = \prod_{k \in S_x} (u_{q,k}^{t_q} C^{\mu_{p,k}})^{s_i} & B_x = M \left(e(g_q, g)^{\alpha_{q,i} t_q} T\right)^{s_i} &
\end{array}$$

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. It follows from our assumption ϵ must be negligible.

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. It gives out all keys except $K_{i,j}$ and then gives out $K_{i,j}$ if the adversary gives bit $\tilde{s} = 1$.

Next, it receives a target set S and a challenge message, $M \in \mathbb{G}_T$ from the adversary. It then 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. It follows from our assumption that the adversary's advantage must be negligible.

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 Decision 3-Party Diffie-Hellman game with advantage ϵ.

The simulator first receives a 3-Party Diffie-Hellman challenge, $k_q, A = k_q^a, B = k_q^b, C = k_q^c, T \in \mathbb{G}_q$ (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: $r_{p,1}, \dots, r_{p,m}, c_{p,1}, \dots, c_{p,m}, \alpha_{p,1}, \alpha_{p,m} \in \mathbb{Z}_p$. Then, it chooses $r'_{q,1}, \dots, r'_{q,m}, c'_{q,1}, \dots, c'_{q,m}, \alpha'_{q,1}, \alpha'_{q,m} \in \mathbb{Z}_q$. Finally, it chooses $\mu_{q,1}, \dots, \mu_{q,m} \in \mathbb{Z}_p$ at random and chooses random $u_{p,1}, \dots, u_{p,m} \in \mathbb{G}_p$. It then sets $u_{q,1} = A^{\mu_{p,1}}, \dots, u_{q,m} = A^{\mu_{q,m}}$. It finally assigns $u_i = u_{q,i} u_{p,i}$ for $i = 1, \dots, m$ and chooses random $\delta \in \mathbb{Z}_n$ and $\gamma \in \mathbb{Z}_p$.

The simulator can now publish the parameters as:

$$\left[\begin{array}{c}
g, h, V = g^\delta g_p^\gamma, \tilde{V} = h^\delta \\[1ex]
E_q = k_q^\beta, E_x = A^{r'_{q,m}} g_p^{r_{p,x}} : x \neq i, \quad E_i = k_q^{r'_{q,i}} g_p^{r_{p,i}} \\[1ex]
E_{q,x} = A^{\beta r'_{q,m}} : x \neq i, \quad E_{q,i} = k_q^{\beta r'_{q,i}} \\[1ex]
F_x = A^{d r'_{q,m}} g_p^{d r_{p,x}} : x \neq i, \quad F_i = k_q^{d r'_{q,i}} g_p^{d r_{p,i}} \\[1ex]
F_{q,x} = A^{d \beta r'_{q,m}} : x \neq i, \quad F_{q,i} = k_q^{d \beta r'_{q,i}} \\[1ex]
G_1 = e(g_p, g_p)^{\alpha_1} e(A, A)^{\alpha_1}, \dots, G_m = e(g_p, g_p)^{\alpha_m} e(A, A)^{\alpha_m}, \\[1ex]
G_{q,1} = e(A, A)^{\beta \alpha_1}, \dots, G_{q,m} = e(A, A)^{\beta \alpha_m} \\[1ex]
H_1 = k_q^{c'_{q,1}} g_p^{c_{p,1}}, \dots, H_m = k_q^{c'_{q,x}} g_p^{c_{p,m}} \\[1ex]
U_1 = u_1, \dots, U_m = u_m, U_{q,1} = u_{q,1}^\beta, \dots, U_{q,m} u_{q,m}^\beta
\end{array} \right]$$

The simulator chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all (x, y) and keys are generated as:

$$\begin{aligned} K_{x,y} &= \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, , d_{y+1}, \ldots, d_m\right) \\ &= \begin{cases} (Ag_p)^{\alpha_x} g_p^{r_{p,x} c_{p,y}} k^{r'_{q,x} c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : x = i \\ (Ag_p)^{\alpha_x} g_p^{r_{p,x} c_{p,y}} A^{r'_{q,x} c'_{p,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : x \neq i \end{cases} \end{aligned}$$

The simulator gives the adversary all keys except $K_{i,j}$. The adversary gives the query bit $\tilde{s}$ and if $\tilde{s} = 1$ the simulator gives key $K_{i,j}$ The simulator next receives a target set S and a challenge message $M \in \mathbb{G}_T$ from the adversary. It then chooses random $\delta \in Z_p$, $(v_{1,1}, v_{1,2}, v_{1,3})$,…,$(v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n$, $w_1, \ldots, w_m \in \mathbb{Z}_n$,$z'_1, \ldots, z'_m \in \mathbb{Z}_n$, and $s'_{q,i+1}, \ldots, s'_{q,m} \in \mathbb{Z}_q$.

It creates the challenge ciphertext as:

$$\begin{array}{llll} \text{if } x > i & R_x = k_q^{s'_{q,x} r_{q,x}} & \tilde{R}_x = k_q^{d s'_{q,x} r_{q,x}} & A_x = B^{s'_{q,x}} \\ & T_x = B^{s'_{q,x} \sum_{k \in S_x} \mu_{q,k}} & B_x = Me(A, B)^{\alpha_x s'_{q,x}} & \\ & & & \\ \text{if } x = i & R_x = C^{s'_{q,x}} g_p^{s_p r_{q,x}} & \tilde{R}_x = C^{d s'_{q,x}} g_p^{d s_p r_{q,x}} & A_x = T g_p^{\delta} \\ & T_x = T^{\sum_{k \in S_x} \mu_{q,k}} (\prod_{k \in S_x} u_{p,k})^{\delta} & B_x = e(g, g)^{\gamma} & \\ & & & \\ \text{if } x < i & R_x = g^{v_{x,1}} & \tilde{R}_x = h^{v_{x,1}} & A_x = g^{v_{x,2}} \\ & T_x = (\prod_{k \in S_x} u_k)^{v_{x,2}} & B_x = e(g, g)^{v_{x,3}} & \\ & & & \\ \forall y & C_y = B^{c'_{q,y}} g_p^{z'_y} h^{w'_y} & \tilde{C}_y = g^{w'_y} & \end{array}$$

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 By the assumptions above we can now bound the Adversary's advantage in distinguishing between experiments H_2 and H_3 by $2\epsilon_{DM3DH} + \epsilon_{BSD}$, 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

$\beta, r_{q,1}, \ldots, r_{q,m}, c_{q,1}, \ldots, c_{q,m} \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n$. It then sets $h_p = B$ and picks blinding factors $r'_{p,1}, \ldots, r'_{p,m}, c'_{p,1}, \ldots, c'_{p,m} \in \mathbb{Z}_p$. Finally, $u_{q,1}, \ldots, u_{q,m} \in \mathbb{G}_q$ and $u_{p,1}, \ldots, u_{p,m} \in \mathbb{G}_p$ are chosen at random, then it sets $u_1 = u_{q,1}u_{p,1}, \ldots, u_m = u_{q,m}u_{p,m}$. Additionally, random $\delta \in \mathbb{Z}_n$ and $\gamma \in \mathbb{Z}_p$ are chosen.

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

$$\left[\begin{array}{c}
g, h, V = g^{\delta} g^{\gamma}, \tilde{V} = h^{\delta} \\[1em]
E = g_q^{\beta}, E_1 = g_q^{r_{q,1}} g_p^{r'_{p,1}}, \ldots, E_m = g_q^{r_{q,m}} g_p^{r'_{p,m}}, E_{q,1} = g_q^{\beta r_{q,1}}, \ldots, E_{q,m} = g_q^{\beta r_{q,m}} \\[1em]
F_1 = h_q^{r_{q,1}} h_p^{r'_{p,1}}, \ldots, F_m = h_q^{r_{q,m}} h_p^{r'_{p,m}}, F_{q,1} = h_q^{\beta r_{q,1}}, \ldots, F_{q,m} = h_q^{\beta r_{q,m}} \\[1em]
G_1 = e(g,g)^{\alpha_1}, \ldots, G_m = e(g,g)^{\alpha_m}, G_{q,1} = e(g_q, g_q)^{\beta \alpha_1}, \ldots, G_{q,m} = e(g_q, g_q)^{\beta \alpha_m} \\[1em]
H_y = g_q^{c_{q,y}} g_p^{c'_{p,y}} : y \neq j, \quad H_j = g_q^{c_{q,j}} C^{c'_{p,j}} \\[1em]
U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^{\beta}, \ldots, U_{q,m} = u_{q,m}^{\beta}
\end{array}\right]$$

The simulator chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all (x, y) and keys are generated as:

$$\begin{aligned}
K_{x,y} &= \left(d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1},, d_{y+1}, \ldots, d_m\right) \\
&= \begin{cases} g^{\alpha_x} g_q^{r_{q,x} c_{q,y}} g_q^{r'_{p,x} c'_{q,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}},, u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : y \neq j \\ g^{\alpha_x} g_q^{r_{q,x} c_{q,y}} C^{r'_{p,x} c'_{q,y}} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}},, u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} & : y = j \end{cases}
\end{aligned}$$

The simulator gives the adversary all keys except $K_{i,j}$. The adversary gives the query bit $\tilde{s}$ and if $\tilde{s} = 1$ the simulator gives key $K_{i,j}$ The simulator next receives a target set S and a challenge message $M \in G_T$ from the adversary. The simulator then chooses exponents $(v_{1,1}, v_{1,2}, v_{1,3})$,..., $(v_{i,1}, v_{i,2}, v_{i,3}) \in \mathbb{Z}_n$, 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 $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w'_1, \ldots, w'_m \in \mathbb{Z}_n$. It then creates the ciphertext as:

$$\begin{array}{llll}
\text{if } x > i+1 & R_x = g_q^{s_{q,x} r_{q,x}} & \tilde{R}_x = h_q^{s_{q,x} r_{q,x}} & A_x = g_q^{s_{q,x} t_q} \\
 & T_x = (\prod_{k \in S_x})^{s_{q,x} t_q} & B_x = M e(g_q, g_q)^{\alpha_x s_{q,x} t_q} & \\[1.5em]
\text{if } x \leq i & R_x = g^{v_{x,1}} & \tilde{R}_x = h^{v_{x,1}} & A_x = g^{v_{x,2}} \\
 & T_x = (\prod_{k \in S_x})^{v_{x,2}} & B_x = e(g,g)^{v_{x,3}} & \\[1.5em]
\text{if } y > j & C_y = g_q^{c_{q,y} t_q} h^{w'_y} & \tilde{C}_y = A^{-c'_{p,y}} g^{w'_y} & \\
\text{if } y = j & C_y = g_q^{c_{q,y} t_q} T h^{w'_y} & \tilde{C}_y = g^{w'_y} & \\
\text{if } y < j & C_y = g_q^{c_{q,y} t_q} g_p^{z_y} h^{w'_y} & \tilde{C}_y = g^{w'_y} &
\end{array}$$

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 $m \cdot \epsilon_{DM3DH}$. By our assumption the adversary's advantage is negligible. $\square$

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 Diffie-Hellman Subgroup Decision game. In this game the simulator does not know the factors of n.

The simulator first takes in the challenge $g, h, A = g_q^a, B = h_q^a, C = g^b g_p^c, D = h^b, T$. For the simulation $a = \beta$. The simulator then chooses random exponents $r_1, \dots, r_m, c_1, \dots, c_m$,$\alpha_1$,. . .,$\alpha_m, \mu_1$, . . . , $\mu_m \in \mathbb{Z}_n$.

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

$$\left[\begin{array}{c} g, h, V = C, \tilde{V} = D \\ E_1 = g^{r_1}, \dots, E_m = g^{r_m}, E_{q,1} = A^{r_1}, \dots, E_{q,m} = A^{r_m} \\ F_1 = h^{r_1}, \dots, F_m = h^{r_m}, F_{q,1} = A^{r_1}, \dots, F_{q,m} = B^{r_m} \\ G_1 = e(g,g)^{\alpha_1}, \dots, G_m = e(g,g)^{\alpha_m}, G_{q,1} = e(A,g)^{\alpha_1}, \dots, G_{q,m} = e(A,g)^{\alpha_m} \\ H_1 = g^{c_1}, \dots, H_m = g^{c_m} \\ U_1 = g^{\mu_1}, \dots, U_m = g^{\mu_m}, U_{q,1} = A^{\mu_1}, \dots, U_{q,m} = A^{\mu_m} \end{array}\right]$$

The simulator chooses random $\sigma_{x,y} \in \mathbb{Z}_n$ for all (x, y) and keys are generated as:

$$\begin{aligned} K_{x,y} &= \left(d'_{x,y}, d''_{x,y}, d_1, \dots, d_{y-1}, , d_{y+1}, \dots, d_m\right) \\ &= g^{\alpha_x} g^{r_x c_y} u_y^{\sigma_{x,y}},\ g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \dots, u_{y-1}^{\sigma_{x,y}}, , u_{y+1}^{\sigma_{x,y}}, \dots, u_m^{\sigma_{x,y}} \end{aligned}$$

The simulator gives the adversary all keys except $K_{i,j}$. The adversary gives the query bit $\tilde{s}$ and if $\tilde{s} = 1$ the simulator gives key $K_{i,j}$ The simulator next receives a target set S and a challenge message $M \in G_T$ from the adversary. It then chooses a random $t \in \mathbb{Z}_n$, $w_1, \dots, w_m$, $s_1, \dots, s_m \in Z_n$, and $(v_{1,1}, v_{1,2}, v_{1,3})$,. . . , $(v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n$.

The challenge ciphertext for all $C_y, \tilde{C}_y$ and $R_x, \tilde{R}_x, A_x, T_x, B_x$ 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 $x > i + 1$ $R_x, \tilde{R}_x, A_x, T_x, B_x$ are created in a straightforward manner using A.

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

$R_{i+1} = T^{s_{i+1} r_{i+1}}$ $\quad \tilde{R}_{i+1} = T^{d s_{i+1} r_{i+1}}$ $\quad A_{i+1} = T^{s_{i+1} t}$

$T_{i+1} = T^{s_{i+1} t \sum_{k \in S_x} \mu_k}$ $\quad B_{i+1} = M e(T, g)^{\alpha_{i+1} s_{i+1} t}$

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

□