A Fully Collusion Resistant Broadcast, Trace, and Revoke System

Dan Boneh† Brent Waters‡
Stanford University SRI International

September 28, 2006

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 sublinear size ciphertexts and private keys that is secure against adaptive adversaries. The system is publicly traceable.

1 Introduction

A broadcast encryption system [16] 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 [11] 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 $D$. Then there is a tracing algorithm $\text{Trace}$ that takes the public key $PK$ as input and interacts with $D$ as a black-box oracle. The algorithm outputs the identity of at least one of the users who created $D$. That is, $\emptyset \neq \text{Trace}^{D}(PK) \subseteq T$. 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 systems are said to be publicly traceable.

Trace and Revoke [26, 25] 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 $D$ using the private keys of

---

*This paper is the full version of [8]
†Supported by NSF and the Packard Foundation
‡Supported by NSF
users $T \subseteq \{1, \ldots, N\}$. The tracing algorithm then interacts with $D$ and identifies one of the active keys in the pirate’s possession, namely a key of user $t \in T \cap S$. We write $\emptyset \neq \text{Trace}^D(PK, S) \subseteq T \cap S$.

The broadcaster revokes user $t$ by encrypting future broadcasts to the set $S' \leftarrow S \setminus \{t\}$. If the pirate decoder $D$ can still decrypt these broadcasts, we run the tracing algorithm $\text{Trace}^D(PK, S')$ 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 $D$ 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. Since a party performing the tracing needs no secrets, the overall system remains secure even if this party is compromised. 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

$$(\text{Setup}_{\text{ABE}}, \text{Encrypt}_{\text{ABE}}, \text{Decrypt}_{\text{ABE}})$$

The encryption algorithm $\text{Encrypt}_{\text{ABE}}(S, PK, i, M)$, 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 require that

- The output of $\text{Encrypt}_{\text{ABE}}(S, PK, N + 1, M)$ contains no information about $M$, and
- For $i \in S$ the distribution generated by algorithm $\text{Encrypt}_{\text{ABE}}(S, PK, i, M)$ is indistinguishable from the distribution generated by $\text{Encrypt}_{\text{ABE}}(S, PK, i + 1, M)$ 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 $\text{Encrypt}_{\text{ABE}}(S, PK, 1, M)$, 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, 25, 22] and was recently made explicit in [6]. The tracing system of [6], however, requires 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. Building a broadcast system secure against adaptive adversaries was left as an open problem in [4].

1.1 Related work

Broadcast encryption systems are often designed for the case when the pirate has fewer than some $t$ private keys [16, 36, 37, 1, 38, 26, 14, 19]. Several elegant constructions [25, 13, 21, 20], 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 [14], 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 [11, 35, 33, 34, 24, 27, 3, 17, 12, 30, 2, 32, 31, 38, 22, 23]. A recent system based on pairings [6] resists arbitrary collusion and has constant size private keys and $O(\sqrt{N})$ 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 provided in [28, 29, 39, 23, 10].

Several trace and revoke systems are available [26, 18, 25, 38, 14, 15, 21, 20] that are designed for broadcasting to large sets.

Table 1 summarizes the existing sub-linear size fully collusion resistant systems currently available. Here $N$ is the total number of users in the system. As usual, all the expressions in the table should be multiplied by the security parameter.

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

Table 1: Sub-linear size fully collusion resistant systems

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:

\[ \text{Setup}_{\text{ABE}}(N, \lambda) \] A probabilistic algorithm that takes as input \( N \), the number of users in the system, and a security parameter \( \lambda \). The algorithm runs in polynomial time in \( \lambda \) and outputs a public key \( PK \) and private keys \( SK_1, \ldots, SK_N \), where \( SK_u \) is given to user \( u \).

\[ \text{Encrypt}_{\text{ABE}}(S, PK, i, M) \] Takes as input a subset of users \( S \subseteq \{1, \ldots, N\} \), 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 \( S \cap \{i, \ldots, N\} \).

\[ \text{Decrypt}_{\text{ABE}}(S, j, SK_j, C, PK) \] 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 \( \perp \).

**Correctness property.** 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 \( M \):

\[
\text{Let } (PK, (SK_1, \ldots, SK_N)) \xRightarrow{R} \text{Setup}_{\text{ABE}}(N, \lambda) \text{ and } C \xRightarrow{R} \text{Encrypt}_{\text{ABE}}(S, PK, i, M).
\]

If \( j \in S \) and \( j \geq i \) then \( \text{Decrypt}_{\text{ABE}}(S, j, SK_j, C, PK) = M \).

**Security.** We define security of an ABE system using two games. The first game is a message hiding game and says that an adversary cannot break semantic security when encrypting 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, \( N \).

For simplicity, we define our games for security against an adversary that mounts a chosen-plaintext attack (CPA). We can easily extend them to handle chosen-ciphertext attacks (CCA) by giving the adversary access to a decryption oracle for each user in the system.

**Game 1.** The first game, called Message Hiding, says that an adversary cannot break semantic security when encrypting using index \( i = N + 1 \). The game proceeds as follows:

- **Setup** The challenger runs \( \text{Setup}_{\text{ABE}}(N, \lambda) \) and gives the adversary \( PK \) and all secret keys \( \{SK_1, \ldots, SK_N\} \).

- **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

\[
C \xRightarrow{R} \text{Encrypt}_{\text{ABE}}(S, PK, N + 1, M_\beta)
\]

to the adversary.

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

We define the advantage of adversary \( A \) in winning the game as \( \text{MH Adv}_A = |\Pr[\beta' = \beta] - 1/2| \).
The second game, called Index Hiding, 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 $Setup_{ABE}(N, \lambda)$ and gives the adversary $PK$ and the set of private keys $\{SK_j \text{ s.t. } j \neq i\}$

- **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 $i \not\in S$. The challenger flips a coin $\beta \in \{0, 1\}$ and sends $C \leftarrow Encrypt_{ABE}(S, PK, i + \beta, M)$ to the adversary.

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

We define the advantage of adversary $A$ as the quantity $IH\text{ Adv}_A[i] = |\Pr[\beta' = \beta] - 1/2|$. 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 $i \not\in S$ then even if everyone colludes they cannot distinguish whether $i$ or $i + 1$ was used to create $C$. Indeed, when $i \not\in S$ 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 $A$ we have that $MH\text{ Adv}_A$ and $IH\text{ Adv}_A[i]$ for $i = 1, \ldots, N$, are negligible functions of $\lambda$.

### 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 $\lambda$.

#### 2.2.1 Broadcast encryption secure against adaptive attacks

Let $E = (Setup_{ABE}, Encrypt_{ABE}, Decrypt_{ABE})$ be a secure ABE system. Define

$$Encrypt(S, PK, M) = Encrypt_{ABE}(S, PK, 1, M)$$

We show that $E_{BE} = (Setup_{ABE}, Encrypt, Decrypt_{ABE})$ 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 $N, \lambda$, the extended message hiding 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 $\text{Setup}_{\text{ABE}}(N, \lambda)$ 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 $S_0 = \{1, \ldots, N\} \setminus S_0$.

• **Challenge** The adversary outputs a set $S \subseteq S_0$ and two equal length messages $M_0, M_1$. The challenger flips a coin $\beta \in \{0, 1\}$ and sends $C_R \leftarrow \text{Encrypt}_{\text{ABE}}(S, \text{PK}, i, M_\beta)$ 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 $\beta$.

We define the advantage of adversary $A$ in winning the game as $\text{MH Adv}_A[1] = |\Pr[\beta' = \beta] - 1/2|$. The main point is that $\text{MH Adv}_A[1]$ is the same quantity used to define broadcast encryption security against adaptive attackers [14, 4] for $\mathcal{E}_{\text{BE}}$. Hence, if we prove that $\text{MH Adv}_A[1]$ is negligible then $\mathcal{E}_{\text{BE}}$ 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 $\text{MH Adv}_A[1]$ is a negligible function of $\lambda$ for any polynomial time adversary $A$.

**Proof sketch.** Suppose $\text{MH Adv}_A[1] > \epsilon$ for some adversary $A$ and non-negligible $\epsilon$. Since $\mathcal{E}$ is a secure ABE we know that $\text{MH Adv}_A$ (defined in Game 1) is negligible. It follows that $\text{MH Adv}_A[N+1]$ is negligible. For simplicity, say $\text{MH Adv}_A[N+1] = 0$. Then, by the standard hybrid argument there exists a $j \in \{1, \ldots, N\}$ such that


In other words, this $A$ is somehow able to distinguish $\text{Encrypt}_{\text{ABE}}(S, \text{PK}, j, M)$ from $\text{Encrypt}_{\text{ABE}}(S, \text{PK}, j + 1, M)$ for some $M$ and $S$. But then $A$ can be directly used to win the ABE index hiding game.

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

$$|\text{MH Adv}_A[i] - \text{MH Adv}_A[i + 1]| \leq 2 \cdot \text{IH Adv}_B[i]$$

(1)

Then


$$\sum_{i=1}^{N} |\text{MH Adv}_A[i] - \text{MH Adv}_A[i + 1]| \leq$$

$$2 \sum_{i=1}^{N} \text{IH Adv}_B[i]$$

But since $\mathcal{E}$ is a secure ABE we know that $\text{MH Adv}_A[N + 1]$ and $\text{IH Adv}_B[i]$ for $i = 1, \ldots, N$ are negligible for any polynomial time $A$. Therefore, $\text{MH Adv}_A[1]$ is negligible, as required. \qed
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 $E_B$ above. The tracing algorithm uses a general tracing method, previously used in [3, 25, 22, 6]. We use the notation from Appendix A. For a given $\epsilon > 0$ and a set $S_D$ the tracing algorithm $Trace^D(S_D, PK, \epsilon)$ 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 steps $8\lambda(N/\epsilon)^2$ times:
      
      i. Sample $M$ from the finite message space at random.
      ii. Let $C \leftarrow Encrypt_{ABE}(S_D, PK, i, M)$.
      iii. Call oracle $D$ on input $C$, and compare the output of $D$ to $M$.
   
   (b) Let $\hat{p}_i$ be the fraction of times that $D$ 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 $E_{TR} = (Setup_{ABE}, Encrypt, Decrypt_{ABE}, Trace)$ be the resulting trace and revoke system. Note $E_{TR}$ is just the broadcast system $E_B$ with the tracing algorithm $Trace$. We show that $E_{TR}$ is secure in the sense of Definition A.1, namely fully collusion resistant against an adaptive adversary.

**Theorem 2.3.** If $E$ is a secure ABE then for $E_{TR}$ the quantity $TAdv_A$ defined in Appendix A.1 is negligible.

**Proof sketch.** Let $(D, S_D)$ be the pirate decoder and the recipient set output by the adversary. Define

$$p_i = \Pr[D(Encrypt_{ABE}(S_D, PK, i, M)) = M]$$

We know that $p_1 \geq \epsilon$ and $p_{N+1}$ is negligible. The former follows from the fact that $D$ is a useful decoder. The latter 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 $Trace^D(S_D, PK, \epsilon)$ is non-empty.

It remains to show that whenever $\hat{p}_j - \hat{p}_{j+1} > \epsilon/(4N)$ we have that $j \in S_D \cap U$. 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 $j \in S_D$. Clearly $j \in S_D$ since otherwise, even given all the secret keys, there is no hope of distinguishing $p_j$ from $p_{j+1}$. But if $j \in S_D$ and $j \notin U$ then $D$ must distinguish $p_j$ from $p_{j+1}$ without the key $SK_j$. Again, such a $D$ can be directly used to win the ABE index hiding game. Hence, $j \in S_D \cap U$. \qed
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 $G$ be an algorithm called a group generator that takes as input a security parameter $\lambda \in \mathbb{Z}_{>0}$ and outputs a tuple $(p, q, G, G_T, e)$ where $p, q$ are two distinct primes, $G$ and $G_T$ are two cyclic groups of order $n = pq$, and $e$ is a function $e : G^2 \to G_T$ satisfying the following properties:

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

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

To summarize, $G$ outputs the description of a group $G$ of order $n = pq$ with an efficiently computable bilinear map. We will use the notation $G_p, G_q$ to denote the respective subgroups of order $p$ and order $q$ of $G$.

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 $G_p$ and the last two are over the composite order group $G$.

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

$(p, q, G, G_T, e) \xleftarrow{\$} G(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{\$} G_p, \quad a, b, c \xleftarrow{\$} \mathbb{Z}_p$

$\bar{Z} \leftarrow \left(n, G, G_T, e), \quad g_p^a, \quad g_p^b, \quad g_p^{(b^2)}\right)$

$T \leftarrow g_p^{abc}$

Output $(\bar{Z}, T)$

For an algorithm $A$, define $A$’s advantage in solving the decision 3-party Diffie-Hellman problem for $G$ as:

$D3DH_{G,A}(\lambda) := \left| \Pr[A(\bar{Z}, T) = 1] - \Pr[A(\bar{Z}, R) = 1] \right|$

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

8
Definition 3.1. We say that $G$ satisfies the decision (modified) 3-party Diffie-Hellman assumption (D3DH) if for any polynomial time algorithm $A$ we have that $\text{D3DH Adv}_{G,A}(\lambda)$ is a negligible function of $\lambda$.

The assumption is a little stronger than the corresponding assumption in [6] since we 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 $G_q$ is indistinguishable from a random element in $G$, even when an ElGamal encryption of a $G_p$ element is provided. More precisely, for a given group generator $G$ define the following distribution $P(\lambda)$:

$$(p, q, G, G_T, e) \xleftarrow{\mathcal{R}} G(\lambda), \quad n \leftarrow pq$$
$$g, h \xleftarrow{\mathcal{R}} G, \quad v_p \xleftarrow{\mathcal{R}} G_p$$
$$a \xleftarrow{\mathcal{R}} Z_q, \quad b \xleftarrow{\mathcal{R}} Z_n,$$
$$\bar{Z} \leftarrow ((n, G, G_T, e), g, h, g^b v_p, h^b, g^{pa}, h^{pa})$$
Output $\bar{Z}$

For an algorithm $A$, define $A$’s advantage in solving the Diffie-Hellman Subgroup Decision problem for $G$ as:

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

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

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

The Diffie-Hellman subgroup decision assumption is a little 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 $G_p$. Furthermore, the adversary is given $g^{pa}, h^{pa} \in G_q$.

**Bilinear Subgroup Decision Assumption.** The Bilinear Subgroup Decision (BSD) assumption states that a random order $p$ element in $G_T$ is indistinguishable from a random element in $G_T$ when $g_p, g_q \in G$ are given. More precisely, for a given group generator $G$ define the following distribution $P(\lambda)$:

$$(p, q, G, G_T, e) \xleftarrow{\mathcal{R}} G(\lambda), \quad n \leftarrow pq, \quad g_p \xleftarrow{\mathcal{R}} G_p, \quad g_q \xleftarrow{\mathcal{R}} G_q,$$
$$\bar{Z} \leftarrow ((n, G, G_T, e), g_p, g_q)$$
Output $\bar{Z}$

Define $A$’s advantage in solving the bilinear subgroup decision problem for $G$ as:

$$\text{BSD Adv}_{G,A}(\lambda) := \left| \Pr[A(\bar{Z}, e(T, g)) = 1] - \Pr[A(\bar{Z}, e(R, g)) = 1] \right|$$
where \( \bar{Z} \leftarrow P(\lambda), \ T \leftarrow G_p, \) and \( R \leftarrow \mathbb{G}. \) Here \( g \) is an arbitrary generator of \( \mathbb{G}. \)

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

## 4 An Efficient Augmented Broadcast Encryption System

We construct 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].

### 4.1 Difficulty of Achieving Trace and Revoke

A trace and revoke system cannot be constructed by naively combining a broadcast encryption system and a tracing system. 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, if 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!

### 4.2 Our Approach

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 construct 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 for each user to prevent such attacks. The more straightforward combination of [4] and [6] results in a scheme that is insecure since a revoked user can still break the “privacy” property and prevent other colluders from being traced, although we do not show this here. A consequence of using randomized keys is that each user must have \( O(\sqrt{N}) \) size private key storage.

The other primary 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 \( G_p \) subgroup, while kept well-formed in the \( G_q \) subgroup. The most natural way to do this is to give an element of \( G_p \) as part of the
public parameters. However, giving out such an element allows an attacker to break the scheme. In our construction we construct the parameters in such a way that allows for this type of public encryption without giving out an element of $\mathbb{G}_p$.

### 4.3 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 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.

### 4.4 ABE 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:

\[
\text{Setup}_{\text{ABE}}(N = m^2, \lambda).
\]

The setup algorithm takes as input the number of users $N$ and a security parameter $\lambda$. 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

\[
\delta, r_1, \ldots, r_m, \quad c_1, \ldots, c_m, \quad \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n
\]

\[
\beta \in \mathbb{Z}_q, \quad \gamma \in \mathbb{Z}_p
\]
The public key PK includes the description of the group and the following elements:

\[
\begin{bmatrix}
g, h, \tilde{V} = g^\delta g^\gamma, V = h^\delta \\
E_q = g^\beta, E_1 = g^\beta_1, \ldots, E_m = g^{\beta_m}, \\
E_q,1 = g^{\beta_1}, \ldots, E_q,m = g^{\beta_m}, \\
F_1 = h^{r_1}, \ldots, F_m = h^{r_m}, \\
F_q,1 = h^{q_1}, \ldots, F_q,m = h^{q_m}, \\
G_1 = e(g,g)^{\alpha_1}, \ldots, G_m = e(g,g)^{\alpha_m}, \\
G_q,1 = e(g_q,g_q)^{\beta_1}, \ldots, G_q,m = e(g_q,g_q)^{\beta_m}, \\
H_1 = g^{c_1}, \ldots, H_m = g^{c_m}, \\
U_1 = u_1, \ldots, U_m = u_m, \\
U_q,1 = u_{q,1}, \ldots, U_q,m = u_{q,m}
\end{bmatrix}
\]

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:

\[
\text{SK}_{x,y} = (d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, d_{y+1}, \ldots, d_m) = \\
= \left( g^{\alpha_1} g^{\sigma_{x,y}} u_y^{\sigma_{x,y}}, g^{\sigma_{x,y}}, u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, u_{y+1}^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}} \right)
\]

The public parameters \(u_{q,1}^{\beta}, \ldots, u_{q,m}^{\beta}\) are related to the broadcast portion 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 \(g^{\alpha_1}\) blinded by \(g^{\sigma_{x,y}}\), which is related to traitor tracing and \(u_y^{\sigma_{x,y}}\), which is related to the broadcast encryption system. An important technical point is that since \(d'_{x,y}\) contains both pieces multiplied together, 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.

\(\text{Encrypt}_{\text{AHE}}(S, PK, (i, j), M)\).

The \(\text{Encrypt}_{\text{AHE}}\) 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

\[
t, \kappa, w_1, \ldots, w_m, s_1, \ldots, s_m \in \mathbb{Z}_n
\]

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

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

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:
if $x > i$: 
\[ R_x = E_{q,x}^{s_x} \]
\[ T_x = \left( \prod_{k \in S_x} U_{q,k} \right)^{s_x t} \]
\[ B_x = MG_{q,x}^{s_x t} \]
\[ A_x = E_{q,t}^{s_x} \]

if $x = i$: 
\[ R_x = E_{x}^{s_x} \]
\[ T_x = \left( \prod_{k \in S_x} U_{k} \right)^{s_x t} \]
\[ B_x = MG_{x}^{s_x t} \]
\[ A_x = g^{s_x t} \]

if $x < i$: 
\[ R_x = g^{\nu_x,1} \]
\[ T_x = \left( \prod_{k \in S_x} U_{k} \right)^{\nu_x,2} \]
\[ B_x = e(g, g)^{\nu_x,3} \]
\[ A_x = g^{\nu_x,2} \]

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

if $y \geq j$: 
\[ C_y = H_{q,y}^{t} \nu_{w_y} \]
\[ \tilde{C}_y = g^{w_y} \]

if $y < j$: 
\[ C_y = H_{q,y}^{t} \nu_{w_y}^{\nu_{s_b}} \]
\[ \tilde{C}_y = g^{w_y} V^{b_y} \]

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

\[ \left( (R_x, \tilde{R}_x, T_x, A_x, B_x)^m_{x=1}, (C_y, \tilde{C}_y)^m_{y=1} \right) \]

The $T_x$ values can be viewed as 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.

$Decrypt_{ABE}(S, (x, y), SK_{x,y}, C, PK)$.

If user $(x, y) \in S$ it can attempt to decrypt by first computing a temporary key

\[ K'_{x,y} = d_{x,y}^{t} \prod_{k \in S_x}_{k \neq y} d_{x,y,k} \]

Then it computes:

\[ B_x / \left( e(K'_{x,y}, A_x) e(\tilde{R}_x, \tilde{C}_y) / \left( e(R_x, C_y) e(T_x, d_{x,y}^{t}) \right) \right) \]

Suppose that the ciphertext was encrypted to index $(i, j)$ and that $x > i$ then in decryption, the pairing $e(K'_{x,y}, A_x)$ gives the value

\[ e(g, g_q)^{\alpha_x s_x t} e(g, \prod_{k \in S_x} U_{q,k})^{s_x t} e(g, g_q)^{s_x t \nu_x c_y} \]

The other pairings are used to divide out

\[ e(g, \prod_{k \in S_x} U_{q,k})^{s_x t \nu_x c_y} e(g, g_q)^{s_x t c_y} \]

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

We prove security of our Augmented Broadcast Encryption system by showing that it is secure under both games defined in Section 2. The proof structure 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 \(M_\beta\) to set \(S\) and index \(N + 1\). Since, all \(B_x\) values contain no information about the ciphertext the bit \(\beta\) 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]. The main security theorem is as follows.

**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 \(A\) can win the Index-Hiding game with non-negligible advantage.

We prove the theorem with a sequence of lemmas and claims. First we consider the case where the adversary \(A\) chooses to distinguish between an encryption to indices \((i, j)\) and \((i, j + 1)\) where \(j < m\). We state the following lemma whose proof is given in Appendix C.1.

**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 an encryption to \((i, j + 1)\) in the Index Hiding game with non-negligible advantage.

In this game we build a simulator that will guess 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.

Next we consider the case when the adversary \(A\) attempts to distinguish between an encryption to \((i, m)\) and one to \((i + 1, 1)\) for some \(1 \leq i < m\). We refer to the rows with ciphertexts in the \(G_q\) subgroup as “greater than” rows and the the row with well formed ciphertexts in \(G\) 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 $G_p$ subgroup for all $y \geq j$. We state our lemma and then discuss its proof.

**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 $A$ can distinguish between an encryption to $(i, m)$ and an encryption to $(i+1, 1)$ in the Index Hiding game with non-negligible advantage.

To prove the lemma 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.

The following claims, whose proofs are given in Appendix C.2 through Appendix C.4, show that all these hybrid games are indistinguishable. The proof of the first claim, Claim 5.4, is identical to the proof of Lemma 5.2.

**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.

**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.

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

**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.

Lemma 5.3 follows by summing the maximum adversarial advantages across the hybrid experiments. Theorem 5.1 follows from Lemma 5.2 and 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 [9] for security against CCA attacks. We hope that future research will lead to ABEs with even shorter ciphertexts and private keys.

References


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 and tracing requires no secrets. We refer to [6] for more information on the definition of traitor tracing systems. A TR system consists of four algorithms:

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

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


**Decrypt**$(S, j, SK_j, C, PK)$ 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 $\bot$.

**Trace**$^D(S, PK, \epsilon)$ The tracing algorithm is an oracle algorithm that interacts with a pirate decoder $D$. The algorithm is given as input a set $S \subseteq \{1, \ldots, N\}$, the public key $PK$ and a parameter $\epsilon$, and runs in time polynomial in the security parameter $\lambda$ and $1/\epsilon$. Only values of $\epsilon$ that are polynomially related to $\lambda$ are considered valid inputs to $Trace$. The tracing algorithm outputs a set $T \subseteq \{1, \ldots, N\}$.

**Correctness.** 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 $(PK, (SK_1, \ldots, SK_N)) \xleftarrow{\xi} Setup_{ABE}(N, \Lambda)$ and $c \xleftarrow{\xi} Encrypt(S, PK, M)$.

If $j \in S$ then $Decrypt(S, j, SK_j, c, PK) = M$.

### A.1 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 $[14, 4]$. We described the game in Section 2.2.1. We let $\text{MHAdv}_A[1]$ denote the advantage of $A$ 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 $D$ that will decrypt all ciphertexts encrypted for a certain set $S_D$. The tracing algorithm’s goal is extract from $D$ at least one of the keys $u \in S_D$ that were used to construct $D$. The broadcaster will encrypt all future messages to the set $S' = S_D \setminus \{u\}$. If the decoder $D$ 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 $S_D$. This process continues, iteratively shrinking $S'$, until $D$ stops functioning.

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

1. The challenger runs $Setup(N, \lambda)$ to obtain $PK$ and private keys $SK_1, \ldots, SK_N$. It provides $PK$ to $A$.
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 $A$ outputs a set $S_D \subseteq \{1, \ldots, N\}$ and a pirate decoder $D$ which is a probabilistic circuit that takes as input ciphertexts in $C$ and outputs some message $M$.
4. The challenger now runs $Trace^D(S_D, PK, \epsilon)$ to obtain a set $T \subseteq \{1, \ldots, N\}$.

We say that the adversary $A$ wins the game if the following two conditions hold:
• For a randomly chosen \( M \) in the finite message space, we have that
\[
\Pr[D(\text{Encrypt}(S_D, PK, M)) = M] \geq \epsilon
\]
A pirate decoder satisfying this condition is said to be a useful decoder.
• The set \( T \) is either empty, or is not a subset of \( U \cap S_D \).

We denote by \( \text{TR Adv}_A \) the probability that adversary \( A \) 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 \( \epsilon \). 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 [22].

**Definition A.1.** We say that an \( N \)-user Trace and Revoke system is secure if for all polynomial time adversaries \( A \) we have that \( \text{MH Adv}_A[1] \) and \( \text{TR Adv}_A \) are negligible functions of \( \lambda \).

## 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 \( B \) plays the index hiding game with input \( i \) and works as follows:

• **Setup** \( B \) receives PK and the set of private keys \( \{\text{SK}_j \text{ s.t. } j \neq i\} \) from its challenger. \( B \) runs adversary \( A \) and gives it PK.

• **Query** \( A \) issues adaptive private key queries. To respond to a query for \( \text{SK}_j \) adversary \( B \) does:
  - If \( j \neq i \) then \( B \) simply gives \( \text{SK}_j \) to \( B \).
  - Otherwise, \( j = i \). Then \( B \) sends the bit \( \tilde{s} = 1 \) to its challenger and receives \( \text{SK}_i \) in return. It gives \( \text{SK}_i \) to \( A \).

Let \( S_0 \subseteq \{1, \ldots, N\} \) denote the set of private keys requested by \( A \). Define the complement as \( \overline{S_0} = \{1, \ldots, N\} \setminus S_0 \).

• **Challenge** Finally, \( A \) outputs a set \( S \subseteq \overline{S_0} \) and two equal length messages \( M_0, M_1 \). Then \( B \) flips a coin \( \gamma \in \{0, 1\} \) and gives \( S \) and \( M_{\gamma} \) to its challenger. Note that if \( B \) sent \( \tilde{s} = 1 \) to its challenger then \( i \not\in S \), as required.

\( B \) receives back \( C \leftarrow \text{Encrypt}_{\text{ABE}}(S, PK, i + \beta, M_{\gamma}) \) for some random \( \beta \in \{0, 1\} \). It gives \( C \) to \( A \).

• **Guess** Finally, \( A \) outputs a guess \( \gamma' \) for \( \gamma \). If \( \gamma' = \gamma \) then \( B \) outputs 0. Otherwise, \( B \) outputs 1.

Now, observe that when \( \beta = 0 \) then \( B \) is emulating perfectly an \( \text{MH Adv}_A[i] \) challenger. When \( \beta = 1 \) then \( B \) is emulating perfectly an \( \text{MH Adv}_A[i+1] \) challenger. A standard argument now shows that \( |\text{MH Adv}_A[i] - \text{MH Adv}_A[i+1]| \leq 2 \cdot \text{IH Adv}_B[i] \) 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 \(A\) that breaks the Index Hiding game with advantage \(\epsilon\). We build a simulator as follows. The simulator receives the (Modified) 3-party Diffie-Hellman challenge from the simulator as:

\[
\begin{align*}
    g_p, A = g_p^a, B = g_p^b, C = g_p^c, D = g_p^{b^2}, T.
\end{align*}
\]

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 \(u_{p,1} = g_p^{\alpha_1}, \ldots, u_{p,m} = g_p^{\alpha_m}\) 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 \(g = \)
$$g_qg_p, h = h_qB,$$ and publishes

$$g = g_qg_p, h = h_qB, V = g^\delta g_p^\gamma, \bar{V} = h^\delta$$

$$E_q = g_{q}^\beta, E_x = g_{q}^{r_{x,q}}g_{p,x}^{r_{x,p},x} : x \neq i, \quad E_i = g_{q}^{r_{i,q}}B_{i,p,i}$$

$$E_{q,1} = g_{q}^{\beta_{q,1}}, \ldots, E_{q,m} = g_{q}^{\beta_{q,m}}$$

$$F_x = g_{q}^{r_{x,q}}B_{x,p,x}^{r_{x,p},x} : x \neq i, \quad F_i = g_{q}^{r_{i,q}}D_{i,p,i}$$

$$F_{q,1} = h_{q}^{\beta_{q,1}}, \ldots, F_{q,m} = h_{q}^{\beta_{q,m}}$$

$$G_1 = e(g, g)^{\alpha_1}, \ldots, G_m = e(g, g)^{\alpha_m}, G_{q,1} = e(g_q, g_q)^{\alpha_1}, \ldots, G_{q,m} = e(g_q, g_q)^{\alpha_m}$$

$$H_y = g_{q}^{c_{y,q}^{x,y} g_{p,y}^{c_{y,p,y}} : y \neq j, \quad H_j = g_{q}^{c_{j,q}^{y} C_{p,j}^{y}}$$

$$U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^\beta, \ldots, U_{q,m} = u_{q,m}^\beta$$

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:

$$K_{x,y} = (d_{x,y}^p, d_{x,y}^q, d_1, \ldots, d_{y-1}, d_{y+1}, \ldots, d_m)$$

$$= \begin{cases} (g^{\alpha_x}g_{q}^{r_{x,q} c_{q,y}^{x,y}} g_{p,x}^{r_{x,p} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, g_{p,x}^{r_{p,x} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, \ldots, u_{q,y-1}^{\sigma_{x,y}}, u_{q,y+1}^{\sigma_{x,y}}, \ldots, u_{q,m}^{\sigma_{x,y}}) : x \neq i, y \neq j \\ (g^{\alpha_x}g_{q}^{r_{x,q} c_{q,y}^{x,y}}, g_{p,x}^{r_{p,x} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, g_{p,x}^{r_{p,x} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, \ldots, u_{q,y-1}^{\sigma_{x,y}}, u_{q,y+1}^{\sigma_{x,y}}, \ldots, u_{q,m}^{\sigma_{x,y}}) : x = i, y \neq j \\ (g^{\alpha_x}g_{q}^{r_{x,q} c_{q,y}^{x,y}} C_{p,x}^{y} g_{p,x}^{r_{p,x} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, g_{p,x}^{r_{p,x} c_{p,y}^{x,y}} u_{q,y}^{\sigma_{x,y}}, \ldots, u_{q,y-1}^{\sigma_{x,y}}, u_{q,y+1}^{\sigma_{x,y}}, \ldots, u_{q,m}^{\sigma_{x,y}}) : x \neq i, y = j \\ \end{cases}$$

We note that all the simulator creates public and simulator 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 $s_{q,i}, \ldots, s_{q,m} \in \mathbb{Z}_q$, $t_q \in \mathbb{Z}_q$, and $\kappa \in \mathbb{Z}_n$. 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 \( x > i \) \[ R_x = g^{s_{q,x}r_{q,x}} \quad T_x = (\prod_{k \in S_x} u_{q,k})^s_{q,x}t_q \]
\[ \hat{R}_x = h^{s_{q,x}r_{q,x}} \quad B_x = M(e(g_q, g_q)^{\alpha_x s_{q,x}t_q}) \]
\[ A_x = g^{s_{q,x}t_q} \]

if \( x = i \) \[ R_x = g^{s_{q,x}r_{q,x}} \quad T_x = (\prod_{k \in S_x} u_{q,k})^s_{q,x}t_q A_{p,q}^{\mu_x s_{q,x}t_q} \]
\[ \hat{R}_x = h^{s_{q,x}r_{q,x}} \quad B_x = M(e(g_q, g_q)^{\alpha_x s_{q,x}t_q}e(g_p, A)^{\alpha_x s_{p}}) \]
\[ A_x = g^{s_{q,x}t_q}A_{p}^{\alpha_x s_{p}} \]

if \( x < i \) \[ R_x = g^{v_{x,1}} \quad T_x = (\prod_{k \in S_x} u_{k})^{v_{x,2}} \]
\[ \hat{R}_x = h^{v_{x,1}} \quad B_x = e(g,g)^{v_{x,3}} \]
\[ A_x = g^{v_{x,2}} \]

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 \( \gamma \) from \( A \) 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 \( A \)’s advantage.

**Case II**

The simulator 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 \( \delta, \alpha_1, \ldots, \alpha_m \in \mathbb{Z}_n \). 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 \( u_{p,1} = g_{p,1}^{\mu_{p,1}}, u_{p,j-1} = g_{p, j-1}^{\mu_{p, j-1}}, \ldots, u_{p,m} = g_{p, m}^{\mu_{p, m}} \). Then it sets \( u_{p,j} = B_{p,i}^{r_{p,i} g_{p,j}} \). 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

\[
\begin{align*}
g &= g_q g_p, h = h_q B, V = g^\delta g_q^\gamma, \bar{V} = h^5 \\
E_q &= g_q^\beta, E_x = g_q^{r_{q,x}} g_p^{r_{p,x}} : x \neq i, \quad E_i = g_q^{r_{q,i}} B_{p,i}^{r_{p,i}} \\
E_{q,1} &= g_q^{\beta_{r_{q,1}}}, \ldots, E_{q,m} = g_q^{\beta_{r_{q,m}}} \\
E_{x,1} &= g_{p}^{\beta_{r_{p,1}}}, \ldots, E_{x,m} = g_{p}^{\beta_{r_{p,m}}} \\
F_x &= g_q^{r_{q,x}} B_{p,x}^{r_{p,x}} : x \neq i, \quad F_i = g_q^{r_{q,i}} B_{p,i}^{r_{p,i}} \\
F_{q,1} &= h^{\beta_{r_{q,1}}}, \ldots, F_{q,m} = h^{\beta_{r_{q,m}}} \\
G_1 &= e(g, g)^{\alpha_1}, \ldots, G_m = e(g, g)^{\alpha_m}, G_{q,1} = e(g_q, g_q)^{\beta_{r_{q,1}}}, \ldots, G_{q,m} = e(g_q, g_q)^{\beta_{r_{q,m}}} \\
H_y &= g_q^{c_{q,y}} g_p^{r_{p,y}} : y \neq j, \quad H_3 = g_q^{c_{q,3}} C_{p,3} \\
U_1 &= u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}, \ldots, U_{q,m} = u_{q,m}
\end{align*}
\]
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:

$$K_{x,y} = (d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, d_{y+1}, \ldots, d_m)$$

$$= \begin{cases} 
(g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, \ldots, u_{y_{x,y}}, u_{y_{x,y}}, \ldots, u_{m_{x,y}}) : x \neq i, y \neq j \\
(g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, \ldots, u_{y_{x,y}}, u_{y_{x,y}}, \ldots, u_{m_{x,y}}) : x = i, y \neq j \\
(g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{y_{x,y}}, \ldots, u_{y_{x,y}}, u_{y_{x,y}}, \ldots, u_{m_{x,y}}) : x = i, y = j \\
(g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{x,y}, g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{x,y}, \ldots, u_{x,y}, u_{x,y}, \ldots, u_{m_{x,y}}) : x = j, y = i \\
(g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{x,y}, g_{qy} \gamma_{x,y,q} g_{yp} \gamma_{p,x} u_{x,y}, \ldots, u_{x,y}, u_{x,y}, \ldots, u_{m_{x,y}}) : x = j, y = j 
\end{cases}$$

For the computation of key $K_{i,j}$ the simulator implicitly uses a value of $\sigma_{p,i,j} = -c'_{p,i,j} + \sigma'_{p,i,j}$. 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 $s$, if $s = 0$ 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 $M \in \mathbb{C}_T$. The simulator then chooses exponents $(v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{t-1,1}, v_{t-1,2}, v_{t-1,3}) \in \mathbb{Z}_n^3$ and exponents $s_{q,i}, \ldots, s_{q,m} \in \mathbb{Z}_q$, $t_q \in \mathbb{Z}_q$, and $k \in \mathbb{Z}_q$. 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:

$$\begin{align*}
\text{if } x > i & \quad R_x = g_{qy}^{s_{q,x,y}} s_{q,x,y} A_x = g_{qy}^{s_{q,x,y}} \\
& \quad T_x = (\prod_{k \in E_x} u_{q,k})^{s_{q,x,y}}. \\
\text{if } x = i & \quad R_x = g_{qy}^{s_{q,x,y}} s_{q,x,y} B_x = M(g_q, g) \gamma_{x,y}^{s_{q,x,y}}. \\
& \quad T_x = (\prod_{k \in S_x} u_{q,k})^{s_{q,x,y}} A_x = g_{qy}^{s_{q,x,y}} A^{s_{q,x,y}}. \\
\text{if } x < i & \quad R_x = g_{qy}^{v_{x,1}} s_{q,x,y} A_x = g_{qy}^{v_{x,1}}. \\
& \quad T_x = (\prod_{k \in E_x} u_{q,k})^{v_{x,2}}. \\
\text{if } y > j & \quad C_y = g_{qy}^{c_{q,y} w_{y}} h^{w_{y}} \quad \hat{C}_y = A^{c_{q,y} w_{y}} h^{w_{y}}. \\
\text{if } y = j & \quad C_y = g_{qy}^{c_{q,y} w_{y}} T h^{w_{y}} \quad \hat{C}_y = g^{w_{y}}. \\
\text{if } y < j & \quad C_y = g_{qy}^{c_{q,y} w_{y}} h^{w_{y}} \quad \hat{C}_y = g^{w_{y}}. 
\end{align*}$$

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 $\gamma$ from $A$ 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 $A$’s advantage, which by our assumption must be negligible. □

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 $G_T$.

Consider an adversary $A$ that distinguishes between the two experiments with probability $\epsilon$. We construct a simulator that plays the Decisional DBDH game with advantage $\epsilon$.

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 $g_{p,i} = B, g_{p,i}^{\beta} = g_p^{-b}, g_{p,i}^c = C$, and $g_{p,i}^{\beta c} = A^{-1} g_{p,i}^{\beta c}$ 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 $g_p^T$ only appears in the term $A_1$.

The simulator chooses $g_q \in \mathbb{G}, d \in \mathbb{Z}_m$ 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 $\beta, c_{q,1}, \ldots, c_{q,m} \in \mathbb{Z}_q, r_{q,i} \in \mathbb{Z}_q, r_{i-1}, r_{i+1}, \ldots, r_m \in \mathbb{Z}_n$, $\alpha_{q,i} \in \mathbb{Z}_q$, and $\alpha_{1,}, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_m \in \mathbb{Z}_n$. 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 $u_{p,1} = g_{p,i}^{u_{q,1}}, \ldots, u_{p,m} = g_{p,i}^{u_{q,m}}$. It finally assigns $u_i = u_{q,i} u_{p,i}$ for $i = 1, \ldots, m$ and chooses random $\delta \in \mathbb{Z}_m$ and $\gamma \in \mathbb{Z}_p$.

The public parameters are published as:

$$
\begin{align*}
&g, h, V = g^\delta g^\gamma, \tilde{V} = h^\delta \\
&E_q = g_q^\beta, E_x = g^{r_x} : x \neq i, \quad E_i = g_q^{r_q,i}, B \\
&E_{q,x} = g_q^{\beta r_x} : x \neq i, \quad E_{q,i} = g_q^{\beta r_q,i} \\
&F_x = g^{d r_x} : x \neq i, \quad F_i = g_q^{d r_q,i} B \\
&F_{q,x} = g_q^{\beta d r_x} : x \neq i, \quad F_{q,i} = g_q^{\beta d r_q,i} \\
&G_x = e(g_q^a, g)^{\alpha_x} : x \neq i, \quad G_i = e(g_q^a, g)^{\alpha_q,i} e(A, B) \\
&G_{q,x} = e(g_q^a, g_q)^{\beta \alpha_x} : x \neq i, \quad G_{q,i} = e(g_q^a, g_q)^{\beta \alpha_q,i} \\
&H_1 = g^{\gamma A^{-1}} g_{p,1}^{\gamma} \ldots, H_m = g^{\gamma A^{-1}} g_{p,m}^{\gamma} \\
&U_1 = u_1, \ldots, U_m = u_m, U_{q,1} = u_{q,1}^\beta, U_{q,m} = u_{q,m}^\beta
\end{align*}
$$
The simulator chooses random \( \sigma_{x,y} \in \mathbb{Z}_n \) for all \((x,y)\) and the private keys are created as:

\[
K_{x,y} = (d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, d_y, 1, \ldots, d_m)
\]

\[
= \begin{cases} 
  g^{\sigma_{x,y}} (g_{q,y} A^{-1} g^{a_y} r_x g^{\sigma_{x,y}} u_{x,y}^1, \ldots, u_{y-1}^{\sigma_{x,y}}, u_y^{\sigma_{x,y}}, \ldots, u_m^{\sigma_{x,y}}) & : x \neq i \\
  (g^{\alpha_i} g_q^{r_i c_y g}) B^{r_i y} u_{x,y}^1, g^{\sigma_{x,y}} u_1^{\sigma_{x,y}}, \ldots, u_{y-1}^{\sigma_{x,y}}, u_y^{\sigma_{x,y}} & : x = i
\end{cases}
\]

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 \( t_q \in \mathbb{Z}_q \), \( \kappa, s_1, \ldots, s_m, w_1, \ldots, w_m, (v_{x,1}, v_{x,2}, v_{x,3}), \ldots, (v_{x,1}, v_{x,2}, v_{x,3}) \in \mathbb{Z}_n \) for itself. The simulator can now create all \( \mathbb{C}_y, \mathbb{C}'_y \) values in a straightforward manner since the \( \mathbb{G}_p \) subgroup components are random. Similarly, all \( R_x, R_x, A_x, T_x, B_z \) values for \( x < i \) are just created randomly and all \( R_x, R_x, A_x, T_x, B_z \) values for \( x > i \) can be created by the simulators knowledge since they only draw from the \( \mathbb{G}_q \) subgroup components which it knows.

Finally, it creates the target row ciphertext as:

\[
R_x = (g^{q,i} B)^{s_i}, \quad T_x = \prod_{k \in S_x} (u_k^{q,i} C^{\mu_k})^{s_i}, \quad \tilde{R}_x = (g^{q,i} B)^{s_i}, \quad A_x = (g^{q,i} C)^{s_i}, \quad B_x = M (e(g, g)^{\alpha_i \tilde{t}_5} T)^{s_i}
\]

If \( T \) is a tuple \( e(g, g)^{\alpha b c} \) then we are in experiment \( H_2 \), otherwise if it is random we are in experiment \( H_{2a} \). The simulator can then repeat the adversary’s guess and play the DBDH game with advantage \( \epsilon \). It follows from our assumption \( \epsilon \) 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 \( \epsilon \) then we can build a simulator that can play the Bilinear Subgroup Decision game with advantage \( \epsilon \).

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 \( T \in \mathbb{G}_T \), then we are in hybrid experiment \( H_{2a} \). Otherwise if \( T \in \mathbb{G}_p \), 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 \( \epsilon \). 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 \( \epsilon \) then we can build a simulator that can play the Decision 3-Party Diffie-Hellman game with advantage \( \epsilon \).

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 \( d_i \in \mathbb{Z}_n, \beta \in \mathbb{G}_q \), and \( g_p \in \mathbb{G}_p \). It then sets \( g = g_p A, h = (g_p A)^d \). Next, it chooses the secrets for the \( \mathbb{G}_p \) subgroup: \( r_{p,1}, \ldots, r_{p,m}, c_{p,1}, \ldots, c_{p,m}, \alpha_{p,1}, \alpha_{p,m} \in \mathbb{Z}_p \). Then, it chooses \( r_{q,1}', \ldots, r_{q,m}', c_{q,1}', \ldots, c_{q,m}', \alpha_{q,1}', \alpha_{q,m}' \in \mathbb{Z}_q \). Finally, it chooses \( \mu_{q,1}, \ldots, \mu_{q,m} \in \mathbb{Z}_p \) at random and chooses random \( u_{p,1}, \ldots, u_{p,m} \in \mathbb{G}_p \). It then sets \( u_{q,1} = A^{\mu_{q,1}}, \ldots, u_{q,m} = A^{\mu_{q,m}} \). 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 \( \gamma \in \mathbb{Z}_p \).

26
The simulator can now publish the parameters as:

\[
\begin{bmatrix}
g, h, V = g^δ g^\gamma, \tilde{V} = h^δ \\
E_q = k_0^β, E_x = A'^x m g^e p, x : x \neq i, \quad E_{q,x} = k_0^β g^e p, i \\
F_x = A'^x m g^d p, x : x \neq i, \quad F_1 = k_0^β g^d p, i \\
G_1 = e(g_p, g_p)^ω e(A, A)^α_1, \ldots, G_m = e(g_p, g_p)^ω e(A, A)^α_m, \\
H_1 = k_0^β g^e p, 1, \ldots, H_m = k_0^β g^e p, m \\
U_1 = u_1, \ldots, U_m = u_m, U_q, 1 = u_β q, 1, \ldots, U_q, m = u_β q, m
\end{bmatrix}
\]

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

\[
K_{x,y} = \left( d'_x, y, d''_x, y, d_1, \ldots, d_y-1, d_y+1, \ldots, d_m \right)
\]

\[
= \left\{ (Ag_p)^α g^r y, g^y, r u_1, \ldots, u_y-1, u_y+1, \ldots, u_m : x = 0 \right\}
\]

\[
\left\{ (Ag_p)^α g^r y, g^y, r A', y, r u_1, \ldots, u_y-1, u_y+1, \ldots, u_m : x \neq 0 \right\}
\]

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 \(δ \in \mathbb{Z}_p\), \((v_{1,1}, v_{1,2}, v_{1,3}), \ldots, (v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n, w_1, \ldots, w_m \in \mathbb{Z}_n, s_{i,1}', \ldots, s_{i,m}', s_{m,i}' \in \mathbb{Z}_n, s_{i+1}', \ldots, s_{m,m}' \in \mathbb{Z}_n, \) and \(κ \in \mathbb{Z}_n\).

It creates the challenge ciphertext as:

- if \(x > i\)
  
  \[
  R_x = k_0^β d'_x, r q, x \\
  T_x = B'^x s_{x,y} \prod_{k \in S_x} μ_k \quad \tilde{R}_x = k_0^β d'_x, r q, x \\
  A_x = B'^x
\]

- if \(x = i\)
  
  \[
  R_x = C'_x, y, g^d p, r q, x \\
  T_x = T \prod_{k \in S_x} \mu_k \quad \tilde{R}_x = C'_x, y, g^d p, r q, x \\
  A_x = T g^δ
\]

- if \(x < i\)
  
  \[
  R_x = g^{v_x, 1} \\
  T_x = (\prod_{k \in S_x} u_k)^v_x \quad \tilde{R}_x = h^{v_x, 1} \\
  A_x = g^{v_x, 2}
\]

\(∀ y \quad C_y = B'^y s_{x,y} h^{v_y} \quad \tilde{C}_y = g^{v_y}\)

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

**Putting it together** By the assumptions above we can now bound the Adversary’s advantage in distinguishing between experiments \(H_2\) and \(H_3\) by \(2ε_{DM3DH} + \epsilon_{BSO}\), 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 $A$ that for some $j$ distinguishes between $H_{3,j}$ and $H_{3,j+1}$ that succeeds with advantage $\epsilon$. 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 $\beta, r_q, r_p, r_q, m, r_q, m, \ldots, r_q, m \in \mathbb{Z}_q$. Additionally, it chooses the exponents $\alpha_1, \ldots, \alpha_m \in \mathbb{Z}_m$. It then sets $h_p = B$ and picks blinding factors $r_p, r_p, m, r_p, m \in \mathbb{Z}_p$. Finally, $u_q, \ldots, u_q, m \in \mathbb{G}_q$ and $u_p, \ldots, u_p, m \in \mathbb{G}_p$ 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 $\gamma \in \mathbb{Z}_n$ are chosen.

The simulator next receives a target set $G$. Everything in the $\mathbb{G}_p$ subgroup for everything in the $\mathbb{G}_q$ subgroup. The simulator chooses random $\gamma \in \mathbb{G}_p$ and picks blinding factors $r_p, r_p, m, r_p, m$. It creates the public keys: $g_p, A = g_p^a, B = g_p^b, C = g_p^c, T$. It creates the public keys:

$$g, h, V = g^\delta g^\gamma, \tilde{V} = h^\delta$$

$$E = g_q^\beta, E_1 = g_q^\beta, r_q, p, 1, \ldots, E_m = g_q^\beta, r_q, m, F_q, 1 = g_q^\beta, r_q, m, F_q, m = g_q^\beta, r_q, m$$

$$F_1 = h_q^\beta, h_q, p, 1, \ldots, F_m = h_q^\beta, h_q, m, F_q, 1 = h_q^\beta, r_q, m, F_q, m = h_q^\beta, r_q, m$$

$$G_1 = e(g, g^\alpha_1, \ldots, G_m = e(g, g^\alpha_m, G_q, 1 = e(g_q, g_q^\alpha_1, \ldots, G_q, m = e(g_q, g_q^\alpha_m$$

$$H_y = g_q^c, g_p^\gamma, g_p^\beta, y \neq j, H_3 = g_q^c, 3 C^\gamma, p, j$$

$$U_1 = u_1, \ldots, U_m = u_m, U_q, 1 = u_q^\beta, \ldots, U_q, m = u_q^\beta, m$$

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

$$K_{x, y} = (d_{x, y}, \alpha_{x, y}, \gamma_{x, y}, d_{x, y}, d_{y, y}+1, \ldots, d_{m})$$

$$= \begin{cases} g^\alpha g_q^{r_q, x, y} u_{x, y}, g_q^{r_q, x, y}, u_{x, y}, u_{x, y}+1, \ldots, u_{x, y+m}, : y \neq j \\ g^\alpha g_q^{r_q, x, y}, C^\gamma, r_q, x, y, u_{x, y}, g^\gamma, g_q^{r_q, x, y}, u_{x, y}, u_{x, y}+1, \ldots, u_{x, y+m}, : y = j \end{cases}$$

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 $(v_1, 1, v_1, 2, v_1, 3), \ldots, (v_i, 1, v_i, 2, v_i, 3) \in \mathbb{Z}_m$, and exponents $s_q, i, 1, \ldots, s_q, i, m \in \mathbb{Z}_q, t_q \in \mathbb{Z}_q$, and $\kappa \in \mathbb{Z}_m$. Next, it chooses random $z_1, \ldots, z_{j-1} \in \mathbb{Z}_p$, $w_1', \ldots, w_m' \in \mathbb{Z}_m$. It then creates the ciphertext as:

$$28$$
if $x \geq i + 1$
\[ R_x = g^{s_q \cdot r_q \cdot x} \]
\[ T_x = \left( \prod_{k \in S_q} \right) s_q \cdot r_q \]
\[ A_x = g^{s_q \cdot r_q} \]

if $x \leq i$
\[ R_x = g^{v_x \cdot 1} \]
\[ T_x = \left( \prod_{k \in S_q} \right) v_x \cdot 2 \]
\[ A_x = g^{v_x \cdot 2} \]

if $y > j$
\[ C_y = g^{\tilde{e}_y \cdot t_q \cdot \kappa \cdot \tilde{w}_y} \]
\[ \tilde{C}_y = A^{-\tilde{e}_y / r_q \cdot \tilde{g}^{w_y}} \]
if $y = j$
\[ C_y = g^{\tilde{e}_y \cdot t_q \cdot T_h \cdot \kappa \cdot \tilde{w}_y} \]
\[ \tilde{C}_y = g^{\tilde{w}_y} \]
if $y < j$
\[ C_y = g^{\tilde{e}_y \cdot t_q \cdot g_p \cdot \kappa \cdot \tilde{w}_y} \]
\[ \tilde{C}_y = g^{\tilde{w}_y} \]

If $T$ forms a 3-party Diffie-Hellman tuple then we simulated $H_{3,j}$, otherwise if $T$ is randomly chosen we simulated $H_{3,j+1}$. The simulator will receive a guess from $A$ 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 $A$’s advantage.

Therefore, we can bound an adversary’s advantage of distinguishing between $H_3$ and $H_4$ as $m \cdot \epsilon_{D_3D_4}$. By our assumption the adversary’s advantage is negligible.

\[ \square \]

C.4 Proof of Claim 5.7

Suppose there exists an adversary $A$ that distinguishes between $H_4$ and $H_5$ with advantage $\epsilon$. 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 $g, h, A = g^a, B = h^b, C = g^b h^c, D = h^b, T$. For the simulation $a = \beta$. The simulator then chooses random exponents $r_1, \ldots, r_m, c_1, \ldots, c_m, \alpha_1, \ldots, \alpha_m, \mu_1, \ldots, \mu_m \in \mathbb{Z}_n$.

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

\[
\begin{bmatrix}
   g, h, V = C, V = D \\
   E_1 = g^{r_1}, \ldots, E_m = g^{r_m}, E_{q,1} = A^{r_1}, \ldots, E_{q,m} = A^{r_m} \\
   F_1 = h^{r_1}, \ldots, F_m = h^{r_m}, F_{q,1} = B^{r_1}, \ldots, F_{q,m} = B^{r_m} \\
   G_1 = e(g, g)^{\alpha_1}, \ldots, G_m = e(g, g)^{\alpha_m}, G_{q,1} = e(A, g)^{\alpha_1}, \ldots, G_{q,m} = e(A, g)^{\alpha_m} \\
   H_1 = g^{\mu_1}, \ldots, H_m = g^{\mu_m} \\
   U_1 = g^{\mu_1}, \ldots, U_m = g^{\mu_m}, U_{q,1} = A^{\mu_1}, \ldots, U_{q,m} = A^{\mu_m}
\end{bmatrix}
\]

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

\[ K_{x,y} = (d'_{x,y}, d''_{x,y}, d_1, \ldots, d_{y-1}, d_y, \ldots, d_m) \]
\[ = g^{s_q x \cdot r_q \cdot y} u_y^{\alpha_{x,y}} g^{s_q x \cdot y} u_1^{\alpha_{x,y}} \ldots u_y^{\alpha_{x,y}} u_{y+1}^{\alpha_{x,y}} \ldots u_m^{\alpha_{x,y}} \]

The simulator gives the adversary all keys except $K_{i,j}$. The adversary gives the query bit $\bar{s}$ and if $\bar{s} = 1$ 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 $\tilde{k}, t \in \mathbb{Z}_n, w_1, \ldots, w_m, s_1, \ldots, s_m \in \mathbb{Z}_n$, and $(v_{i,1}, v_{i,2}, v_{i,3}), \ldots, (v_{i-1,1}, v_{i-1,2}, v_{i-1,3}) \in \mathbb{Z}_n$.
We note that $g^{\tilde{\kappa}} = h^{\kappa'}$ for some unknown $\kappa'$. By simulating the encryption in this manner it is possible to simulate all encryption values, except $C_y$ for $y < j$. However, in this experiment there are no such values since $j = 1$.

For each row $x$ we create five ciphertext components ($R_x, \tilde{R}_x, T_x, A_x, B_x$) for rows other than $i + 1$ as follows:

\[
\begin{align*}
\text{if } x > i + 1: & \quad R_x = E_{q,x}^s, \quad \tilde{R}_x = E_{q,x}^{\tilde{s}}, \quad T_x = (\prod_{k \in S_x} U_{q,k})^{s_t}, \quad B_x = MG_{q,x}^{s_t}, \quad A_x = E_{q,t}^s, \\
\text{if } x < i + 1: & \quad R_x = g^\nu_{x,1}, \quad \tilde{R}_x = g^{\tilde{\nu}_{x,1}}, \quad T_x = (\prod_{k \in S_x} U_{k})^{\nu_{x,2}}, \quad B_x = e(g, g)^{\nu_{x,3}}, \quad A_x = g^{\nu_{x,2}}.
\end{align*}
\]

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

\[\forall y: \quad C_y = H_t^y g^{\tilde{\kappa}w_y}, \quad \tilde{C}_y = g^{w_y}\]

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

\[
\begin{align*}
R_{i+1} &= T_{i+1}^{s_{i+1}r_{i+1}}, \quad \tilde{R}_{i+1} = T_{i+1}^{\tilde{s}_{i+1}r_{i+1}}, \quad A_{i+1} = T_{i+1}^{s_{i+1}t}, \\
T_{i+1} &= T_{i+1}^{s_{i+1}t} \sum_{k \in S_x} \mu_k, \quad B_{i+1} = M e(T, g)^{\alpha_{i+1}s_{i+1}t}.
\end{align*}
\]

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

$\square$