Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption

Amit Sahai*, UCLA Hakan Seyalioglu†, UCLA

Brent Waters‡, University of Texas at Austin

August 1, 2012

Abstract

Motivated by the question of access control in cloud storage, we consider the problem using Attribute-Based Encryption (ABE) in a setting where users' credentials may change and ciphertexts may be stored by a third party. We find that a comprehensive solution to our problem must simultaneously allow for the revocation of ABE private keys as well as allow for the ability to update ciphertexts to reflect the most recent updates. Our main result is obtained by pairing two contributions:

- Revocable Storage. We ask how a third party can process a ciphertext to disqualify revoked users from accessing data that was encrypted in the past, while the user still had access. In applications, such storage may be with an untrusted entity and as such, we require that the ciphertext management operations can be done without access to any sensitive data (which rules out decryption and re-encryption). We define the problem of revocable storage and provide a fully secure construction. Our core tool is a new procedure that we call *ciphertext delegation*. One can apply ciphertext delegation on a ciphertext encrypted under a certain access policy to 're-encrypt' it to a more restrictive policy using only public information. We provide a full analysis of the types of delegation possible in a number of existing ABE schemes.
- Protecting Newly Encrypted Data. We consider the problem of ensuring that newly encrypted data is not decryptable by a user's key if that user's access has been revoked. We give the first method for obtaining this revocation property in a fully secure ABE scheme. We provide a new and simpler approach to this problem that has minimal modifications to standard ABE. We identify and define a simple property called piecewise key generation which gives rise to efficient revocation. We build such solutions for Key-Policy and Ciphertext-Policy Attribute-Based Encryption by modifying an existing ABE scheme due to Lewko et al. [13] to satisfy our piecewise property and prove security in the standard model.

*Research supported in part from a DARPA/ONR PROCEED award, NSF grants 1136174, 1118096, 1065276, 0916574 and 0830803, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, and an Okawa Foundation Research Grant. This material is based upon work supported by the Defense Advanced Research Projects Agency through the U.S. Office of Naval Research under Contract N00014-11-1-0389. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

†Research supported by a NSF Graduate Research Fellowship.

‡Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA through the U.S. Office of Naval Research under Contract N00014-11-1-0382, DARPA N11AP20006, Google Faculty Research award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship, and Packard Foundation Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Department of Defense or the U.S. Government.

It is the combination of our two results that gives an approach for revocation. A storage server can update stored ciphertexts to disqualify revoked users from accessing data that was encrypted before the user's access was revoked. This is the full version of the Crypto 2012 (©IACR) paper.

Keywords. Revocation, attribute-based encryption, delegation

1 Introduction

The need to store information externally has never been higher: With users and organizations expecting to access and modify information across multiple platforms and geographic locations, there are numerous advantages to storing data *in the cloud*. However, there is a natural resistance to the idea of handing over sensitive information to external storage. Since these databases are often filled with valuable data, they are high value targets for attackers and security breaches in such systems are not uncommon, especially by insiders. In addition, organizations with access to extremely sensitive data might not want to give an outside server any access to their information at all. Similar problems can easily arise when dealing with centralized storage within a single organization, where different users in different departments have access to varying levels of sensitive data.

A first step in addressing this problem of trust is to only store information in encrypted form. However, data access is not static – as employees are hired, fired or promoted, it will be necessary to change who can access certain data. A natural solution to this problem is to have users authenticate their credentials before giving them access to data; but such an approach requires a great deal of trust in the server: a malicious party may be able to penetrate the server and bypass authentication by exploiting software vulnerabilities. A solution that avoids this problem is to use cryptographically enforced access control such as attribute-based encryption (ABE) [23]. However, this fails to address the problem that the credentials of a user may change with time. This problem motivated the study of revocation [5] where a periodic (e.g., nightly) key update would only allow non-revoked users to update their keys to decrypt newly encrypted data. Dynamic credentials in the context of stored data, however, present novel challenges that have not been considered in previous studies on revocation. Take the following example.

A Motivating Story. Consider an employee with access to sensitive documents necessary for his work[1]. One day, this employee is terminated and has his access revoked. Now, this employee with insider knowledge of the organization's systems, and who has retained his old key, may attempt to penetrate the database server and decrypt all the files that he once had access to. How can we deal with this type of attack? At first glance, there appears to be an inherent intractability to this problem. Any encrypted file that the user could decrypt with his old key will still be decryptable, after all.

Despite these problems, we believe this situation presents a very real security threat to the organization and is important to address. One method to handle this problem is to decrypt and re-encrypt all stored data every time some employee's credentials are revoked. However, the involvement of secret key information in this process both makes this process cumbersome and opens up the overall system to problematic new vulnerabilities. In general, we want to limit the use of secret key information to *only* key generation and not to database upkeep as the database is handling constant two way communication in our system and is therefore modeled as the most vulnerable party.

[1] While gainfully employed, the worker may have incentives to exercise discretion by only accessing the files necessary for his work and not download all files he has access to. Such discretion may be enforced, for example, through access logs.

We propose a novel method to deal with this problem: We devise a revocable ABE system where the database, using only *publicly available information*, can periodically update the ciphertexts stored on the system, so that as soon as access is revoked for a user U, all stored files (no matter how old) immediately become inaccessible to U after the update process. The database does not even need to know the identities of users whose access was revoked. We emphasize that this is a significant security improvement over decrypting and re-encrypting (which cannot be done with only public information) since in our solution, the database server *never* needs access to any secret keys. Furthermore, secret key holders do not have to interact with the database for the purpose of maintaining access control.

We also note in passing that while re-encrypting each ciphertext after every revocation (in a repeated nesting fashion) can also be applied to solve the problem of access control, this solution is *inefficient* when a ciphertext needs to be updated many times. In such a solution, decryption time will increase linearly and the ciphertext may grow significantly upon each invocation (Even using hybrid encryption, this would add a potentially large ABE header each time).

1.1 Our Results

In this work, we provide the first Revocable ABE scheme that deals with the problem of efficiently revoking stored data. This result is obtained through two main technical contributions:

Revocable Storage Attribute-Based Encryption

We provide ABE encryption schemes with a new property we call *revocable storage*. Revocable storage allows a third party storing ciphertexts to revoke access on previously encrypted data. Additionally, our scheme satisfies strong efficiency guarantees with regard to the lifetime of the database.

We realize revocable storage by introducing a notion of **ciphertext delegation**. In ABE, ciphertext delegation is the ability to restrict a ciphertext with access policy P to a more restrictive policy P' using only *publicly available information*, but without causing the ciphertext size to increase. We initiate the first systematic study of ciphertext delegation for both Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) by analyzing the type of delegation possible in a variety of existing schemes [11, 23, 24, 13].

Protecing Newly Encrypted Data

To utilize revocable storage we need a method for efficiently revoking users credentials such that newly encrypted data will not be decryptable by a user's key if that user's access has been revoked. This topic of revoking credential was considered by Boldyreva et al. [5] in the context of Identity-Based Encryption (and to restricted notions of security for ABE); however was not paired with the revocation of ciphertexts. In addition, ours is the first fully (vs. selectively) secure system[2].

While the Boldyreva et al. system needed to be proven "from scratch", we provide a methodology to obtain a simple construction and proof. We propose a natural modification to standard ABE which we call *Piecewise Key Generation*. Informally (KP-)ABE with piecewise key generation is similar semantically to a standard ABE scheme, except decryption takes as input two keys $K^{(0)}_{P_0,U}$ and $K^{(1)}_{P_1,U}$ from a user U such that decryption only succeeds if both key policies P_0 and P_1 are authorized to decrypt the challenge ciphertext. The scheme is 'piecewise' in that a user may query two different key generation oracles KeyGen_0 or KeyGen_1 in order to get either side of the key adaptively. This allows

[2]Note that a related work [22] proposes a "fully" secure Revocable ABE scheme under a significantly different model, where the revocation list is given as an input to the encryption algorithm. This requirement makes such a scheme unsuitable for the main application of secure storage, since we cannot expect users who create documents to know the credentials of all other users in the system. Indeed, the exact credentials of any user may be a secret meant to be protected.

the user to build his key in a piecewise fashion, after having already seen other pieces of his key. This requirement is close to standard ABE requirements and many existing proof methods extend to prove piecewise security, however, unlike standard ABE, it suffices to imply full revocability in a black-box fashion.

We then show that variants of a method used previously in the context of revocability against stateless receivers in Naor et al. [18] and more closely to our setting in Revocable IBE in Boldyreva et al. [5] succeed in converting *any* ABE scheme with piecewise key generation to a revocable ABE scheme. We give a modification of Lewko et al.'s fully secure ABE scheme [13] that satisfies our requirement and prove its security. Combined with our new techniques for dealing with revocable storage, this yields our Revocable Storage KP-ABE and CP-ABE schemes. Note that we believe our techniques are also compatible with a number of other attribute based encryption schemes [11, 12].

Related Work

Originally proposed by Sahai and Waters [23], *attribute-based encryption* [11, 21, 24, 13, 20] has been an active research area in cryptography in part since it is a primitive with interesting functional applications [10, 5] and can be implemented efficiently [4]. In a key-policy attribute-based encryption (KP-ABE) scheme every secret key is generated with a policy P and ciphertexts are generated with a set of attributes U and decryption is only possible if $P(U) = \mathsf{True}$. The parallel notion where ciphertexts are associated with policies and keys with sets of attributes is called ciphertext-policy attribute-based encryption (CP-ABE). While the problem of delegating a key to a more restrictive key has been considered [11], it is analyzed only in the context of the scheme proposed in the paper. The problem of *revocation* is also a well studied problem, both for general PKI [16, 19, 2, 17, 18, 7, 9], identity based encryption [5, 14] and attribute-based encryption [22]. At a high level, our revocable storage results can be seen as taking methods from forward secure signatures and encryption [6, 3, 1, 15] which were introduced for key management and applying them to ciphertext management by noticing that the key delegation infrastructure can be replicated for the ciphertext through the delegation mechanism we introduce.

2 Preliminaries and Notation

We will assume $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate bilinear pairing whenever it is used. We will use $\vec{v}$ notation for vectors and $[i, j]$ to denote the set of all integers from i to j inclusive or $[i]$ as shorthand for $[1, i]$. We will use $l(\vec{v})$ to denote the dimension of the vector. Throughout this paper, $\log(x)$ will denote the logarithm of x to the base 2. The notation $V(\mathcal{T})$ for a tree $\mathcal{T}$ will denote the set of nodes of this tree. The notation $x \leftarrow X$ for X a randomized algorithm may denote by context either that x is a possible output of that algorithm with positive probability or that x is drawn from the distribution X.

Attribute-Based Encryption

Attribute-based encryption schemes are generally divided into two types depending on if the access policy is embedded in the keys or ciphertexts. We now define KP-ABE, where keys have a access policies incorporated within while ciphertexts are associated with sets of attributes:

Definition 2.1. (Key-Policy ABE) *A KP-ABE scheme with attribute set Ω that supports policies $\mathcal{P}$ with message space $\mathbb{M}$ is defined by the following polynomial time algorithms:*

- **Setup**$(1^\lambda) \to (PK, MSK)$: Setup takes as input the security parameter and outputs the public key and master secret key.

- **KeyGen**$(MSK, P) \to SK_P$: Key generation outputs a secret key with policy $P \in \mathcal{P}$.
- **Encrypt**$(PK, M, S) \to C_S$: Encrypts a message $M \in \mathbb{M}$ under the attribute set $S \subseteq \Omega$.
- **Decrypt**$(SK_P, C_S) \to M$: Decryption successfully recovers the encrypted message if and only if $P(S) = 1$ (in other words, the attribute set satisfies the policy).

The security game requires that an adversary queries keys corresponding to policies, and in the challenge phase requests the encryption of one of two adaptively chosen messages with an attribute set of its choice (generated during the challenge phase). The adversary succeeds if it correctly outputs the encrypted message without ever querying a key for a policy that is satisfied by the attribute set of the challenge ciphertext (we defer a formal security guarantee until we also introduce revocability). A more restrictive notion of *selective security* is also present in the literature in which the adversary must commit to the challenge policy before the security game begins. In a **Ciphertext-Policy ABE** scheme the placement of the policy is reversed, the key generation algorithm takes a set of attributes as input while encryption takes a policy.

Revocable Attribute-Based Encryption

A revocable attribute-based encryption scheme [5] has the added functionality that a user may be placed on a revocation list that will make him unable to decrypt any message encrypted after he was revoked. Such a scheme has a periodic broadcast by a trusted key generation authority that allows the un-revoked users to update their keys and continue decryption. In such a scheme we assume that the total number of users and the number of key updates the scheme can survive are both very large and therefore the scheme's parameters should only depend polylogarithmically on the total number of non-revoked users and time bound.

From this point on we will use the convention that a set of credentials S satisfies a policy P if and only if $P(S) = 1$.

Definition 2.2. (Revocable KP-ABE) *A Revocable KP-ABE scheme with attribute set Ω that supports policies $\mathcal{P}$ with message space $\mathbb{M}$, time bound T and identity length $\mathcal{I}$ consists of the following algorithms:*

- **Setup**$(1^\lambda) \to (PK, MSK)$: Setup takes as input the security parameter and outputs the public key and master secret key.
- **KeyGen**$(MSK, P, ID) \to SK_{P,ID}$: Key generation outputs a secret key with policy $P \in \mathcal{P}$ for the user $ID \in \{0,1\}^{\mathcal{I}}$.
- **Encrypt**$(PK, M, S, t) \to C_{S,t}$: Encrypts $M \in \mathbb{M}$ with attribute set $S \subseteq \Omega$ at time $t \leq T$.
- **KeyUpdate**$(MSK, rl, t) \to K_t$: The update phase takes as input a revocation list rl (a set of strings in $\{0,1\}^{\mathcal{I}}$) and the master secret key and outputs the key update information for time t.
- **Decrypt**$(SK_{P,ID}, K_{t'}, C_{S,t}) \to M$: Decryption successfully recovers the encrypted message if and only if $P(S) = 1$, $t' \geq t$ and ID was not revoked at time t (it was not present on rl when $K_{t'}$ was generated).

Before we define the security game, we first define the oracles that will be used in its definition.

Security Game Oracles. Define the following oracles to use in the security game. These oracles are given access to (PK, MSK) that are generated at the beginning of the security game.

1. The Secret Key Generation oracle $SK(\cdot,\cdot)$ takes as input (P, ID) and returns $SK_{P,ID}$ generated from:
$$SK_{P,ID} \leftarrow \mathsf{KeyGen}(MSK, P, ID).$$
2. The Key Update Generation oracle $K(\cdot,\cdot)$ takes as input t and a revocation list rl and returns K_t generated from:
$$K_t \leftarrow \mathsf{KeyUpdate}(MSK, t, rl).$$

For a p.p.t. adversary A define the following experiment (some additional constraints on the adversary's actions will be enumerated after the experiment's definition). We use the term *challenger* for an internal agent in the security game who participates in the experiment. The challenger's behavior is described in the security game.

RKP-$\textsc{Security}_A(1^\lambda)$:

1. The challenger runs $\mathsf{Setup}(1^\lambda) \rightarrow (PK, MSK)$ and returns PK to A;
2. A is given oracle access to $SK(\cdot,\cdot)$, $K(\cdot,\cdot)$ until it signals the query phase is over;
3. After the query phase, A returns (M_0, M_1, S^*, t^*) to the challenger;
4. The challenger picks b a random bit and returns to A:
$$C_{S^*,t^*} \leftarrow \mathsf{Encrypt}(PK, M_b, S^*, t^*);$$
5. A is once again given oracle access to the two oracles above;
6. A returns a bit b'. The experiment returns 1 if and only if $b' = b$ and the conditions below concerning A's query history are satisfied.

The condition placed on the adversary's queries is as follows: If the adversary makes a query $SK(P, ID)$ that has sufficient credentials to decrypt the challenge ciphertext (in other words, $P(S^*) = 1$), then this ID is revoked for all key generation queries with time at least t^* (in other words, for all queries $K(t, rl)$ with $t \geq t^*$, $ID \in rl$).

Definition 2.3. *A Revocable KP-ABE scheme is secure if for any polynomial time adversary A the advantage of this adversary in the* RKP-Security *game:*

$$2\Pr\left[\text{RKP-}\textsc{Security}_A(1^\lambda) = 1\right] - 1$$

is negligible in λ.

Remark. Note that in previous works on revocation [5] the security definition is presented a little differently. Their definition is much more in line with an actual implementation in which an attacker is not given the authority to query key-updates in a non-sequential order in time (in other words, after seeing a key-update for time t, it can not 'go back in time' and query one for time $t-1$). Their definition corresponds more closely to attacking the scheme in practice, while ours is stronger. We choose our definition both because it is stronger and because it's more straightforward to prove using our techniques. However, for most practical applications (unless there is threat of an attacker being able to generate key updates out of order, or with a time component of his choice), showing the weaker version in [5] would suffice.

3 Revocable Attribute-Based Encryption

The Revocable ABE and IBE constructions given by Boldyreva et al. [5] are built from an underlying ABE scheme through a new transformation they introduce. However, security of the underlying ABE scheme does not imply security of the transformation, and their resulting scheme was proven secure (in the ABE case, in the restricted selective security model) from scratch. In this work we aim to both extend and simplify previous work by investigating the additional properties an ABE scheme needs to satisfy to imply a Revocable ABE scheme following their transformation. Using our result, we modify the fully secure scheme due to Lewko et al. [13] to satisfy our requirement in both the KP-ABE and CP-ABE setting. This yields the first fully secure Revocable KP-ABE and CP-ABE schemes.

The Requirement: Piecewise Key Generation

In our framework the necessary condition an ABE scheme should satisfy in order to imply a revocable ABE scheme is that key generation can be done in a dual componentwise fashion. In the KP-ABE setting keys will have two separate policies P_0 and P_1 such that decryption succeeds for an encryption with attribute set S if and only if $P_0(S) = 1$ and $P_1(S) = 1$. The adversary in the security game is allowed to query these components separately with access to two oracles KeyGen_0 and KeyGen_1 which take as input a policy and an identifier U such that a key $\mathsf{KeyGen}_0(MSK, P_0, U)$ and $\mathsf{KeyGen}_1(MSK, P_1, U')$ can only be combined if $U = U'$ (in our applications this U value will be related to, but will not exactly be a user's identity. For this reason, we switch notation from identities ID to identifiers U at this point).

This security definition is stronger than the standard ABE definition because these components may be queried in an *adaptive* manner, allowing the adversary to build his key *piece by piece*. Note the actual notation we use in our scheme is slightly different than the way it is presented above.

Definition 3.1. (Piecewise Key Generation) *A KP-ABE scheme is said to have* piecewise key generation *with attribute set Ω, supporting policies in $\mathcal{P}$, message space $\mathbb{M}$ and identifier length $\mathcal{I}$ if key generation and encryption are modified to the following syntax:*
KeyGen takes as input the master key MSK, a bit b a policy $P \in \mathcal{P}$ and an identifier $U \in \{0,1\}^{\mathcal{I}}$:

- $\boldsymbol{KeyGen}(MSK, b, P, U) \rightarrow K^{(b)}_{P,U}$.

$\mathsf{Decrypt}$ takes two outputs of KeyGen as the key input instead of one:

- $\boldsymbol{Decrypt}(C_S, K_0, K_1) \rightarrow M$.

We now define correctness of the scheme:

Definition 3.2. (Correctness) *A KP-ABE scheme with* piecewise key generation *is correct if for any $S \subseteq \Omega$ and:*

- $(PK, MSK) \leftarrow \mathsf{Setup}(1^\lambda)$
- $C_S \leftarrow \mathsf{Encrypt}(PK, M, S)$
- $P_0, P_1 \in \mathcal{P}$ *such that* $P_0(S) = P_1(S) = 1$*:*

If $\mathsf{KeyGen}(MSK, 0, P_0, U) \rightarrow K^{(0)}_{P_0,U}$ and $\mathsf{KeyGen}(MSK, 1, P_1, U) \rightarrow K^{(1)}_{P_1,U}$, then,

$$\boldsymbol{Decrypt}(C_S, K^{(0)}_{P_0,U}, K^{(1)}_{P_1,U}) = M.$$

The definition for security for a scheme with *piecewise key generation* is now defined as one would expect: Unless the adversary has queried a single identifier U to

$$\mathsf{KeyGen}(MSK, 0, P_0, U) \text{ and } \mathsf{KeyGen}(MSK, 1, P_1, U)$$

such that $P_0(S) = P_1(S) = 1$, he should not be able to distinguish which message has been encrypted. We formalize this through the following game:

PIECEWISE KPABE SECURITY$_A(1^\lambda)$:

1. The challenger runs $\mathsf{Setup}(1^\lambda) \to (PK, MSK)$ and sends PK to A;
2. A makes queries of the type (b, P, U) for $b \in \{0,1\}, P \in \mathcal{P}$ and $U \in \{0,1\}^{\mathcal{I}}$. The challenger runs $\mathsf{KeyGen}(MSK, b, P, U)$ and returns the key to A;
3. A signals the query phase is over and returns (M_0, M_1, S);
4. The challenger picks $b \xleftarrow{\$} \{0,1\}$ and returns $\mathsf{Encrypt}(PK, M_b, S)$ to A;
5. A has another query phase as previously;
6. A sends a bit b' to the challenger;
7. If for any U, A has queried $\mathsf{KeyGen}(MSK, 0, P_0, U)$ and $\mathsf{KeyGen}(MSK, 1, P_1, U)$ such that $P_0(S) = P_1(S) = 1$ return 0;
8. If $b' = b$ return 1, otherwise return 0.

Definition 3.3. *A KP-ABE scheme with piecewise key generation is secure if for any polynomial time adversary A the advantage of this adversary in the* PIECEWISE KPABE SECURITY *game:*

$$2\Pr\big[\text{PIECEWISE KPABE SECURITY}_A(1^\lambda) = 1\big] - 1$$

is negligible in λ.

3.1 Revocability from Piecewise KP-ABE

Our final construction in Theorem 7.1 will incorporate not only revocability but also the ability to efficiently delegate stored ciphertexts. In this section we will state a restricted corollary of this more robust result that only addresses revocability. We state this result separately to draw attention to the fact that the piecewise key generation is the only necessary condition to achieve revocability. We defer the proof to the more general result. Further intuition as to how piecewise key generation is used will be provided later in the paper.

For our result we will need to make a requirement on exactly what types of policies we are assuming our piecewise KP-ABE scheme can support. Since our ultimate goal is to apply our result to the construction of [13], we will state our result if the KP-ABE scheme supports access policy formatted as LSSS matrices (as in [13]).

Linear Secret Sharing Schemes

We begin with a brief overview of LSSS matrices. A LSSS (linear secret sharing scheme) policy is of the type (A, ρ) where A is an $n \times l$ matrix over the base field $\mathbb{F}$ (whose dimensions may be chosen at the time of encryption) and ρ is a map from $[n]$ to Ω, the universe of attributes in the scheme. A policy (A, ρ) satisfies an attribute set $S \subseteq \Omega$ if and only if $1 = (1, 0, 0, \ldots, 0) \in \mathbb{F}^l$ is contained in

$\mathsf{Span}_{\mathbb{F}}(A_i : \rho(i) \in S)$ where A_i is the i^{th} row of A. An LSSS policy (A, ρ) is called *injective* if ρ is injective.

We now state our result on a black-box reduction between KP-ABE schemes with piecewise key generation supporting LSSS policies to Revocable KP-ABE supporting LSSS policies.

Theorem 3.4. *Let $\mathcal{E}$ be a KP-ABE scheme with piecewise key generation supporting injective LSSS matrices with attribute set of size ω. Then there exists a Revocable KP-ABE scheme $\mathcal{F}$ supporting injective LSSS matrices with time bound T under the same complexity assumptions as $\mathcal{E}$ with an attribute set of size $\omega - 2\log(T)$.*

4 Revocable-Storage Attribute Based Encryption

Motivated by settings in which a third party is managing access on encrypted data, we now present a new direction for revocability. We call the property we achieve *revocable storage* - the ability for a ciphertext to be updated using only the public key so that revoked users can no longer decrypt the refreshed ciphertexts. This is an additional functionality that is added onto standard revocability, allowing an untrusted third party storing ciphertexts to update them in a way to make revoked users unable to decrypt messages they were once authorized to access. This can be thought of as an adaptation of forward security [6] which is used to manage keys, to managing ciphertexts.

Definition 4.1. (Revocable Storage) *A Revocable KP-ABE scheme has Revocable Storage if it has the following additional algorithm:*

- **CTUpdate**$(C_{S,t}, PK) \rightarrow C_{S,t+1}$: The update procedure transforms a ciphertext encrypted at time t to an independent encryption of the same message under the same set S at time $t+1$.

Formally, the requirement is:

For any attribute set $S \subset \Omega$, time $t \in [T-1]$ and message $M \in \mathbb{M}$, if PK and C are such that $\mathsf{KeyGen}(1^\lambda) \rightarrow (MSK, PK)$ and $\mathsf{Encrypt}(PK, M, S, t) \rightarrow C$ then,

$$\mathsf{Encrypt}(PK, M, S, t+1) \equiv \mathsf{CTUpdate}(C, PK).$$

Where $\equiv$ denotes equality in distribution. Note that this is a very strong distributional requirement, as we are insisting that starting from *any* ciphertext allows complete resampling from the output of the encryption algorithm. This is in contrast to merely requiring that the updated ciphertext be a valid ciphertext under the new time component. Having this strong distributional guarantee will be vital in applications where we may be delegating multiple times from the same base ciphertext.

We will call a Revocable KP-ABE scheme with Revocable Storage, a Revocable Storage KP-ABE scheme for brevity. Notice that the above procedure allows us to accomplish our motivating applications; it allows a third party storing ciphertexts to update the ciphertexts after revocation has been done at time t so that only the non-revoked users can continue decrypting. We will impose the restriction that all parameters should only depend polylogarithmically on T, the upper bound for the time component and n, the total number of users in the scheme. It is worth observing that there are simple inefficient ways (without the independence guarantee) to satisfy this requirement assuming a vanilla KP-ABE scheme (i.e. by having $C_{S,t} = \{\mathsf{Encrypt}(PK, M, S, t') : t' \geq t\}$ and having the update procedure simply delete the lowest indexed ciphertext) that depend polynomially on T.

5 Ciphertext Delegation

Revocable storage centers around allowing an untrusted third party to manage access on ciphertexts by incrementing the time component. To accomplish this, we need a process by which a ciphertext can be made harder to decrypt using only public operations in a more efficient way than simply re-encrypting under the more restrictive policy.

We call this problem *ciphertext delegation* - where a user who has access to only the ciphertext and public key may process this information into a completely new encryption under a more restrictive access policy. We consider this problem for attribute based encryption and show a simple method to classify delegation made possible in existing ABE schemes. We say that a ciphertext with a given access policy can be *delegated* to a more restrictive policy if there is a procedure that given any valid encryption of a message under the first policy produces a *independent and uniformly chosen* encryption of the same message under the new access policy. We again stress that delegation is required to produce a new encryption of the same message that is independent of the randomness and access policy of the the original ciphertext being delegated from. This requirement is crucial if multiple delegations from the same base ciphertext are ever used in a scheme. Without this guarantee, multiple delegations may have correlated randomness and the security of the underlying scheme would not imply any security in these applications.

Remark. Note that we only consider CPA security in this paper as the re-randomizability guarantee makes CCA2 security unattainable (as the challenge ciphertext can simply be re-randomized and sent to the decryption oracle). An analysis of possible relaxations of CCA2 security that are possible in our model is an interesting question for future work.

KP-ABE Ciphertext Delegation

For monotone access policies (as are generally considered in the literature [8, 13, 23]), the natural way to restrict access is by removing attributes from the ciphertext's attribute set. Note that for non-monotone access policies, delegation is not achievable without severely limiting the key policies permitted as any change in attribute set will make the ciphertext decryptable to certain simple policies not previously authorized, implying that delegation would violate security if these policies are supported.

Definition 5.1. (KP Delegation) *A KP-ABE scheme $\mathcal{E}$ with message space $\mathbb{M}$ and attribute space Ω is said to have ciphertext delegation if there is an algorithm* Delegate *with the following guarantee: For any $S' \subseteq S \subseteq \Omega$ and any $(PK, MSK) \leftarrow \mathcal{E}.Setup(1^\lambda)$, $M \in \mathbb{M}$ and $C_S \leftarrow \mathcal{E}.Encrypt(PK, M, S)$:*

$$\mathcal{E}.\mathsf{Delegate}(PK, C_S, S') \equiv \mathcal{E}.\mathsf{Encrypt}(PK, M, S').$$

We show briefly how ciphertext delegation is possible in the KP-ABE scheme due to Goyal et al. [8] as the delegation procedures in the other listed schemes follow similarly. In this scheme, a ciphertext C_S with attribute set S is encrypted as:

$$C_S = (S, MY^s, \{T_i^s : i \in S\})$$

where s is chosen uniformly in $\mathbb{Z}_p$ and $\{T_i : i \in S\}, Y$ are part of the public key. To delegate this to an encryption under attribute set $S' \subseteq S$, we first modify the ciphertext to be $(S', MY^s, \{T_i^s : i \in S'\})$ by replacing S with S' and deleting elements in the third component. While this is a valid ciphertext under attribute set S' notice that it is not a valid delegation since we require the delegated ciphertext to be a completely independent encryption. By generating $Y^{s'}$, $\{T_i^{s'} : i \in S'\}$ with s' uniformly chosen from $\mathbb{Z}_p$, the ciphertext can be modified to $(S', MY^{s+s'}, \{T_i^{s+s'} : i \in S'\})$ which is a fresh uniform encryption of M with attribute set S'. A similar analysis also holds for [13, 23] which allows us to conclude:

Theorem 5.2. *The KP-ABE schemes defined in [8, 13, 23] have ciphertext delegation.*

5.1 Ciphertext Policy Delegation

The more involved analysis for delegation comes when considering CP-ABE schemes as the ciphertexts may be associated with complex access policies. The most prominent CP-ABE schemes in the literature are built upon an underlying secret sharing scheme corresponding to their access policy: [23, 4] take their policies input as threshold trees while [24, 13] take their policies as LSSS matrices. The first step in the encryption procedure in these schemes is to share a uniformly chosen secret according to the implied secret sharing scheme (as either Shamir secret sharing applied to threshold trees, defined shortly, or according to a LSSS matrix) with the shares of the secret embedded into certain components of the ciphertext. The encryption scheme is said to be *based on* a given secret sharing scheme if it falls into the above paradigm for this secret sharing scheme. Before we continue, we remind the reader of the syntax for LSSS matrices and threshold trees. Note that LSSS based access structures contain all threshold tree based access structures.

Linear Secret Sharing Notation

We recall some of the basic notation defined previously. A linear secret sharing scheme over $\mathbb{F}$ for a set of players $\mathbb{P}$ is defined through a pair (A, ρ) with A the share generating matrix of dimension $n \times l$ and ρ the assignment function from $[n] \to \mathbb{P}$. To evaluate the shares of a secret, the vector $w = (s, r_1, \ldots, r_{l-1})$ is constructed where s is the secret to be shared and $r_1, \ldots, r_{l-1} \in \mathbb{F}$ are uniformly chosen at random. The share vector is then defined as $\vec{v} = Aw$ with party i receiving all $\vec{v}[j]$ such that $\rho(j) = i$.

Threshold Trees

An access tree is defined as a tree $\mathcal{T}$ where every non-leaf node x has assigned a number n_x (corresponding to the threshold level of that node) and every leaf x is assigned a (not necessarily unique) player $i \in [n]$ (call this value $\beta(x)$). Whether or not an access tree is satisfied by a set of players $A \subseteq [n]$ is determined by first assigning each leaf x, the boolean TRUE if $\beta(x) \in A$ and setting all other leaves assigned to players not in A to FALSE. A non-leaf node x on the second level of the tree is now assigned the value TRUE if at least n_x of its children are TRUE and set to FALSE otherwise. This process is recursed through all levels of the tree and A is said to satisfy $\mathcal{T}$ if the root of $\mathcal{T}$ is finally assigned TRUE. Implementing a secret sharing scheme with the access structure defined by a threshold tree is straightforward using a simple generalization of Shamir secret sharing. The secret to be shared is first assigned to the root node of the tree - every descendent of a fixed node x is then recursively assigned the evaluation of a random polynomial p of degree $n_x - 1$ at the node's index - a non-zero number assigned to the node so that no two children of the same node share the same index and is part of the description of the tree such that the polynomial such that $p(0)$ is the value assigned to x. After this process has recursed through the tree, the player i is returned all leaf values labeled i (note that the labeling and indexing are two different operations). Call this secret sharing scheme, *Shamir secret sharing applied to threshold trees.*

For our applications we will use slightly non-standard notation for secret sharing schemes for notational efficiency. We say that a secret sharing scheme $\mathcal{S}$ is made of two probabilistic polynomial time algorithms Share and Rec. The reconstruction algorithm Rec behaves as expected by reconstructing the shares returned by the parties. The sharing function Share takes as input a secret in $\mathbb{S}$ as well as a policy P such that a group of users $\mathcal{P}$ are authorized to reconstruct the secret if and only if $P(\mathcal{P}) = 1$. Instead of the Share functionality returning one share to each party, it will output $(\vec{v}, \alpha)$ where the components of $\vec{v}$ are elements in the share space and the second output α is an assignment of indices of $\vec{v}$ to $[n]$ (where n is the number of players) that determines which

components of $\vec{v}$ should be sent to which player (if $\alpha(i) = j$, then $\vec{v}[i]$ is sent to player P_j; note there may be multiple elements sent to some or all players).

Delegation Procedures

In this section we analyze policy delegation in CP-ABE, the process by which a ciphertext C_P (encrypted with policy P) can be delegated into a new $C_{P'}$. To this end, we observe that most known CP-ABE schemes actually have the access structure 'built in' to the ciphertext in the form of a secret sharing scheme. Informally, these schemes work as follows: The encryption algorithm will first generate some secret $s \in \mathbb{S}$ at random and internally share it according to the policy it is being encrypted under. In the decryption phase, the decryptor is then able to 'use' all the shares that correspond to attributes he possesses to reconstruct the underlying secret, succeeding in recovering the message if he does. Unfortunately formally defining this intuition is somewhat difficult. It is with this motivation that we define F, the share-extractor that recovers the shares embedded in the ciphertext. We will call a ciphertext valid if it is a possible encryption of some message in the message space $\mathbb{M}$ under some supported policy.

We say a CP-ABE scheme is based on a specific secret sharing scheme $\mathcal{S}$ with share space $\mathbb{S}$ if there is a (possibly inefficient) share-extractor F that on input any valid ciphertext C_P outputs $(s, (\vec{v}, \alpha))$ where $s \in \mathbb{S}$, $(\vec{v}, \alpha) \leftarrow \mathcal{S}.\,\mathsf{Share}(s, P)$. The guarantee is that the shares the extractor recovers correspond correctly to a secret shared under the ciphertext access structure: If $F(C) = (s, (\vec{v}, \alpha))$ with $(\vec{v}, \alpha) \leftarrow \mathcal{S}.\,\mathsf{Share}(s, P)$ then C is a valid ciphertext with policy P. We assume F outputs $\perp$ on any invalid ciphertext input.

Elementary Ciphertext Manipulations

We will define a CP-ABE scheme as having ciphertext delegation if it allows for certain operations to be performed on its ciphertext that manipulate the shares of the underlying secret sharing scheme in a fixed way. A CP-ABE scheme allows *elementary ciphertext manipulations* if there exist the following efficient public operations on valid ciphertexts. As the first requirement does not change the underlying structure of the ciphertext but refreshes the encryption, we call this Property 0.

Property 0. Any well-formed ciphertext under a given access policy can be re-randomized to an independent encryption of the same message under the same policy.

In addition, there are the following four efficient operations on valid ciphertexts (for all examples below let C be a valid ciphertext with $F(C) = (s, (\vec{v}, \alpha))$ and $[n]$ the set of attributes):

Property 1. *Linearly combining shares with the same label:* There is a p.p.t. algorithm Combine such that for any $i, j \leq l(\vec{v})$ with $\alpha(i) = \alpha(j)$ we have $\mathsf{Combine}(C, i, j, a_i, b_j, d) = C'$ with $F(C') = (s, (\vec{v}', \alpha))$ where:

$$\vec{v'}[k] = \begin{cases} \vec{v}[k], & \text{for all } k \neq i; \\ a_i\vec{v}[i] + b_j\vec{v}[j] + d, & \text{if } k = i. \end{cases}$$

Property 2. *Deleting components of $\vec{v}$:* There is a p.p.t. algorithm Delete such that for any $i \leq l(\vec{v})$ we have $\mathsf{Delete}(C, i) = C'$ where $F(C') = (s, (\vec{v}', \alpha'))$, $l(\vec{v'}) = l(\vec{v}) - 1$ and:

$$(\vec{v'}[k], \alpha'(k)) = \begin{cases} (\vec{v}[k], \alpha(k)) & \text{for } k < i; \\ (\vec{v}[k+1], \alpha(k+1)) & \text{if } k \geq i. \end{cases}$$

Property 3. *Adding new entries to $\vec{v}$:* There is a p.p.t. algorithm Add such that for any $i \in [n]$ we have $\mathsf{Add}(C, i) = C'$ where $F(C') = (s, (\vec{v'}, \alpha'))$, $l(\vec{v'}) = l(\vec{v}) + 1$ and:

$$(\vec{v'}[k], \alpha'(k)) = \begin{cases} (\vec{v}[k], \alpha(k)) & \text{for } i \leq l(\vec{v}); \\ (0, i) & \text{if } k = l(\vec{v}) + 1. \end{cases}$$

Property 4. *Swapping entries in $\vec{v}$*: There is a p.p.t. algorithm Swap such that for any $i, j \leq l(\vec{v})$ we have $\mathsf{Swap}(C, i, j) = C'$ where $F(C') = (s, (\vec{v'}, \alpha'))$ and:

$$(\vec{v'}[k], \alpha'(k)) = \begin{cases} (\vec{v}[k], \alpha(k)) & \text{for } k \notin \{i, j\}; \\ (\vec{v}[i], \alpha(i)) & \text{if } k = j; \\ (\vec{v}[j], \alpha(j)) & \text{if } k = i. \end{cases}$$

We now state our observation on known CP-ABE schemes that fall under the framework we have just described. Consulting the constructions for each scheme, it is immediate to observe where the underlying secret sharing scheme is embedded during encryption and how the operations above can be performed.

Theorem 5.3. *The CP-ABE schemes given in SW05 [23], BSW07 [4], Waters11 [24] and LOSTW10 [13] allow for elementary ciphertext manipulations. These schemes are based on Shamir secret sharing, Shamir secret sharing generalized to threshold trees and linear secret sharing schemes respectively.*

The method of allowing ciphertext manipulation in each case is straightforward, and consists of deleting, adding or linearly combining certain elements of the ciphertext (similarly to the method given before Theorem 5.2). We omit the details here as the statement is straightforward to check in each instance.

These elementary ciphertext manipulations are our basic tools for ciphertext delegation - Notice that if a CP-ABE scheme allows delegation of any well-formed ciphertext under policy P to be processed into a well-formed ciphertext under policy P' using the elementary delegation operations, this implies that any ciphertext encrypted with policy P may be delegated to a uniformly random encryption of the same message under policy P' by the re-randomization guarantee. Depending on what secret sharing scheme the CP-ABE scheme is based on, the elementary delegation operations may have significantly different capabilities which we outline below.

Theorem 5.4. *Let $\mathcal{E}$ be a CP-ABE scheme based on Shamir secret sharing generalized to threshold trees that allows for elementary ciphertext manipulations. A ciphertext $C_{\mathcal{T}}$ encrypted under threshold tree $\mathcal{T}$ can be delegated to a ciphertext $C_{\mathcal{T}'}$ if $\mathcal{T}'$ can be derived from $\mathcal{T}$ by any sequence of the following operations:*

1. *Inserting a node x along an edge with $n_x = 1$ (In other words, splitting an edge into two, connected by a node x). Note that this does not affect the access structure, but does change the structure of the tree.*
2. *For any node x, increasing its threshold, $n_x \to n_x + 1$ while optionally adding another leaf y labeled arbitrarily with index not equal to that of any sibling.*
3. *Deleting a subtree.*

Proof. Properties *1.* and *3.* are trivial as a valid secret sharing under one tree will also be valid under the second after *1.* is performed and *3.* only requires repeated invocations of Delete. We now give the full proof of *2.* A similar method was used for key delegation in [11].

Proof of *2.* Take C with $F(C) = (s, (\vec{v}, \alpha))$ where $(\vec{v}, \alpha) \leftarrow \mathsf{Share}(s, \mathcal{T})$ for the threshold tree $\mathcal{T}$. We will show how to increase the threshold of a node x while inserting the leaf node labeled $i \in [n]$

with index μ, if this leaf node is not desired it can simply be deleted by the Delete functionality after the delegation. We will insert this new share to be the last entry of the share vector, which can be changed with the Swap functionality if desired.

Let $q(X)$ be the $n_x - 1$ degree polynomial associated with the node x (recall that in threshold trees, each node is associated with a polynomial). We will show how to change the implied polynomials in the tree so that the polynomial at this node is actually changed to $q'(X) = (1 - X/\mu)q(X)$. This will add 1 to the degree of the polynomial at x, corresponding to incrementing n_x while still interpolating to the same point as q at 0. To change the polynomial to the above value with an additional leaf y assigned to i added as a child to x:

1. Use $\mathsf{Add}(C, i)$ to set the share at this new vector component to 0.
2. The remaining net effect of changing the polynomial as above will be to multiply the shares of leaves that are descendants of a child z of x by $(1 - \mathsf{index}(z)/\mu)$ where the multiplication is done using Combine to scale entries of $\vec{v}$. This is implicitly changing the value assigned to the node z and all descendants of z by multiplying the previous value with $(1 - \mathsf{index}(z)/\mu)$.

□

Note that the delegation allowed through the elementary ciphertext manipulations on Shamir threshold tree based schemes can be elegantly quantified, the effect on the access structure of each of the operations we discuss above can be immediately seen. Similar delegation properties are possible for general LSSS schemes in a straightforward application of the manipulations.

Theorem 5.5. *Let $\mathcal{E}$ be a CP-ABE scheme based on linear secret sharing matrices that allows for elementary ciphertext manipulations. A ciphertext C_A encrypted under policy (A, ρ) can be delegated to a ciphertext $C_{A'}$ with policy (A', ρ') if (A', ρ') can be derived from (A, ρ) through any sequence of the following operations:*

1. *Swapping rows: The output of the function ρ remains unchanged except for at these rows, where it is also swapped. Note that this does not affect the access structure.*
2. *Deleting rows: The output of ρ' on a row of A' corresponds to the output of ρ on the corresponding row of A (before it was shifted by deletion).*
3. *Adding a new row : Let A be $m \times n$, then A' is $m + 1 \times n + 1$ with A the upper left $m \times n$ submatrix of A' equal to A and $m + 1 \times n + 1$ entry equal to 1. The remaining entries in the $n + 1^{st}$ column are set 0, but $\rho(m + 1)$ and the remaining entries in the $m + 1^{st}$ row may be assigned arbitrarily.*
4. *Linearly combining rows: If $\rho(i) = \rho(j)$, A' may be achieved from A by multiplying row i by a constant and adding it to row j to get a new value for row j.*

Proof. Deleting, swapping and combining rows can be easily achieved by using the Delete, Swap and Combine operations respectively as doing these operations on the rows of A and then sharing the secret is equivalent to first sharing the secret and then performing these operations on the resulting shares.

To add a new row as described in 3 it is sufficient to use $\mathsf{Add}(C, j) = C'$ on the ciphertext C where j is the new labeling of $m + 1^{st}$ row. To see that this achieves the valid delegation let $F(C) = (s, (\vec{v}, \alpha))$. Then there is some implied vector $w = (s, r_1, r_2, \ldots, r_{n-1})$ such that $\vec{v} = Aw$. Then, $(s, (\vec{v} \circ 0, \alpha'))$ where α' agrees with α on the first m rows, and has $\alpha'(m + 1) = j$ satisfies the property that $\vec{v} \circ 0 = A'w'$ where $w' = (s, r_1, r_2, \ldots, r_n, -(a_1 s + a_2 r_1 + \ldots, a_n r_{n-1})$ and $(a_1, a_2, \ldots, a_n)$ are the new values for the first n entries of the newly added $m+1^{st}$ row (recall, these can be assigned arbitrarily). This implies that the ciphertext returned by the Add procedure is valid under the desired access policy. □

KEY DELEGATION. Note that the methods we propose for ciphertext delegation can also be applied to delegation of key policies in some existing KP-ABE schemes [8, 23]. Some key delegation techniques possible for a KP-ABE scheme based on threshold trees are given in [8].

6 Managing the Time Structure Efficiently

To solve the problem of managing time efficiently we use a binary tree $\mathcal{Q}$ of depth $\log(T) = r$ (from now on we will assume T is a perfect power of 2 for notational convenience, if not then the r will just be taken to $\log(T)$ rounded up to the next integer), in a method used in previous forward secure literature [6, 3, 1, 15]. Nodes of this tree will correspond to different attribute sets, while a single encryption of the delegatable scheme, interestingly, will be comprised of multiple encryptions from the underlying KP-ABE scheme, each one corresponding to an attribute set from a different node of the tree. While only one of these ciphertext components may be necessary for a secret key holder to decrypt, the ciphertexts include multiple components for delegation purposes. We stress again that the construction we present, as well as the properties we use are in present in the cited works, we present a full proof here mainly for self-containment and to present the result in a form more compatible with our application.

Labeling Nodes of the Tree

Associate with each leaf of $\mathcal{Q}$ a string corresponding to its path from the root with 0's denoting that the path traverses the left child of the previous node and 1's indicating traversing through the right child. As an example, the string 0^r corresponds to the leftmost leaf of the tree while $0^{r-1} \circ 1$ corresponds to its right sibling. Associate non-leaf nodes to strings by the path from the root by using $*$ values to pad the string to r bits. For example, the root node will be referred to by the string $*^r$ while $0 \circ *^{r-1}$ refers to its left child. The string associated with a node v will be labeled $b(v)$. We refer to the node associated with a string x as v_x; notice this gives a bijection between the time components $t \in \{0,1\}^r$, and the leaves of the tree $\mathcal{Q}$.

Managing the access structure will require associating each time $t \in [T]$ with a *set* of nodes in the tree through a process we describe below. The following theorem will be the main method through which we will handle the time component of our final revocable storage construction. For the theorem statement we will replace the time bound T with q to avoid confusing it with the trees, that will be called $\mathcal{T}$. The value r is now $\log(q)$. Below the term 'efficiently computable' means in time linear in r. We will also use the convention that a node v is considered to be both an ancestor and descendant of itself.

Theorem 6.1. *Let $\mathcal{Q}$ be a tree of depth r for $r \in \mathbb{N}$ and let $q = 2^r$. There are efficiently computable subsets of $V(\mathcal{Q})$ (the node set of $\mathcal{Q}$), $\{\mathcal{T}_i : i \in [q]\}$ such that for all $t \in \{0,1\}^r$:*

- *Property 1. $\mathcal{T}_t$ contains an ancestor of $v_{t'}$ if and only if $t \leq t'$;*
- *Property 2. If $u \in \mathcal{T}_{t+1}$ then there is an ancestor of u in $\mathcal{T}_t$;*
- *Property 3. $|\mathcal{T}_t| \leq r$.*

We first give some intuition of how this sequence of trees will be used in the scheme. A secret key for time t will be associated with the leaf v_t of $\mathcal{Q}$ while a ciphertext at time t' will be associated with the set of nodes $\mathcal{T}_{t'}$. A key for time t will succeed in decryption (by using the underlying KP-ABE scheme) if and only if v_t is a descendant of a node corresponding to the ciphertext. Property 1. above will then imply that a key at time t will only succeed in decrypting ciphertexts from earlier times.

Additionally, in our implementation delegation will be possible by traversing down the tree - a ciphertext associated with a set of nodes will be delegatable to a ciphertext associated with another set if and only if for every node in the target set (for the ciphertext being delegated to), one of its ancestors is in the first set (the set associated with the ciphertext being delegated from). Property 2. allows us to conduct linear delegation. Property 3. guarantees that this process can be done efficiently.

Proof. We now describe the process to construct $\mathcal{T}_t$. Let $\mathcal{T}_0$ consist only of the root of $\mathcal{Q}$. For all $t \in [1, q]$, let v_t be as before and Path_t be the set of nodes from v_t to the root of $\mathcal{Q}$ (including v_t and the root). Then, $\mathcal{T}_t$ will be the set of right children of Path_{t-1} that are not included in Path_{t-1}. Notice that since the number of elements in $\mathcal{T}_t$ is at most the number of elements in Path_{t-1} (which is of size r), we have Property 3. immediately.

Proof of Property 1. We begin by showing that if $t \leq t'$ then $\mathcal{T}_t$ contains an ancestor of $v_{t'}$. Notice the statement is trivial if $t = 0$ since all nodes are descendants of $\mathcal{T}_0$. We can therefore consider only $t > 0$ to complete the analysis.

Let w be the lowest common ancestor of v_{t-1} and $v_{t'}$. Notice the node $v_{t'}$ is to the right of v_{t-1} (by the naming convention the nodes on the bottom layer are arranged in ascending order from left to right). Therefore $v_{t'}$ is a descendant of the right child of w while v_{t-1} is a descendant of its left child. Since w is in Path_{t-1} and its right child isn't (since v_{t-1} is a descendant of the left child of w), the right child of w is in $\mathcal{T}_t$. Since we already established that $v_{t'}$ is a descendant of the right child of w, this implies $v_{t'}$ is a descendant of a node in $\mathcal{T}_t$.

We now show the other direction of Property 1. Let $t > t'$ and we will show $\mathcal{T}_t$ does not contain an ancestor of $v_{t'}$. First notice that if $t = t' + 1$ the statement is trivial since $\mathcal{T}_t$ does not contain any elements in $\mathsf{Path}_{t-1} = \mathsf{Path}_{t'}$ by construction.

So we can assume $t > t' + 1$. Similarly to the other direction, if $\mathsf{Path}_{t'}$ and Path_{t-1} diverge after some node w, Path_{t-1} follows the right child, while $\mathsf{Path}_{t'}$ follows the left. This implies $\mathsf{Path}_{t'}$ never overlaps with the right child of a member of Path_{t-1} unless this child is also in Path_{t-1}. This implies that $\mathsf{Path}_{t'} \cap \mathcal{T}_t = \emptyset$ as desired.

Proof of Property 2. Take any $u \in \mathcal{T}_{t+1}$ and we will show that u has an ancestor in $\mathcal{T}_t$. If $t = 0$ the statement is trivial since $\mathcal{T}_0$ is the root node. We therefore can assume that $t > 0$ to conclude the proof.

By construction u is the right child of its parent $p(u) \in \mathsf{Path}_t$ and $u \notin \mathsf{Path}_t$. Note that u is never the root node by construction. Let w be the lowest common ancestor of v_{t-1} and v_t. Then, similarly to the proof of the previous property, v_t is a descendant of the right child of w. There are then two cases to consider, depending whether or not $p(u) \in \mathsf{Path}_{t-1}$:

1. We first address the case when $p(u) \in \mathsf{Path}_{t-1}$. If $u \notin \mathsf{Path}_{t-1}$ then this implies $u \in \mathcal{T}_t$ as it is the right child of a node in Path_{t-1} that is not also in Path_{t-1}. This implies u has an ancestor in $\mathcal{T}_t$ as desired.

   On the other hand, consider the case where $u \in \mathsf{Path}_{t-1}$. This implies v_{t-1} is a descendent of u. However, since $u \in \mathcal{T}_{t+1}$ we know that $u \notin \mathsf{Path}_t$ by the construction of $\mathcal{T}_{t+1}$. Therefore, $p(u) = w$ as we have assumed it is in Path_t and Path_{t-1} and only Path_{t-1} continues through u. However, similarly to the proof of the previous property, Path_{t-1} continues down the left child of w. But by assumption u is the right child of $p(u) = w$, a contradiction.

2. We next address the case when $p(u) \notin \mathsf{Path}_{t-1}$. Since $p(u) \in \mathsf{Path}_t$, $p(u)$ is a descendant of w since $p(u)$ must be a descendant of the node where the paths join if it is not on both paths. Notice $p(u) \neq w$ since $p(u) \notin \mathsf{Path}_{t-1}$. Since Path_t continues down the right child of w and $p(u)$ is on a lower level than w, this implies $p(u)$ is a descendant of the right child of w. The right child of w is a member of $\mathcal{T}_t$ since it is a right child of a node in Path_{t-1} and is not a member of Path_{t-1} (since Path_{t-1} continues down the left child of w). This implies $p(u)$ and therefore u is a descendant of a member of $\mathcal{T}_t$.

□

7 Revocable Storage KP-ABE

By combining the method above for achieving linear delegation with our fully secure KP-ABE scheme with piecewise key generation and ciphertext delegation (given in Section 8), we will now show the following theorem. We defer the construction of our KP-ABE scheme with the required guarantees until after this section as the specific construction and security proof is involved and unnecessary for understanding the connection between piecewise key generation, delegation and revocable storage.

Theorem 7.1. *Let $\mathcal{E}$ be KP-ABE scheme with ciphertext delegation and piecewise key generation that supports injective LSSS matrices with attribute set size ω. Then there exists a Revocable Storage KP-ABE scheme $\mathcal{F}$ that supports injective LSSS matrices with time bound T with attribute set size $\omega - 2\log(T)$.*

To prove the above theorem, we will use a second tree $\mathcal{U}$ for revocation management (as in [18, 5]) with the identities $\{0,1\}^{\mathcal{I}}$ labeling the leaves. For a set of leaves V, the function $\mathcal{U}(V)$ returns [3] a (deterministically chosen) collection of nodes of $\mathcal{U}$ such that some ancestor of a leaf v is included in $\mathcal{U}(V)$ if and only if $v \notin V$. That such a function exists and can be computed in polynomial time in the size of V and $\mathcal{I}$ is shown in [18, 5]. Define $\mathsf{Path}(ID)$ for a string ID (where the name of the node identifies the path from the root to this node, as in Section 6) as the set of nodes from the root of $\mathcal{U}$ to the leaf v_{ID} (including the root and leaf).

We separate the attribute set of $\mathcal{E}$, reserving some attributes to only be used to manage the time component. Write the attribute set of $\mathcal{E}$ as $\Omega' \cup \Omega$ where $|\Omega'| = 2\log(T)$ and label the components of Ω' as:

$$\Omega' = \{\omega_{i,b} : i \in [\log(T)], b \in \{0,1\}\}$$

and for each node y define (where the string $b(y)$ is comprised of $0, 1$ and $*$'s defined in Section 6 corresponding to the path from the root to this node with $*$'s padding the string to length $\log(T)$) the set s_y as follows. For all $i \in [\log(T)]$:

- If $b(y)[i] = 0$, $\omega_{i,0} \in s_y$ and if $b(y)[i] = 1$, $\omega_{i,1} \in s_y$,
- If $b(y)[i] = *$, then $\omega_{i,0}$ and $\omega_{i,1}$ are both included in s_y.

we will shortly explain the significance of defining this set. Next let P_t be the policy defined by [4]:

$$P_t(S) = 1 \text{ if and only if } \omega_{i,t[i]} \in S \ \ \forall \ \ i \in [\log(T)]$$

where $t[i]$ is the i^{th} bit of t. The important observation about P_t and s_y is that $P_t(s_y) = 1$ if and only if y is an ancestor of the leaf corresponding to t and that injective LSSS matrices suffice to express the policies P_t. The encryption and key generation procedures of $\mathcal{F}$ operate over Ω (which

[3] The variable $\mathcal{U}$ is overloaded here, when used as a function, it returns a subset of nodes of the tree $\mathcal{U}$

[4] Depending on the specific implementation of the access structure in the underlying scheme there may be many ways to represent the policy P_t. Any access structure that realizing this policy may be used in these instances.

removes $2\log(T)$ attributes from the scheme). We now describe how to construct our Revocable Storage KP-ABE scheme $\mathcal{F}$ from $\mathcal{E}$ defined above. We use $\mathcal{T}_t$ as defined in Theorem 6.1.

- **Setup**(1^λ) : Return $\mathcal{E}.\ \mathsf{Setup}(1^\lambda) = (PK, MSK)$
- **KeyGen**(MSK, P, ID): For all $x \in\ \mathsf{Path}(ID)$ set

$$SK^{(0)}_{P,x} = \mathcal{E}.\mathsf{KeyGen}(MSK, 0, P, x).$$

  Return:

$$SK^{(0)}_{P,ID} = \{SK^{(0)}_{P,x} : x \in \mathsf{Path}(ID)\}.$$

- **Encrypt**(PK, M, S, t) where $S \subseteq \Omega$: For all $x \in \mathcal{T}_t$ set:

$$C_{S,x} = \mathcal{E}.\mathsf{Encrypt}(PK, M, S \cup s_x).$$

  Return:

$$C_{S,t} = \{C_{S,x} : x \in \mathcal{T}_t\}.$$

- **KeyUpdate**(MSK, rl, t): For all $x \in \mathcal{U}(rl)$ set:

$$SK^{(1)}_{P_t,x} = \mathcal{E}.\ \mathsf{KeyGen}(MSK, 1, P_t, x).$$

  Return:

$$K_t = \{SK^{(1)}_{P_t,x} : x \in \mathcal{U}(rl)\}.$$

- **Decrypt**$(SK_{P,ID}, K_{t'}, C_{S,t})$: If $ID \notin rl$ when $K_{t'}$ was generated, there is some $x \in \mathcal{U}(rl) \cap \mathsf{Path}(ID)$ (by the definition of $\mathcal{U}(V)$). For this x there is:

$$SK^{(0)}_{P,x} \in SK_{P,ID} \text{ and } SK^{(1)}_{P_{t'},x} \in K_{t'}.$$

  Additionally, if $t' \geq t$ there is some $y \in \mathcal{T}_t$ such that y is an ancestor of the leaf $v_{t'}$, which implies $P_{t'}(s_y) = 1$. For this y, take $C_{S,y} \in C_{S,t}$ and return:

$$\mathcal{E}.\mathsf{Decrypt}(SK^{(0)}_{P,x}, SK^{(1)}_{P_{t'},x}, C_{S,y}).$$

  If $P(S) = 1$ then $P(S \cup s_y) = P_{t'}(S \cup s_y) = 1$ implying decryption succeeds.

- **CTUpdate**$(PK, C_{S,t})$: For all $x \in \mathcal{T}_{t+1}$ find $y \in \mathcal{T}_t$ such that y is an ancestor of x. Then there is a $C_{S,y}$ component in $C_{S,t}$. For all such x set:

$$C_{S,x} = \mathcal{E}.\mathsf{Delegate}(PK, C_{S,y}, S \cup s_x)$$

  Which is possible since y being an ancestor of x implies $s_x \subset s_y$. Return:

$$C_{S,t+1} = \{C_{S,x} : x \in \mathcal{T}_{t+1}\}$$

We now describe how security of the underlying $\mathcal{E}$ implies security of $\mathcal{F}$ in the Revocable KP-ABE security game. That $\mathsf{CTUpdate}$ is correct can be observed simply and it therefore only remains to argue that the above is a secure Revocable KP-ABE scheme.

Proof. Let A be such that RKP-SECURITY$_A$ is non-negligible, we will construct an A' such that PIECEWISE KP-ABE$_{A'}$ is non-negligible. We will modify the PIECEWISE security game slightly and give an A' with non-negligible advantage when instead of a single challenge query, the adversary gives a pair of messages (M_0, M_1) as well as a tuple of sets $(S_1^*, S_2^*, \ldots, S_\rho^*)$ and is returned the tuple: $\{\mathsf{Encrypt}(PK, M_b, S_i^*) : i \in [\rho]\}$ by the challenger with the restriction that all S_i^* obey the restriction placed on S^* in the standard game. This implies security in the standard PIECEWISE KP-ABE security game by a standard hybrid argument.

A' begins by initializing the PIECEWISE KP-ABE security game and forwarding PK to A. To respond to an $SK(P, ID)$ query A' sends a query $(0, P, x)$ to its key generation oracle for all $x \in \mathsf{Path}(ID)$ which drawn from the same distribution as the construction above. Similarly, for all queries $K(t, rl)$, A' sends a query $(1, P_t, x)$ for all $x \in \mathcal{U}(rl)$ to its key generation oracle to simulate the key update information.

When A makes a challenge query (M_0, M_1, S^*, t^*) in order the simulate this, A' in the modified game we described above will send as its challenge query (M_0, M_1) and the tuple $S^* \cup s_x$ for al $x \in \mathcal{T}_{t^*}$. Notice that by responding to the queries in this fashion we have perfectly simulated the expected distribution for A. It remains only to show that as long as A does not submit an invalid query that causes the experiment to automatically output 0 our A' has not submitted an invalid query to the PIECEWISE KP-ABE oracle in the modified game.

Take any $S^* \cup s_x$ in the challenge query that A' makes as described above. Take any $y \in \mathcal{U}$ we now claim that either for either $b = 0$ or $b = 1$ all queries of the type (b, P, y) that A' makes while simulating the queries of A, $P(S^* \cup s_x) = 0$.

First consider the case where for some descendent leaf ID of y (which is a $\mathcal{I}$ bit string) that A makes a query to $SK(P, ID)$ with $P(S^*) = 1$. In this case by the guarantee on the queries of A for all $K(t, rl)$ queries with $t \geq t^*$, $ID \in rl$. This implies that for all queries of the type $(1, P_t, z)$ that A' makes, either $t < t^*$ (in which case $P_t(S^* \cup s_x) = 0$ since P_t does not depend on S^* and $P_t(s_x) = 0$ since x is not an ancestor of t because $x \in \mathcal{T}_{t^*}$) or $ID \in rl$ which implies no ancestor of ID is contained in $\mathcal{U}(rl)$ and therefore, z is not an ancestor of ID and therefore $z \neq y$. So we have established that in this case, all $(1, P, y)$ queries have the property that $P(S^* \cup s_x) = 0$ as desired.

Next, consider the case where A does not make a query to a descendent leaf ID of y with $SK(P, ID) = 1$. Then, in simulating A' only makes $(0, P, y)$ queries where $P(S^*) = 0$ which implies $P(S^* \cup s_x) = 0$ (since the policies for the Revocable scheme are only over Ω). This shows the desired statement in both cases, proving the theorem. □

Using the construction given in Section 8 we conclude:

Theorem 7.2. *Under Assumptions 1. 2. and 3. (defined below), when $\mathcal{E}$ is set to be the scheme given in Section 8 the above $\mathcal{F}$ is a secure Revocable Storage KP-ABE scheme supporting injective LSSS Matrices.*

8 Piecewise KP-ABE Construction

We now introduce the assumptions we will be using to build our scheme. These are the same assumptions from the fully secure ABE construction due to Lewko et al. [13]. The notation of giving $\mathbb{G}$ as input will imply that this includes a description of the group, the bilinear pairing operation and the order of the group. We first describe a stateful construction where the key generation algorithm is allowed to maintain state between various invocations for ease of exposition. Through standard methods, we will briefly outline at the end how this state requirement can be removed by using a PRF.

Assumption 1. Let $\mathbb{G}$ be a cyclic group of size $N = p_1 p_2 p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$. Consider the event that we generate $g \leftarrow \mathbb{G}_{p_1}, X_3 \leftarrow \mathbb{G}_{p_3}, T_1 \leftarrow$

$\mathbb{G}_{p_1,p_2}, T_2 \leftarrow \mathbb{G}_{p_1}$ uniformly at random. Assumption 1. states that for any probabilistic polynomial time A:

$$|\Pr[A(\mathbb{G}, g, X_3, T_1) = 0] - \Pr[A(\mathbb{G}, g, X_3, T_2) = 0]$$

is negligible in λ.

Assumption 2. Let $\mathbb{G}$ be a cyclic group of size $N = p_1p_2p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$. Consider the event that we generate $g, X_1 \leftarrow \mathbb{G}_{p_1}$, $X_2, Y_2 \leftarrow \mathbb{G}_{p_2}$, $X_3, Y_3 \leftarrow \mathbb{G}_{p_3}$, $T_1 \leftarrow \mathbb{G}$ and $T_2 \leftarrow \mathbb{G}_{p_1p_3}$ uniformly at random. Assumption 2. states that for any probabilistic polynomial time A (if we let $D = (\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3)$:

$$|\Pr[A(D, T_1) = 0] - \Pr[A(D, T_2) = 0]|$$

is negligible in λ.

Assumption 3. Let $\mathbb{G}$ be a cyclic group of size $N = p_1p_2p_3$ with bilinear map e selected according to the group generator $\mathcal{G}(1^\lambda)$ with target group $\mathbb{G}_T$. Consider the event that we generate $g \leftarrow \mathbb{G}$, $\alpha, s \leftarrow \mathbb{Z}_N$, $X_2, Y_2, Z_2 \leftarrow \mathbb{G}_{p_2}$, $X_3 \leftarrow \mathbb{G}_{p_3}$ uniformly at random. Finally select $T_1 = e(g, g)^{\alpha s}$ and $T_2 \leftarrow \mathbb{G}_T$. Assumption 3. states that for any probabilistic polynomial time A, if we let D denote $D = (\mathbb{G}, \mathbb{G}_T, g, g^\alpha X_2, X_3, g^s Y_2, Z_2)$, then:

$$|\Pr[A(D, T_1) = 0] - \Pr[A(D, T_2) = 0]|$$

is negligible in λ.

Theorem 8.1. *The KP-ABE scheme given below has piecewise key generation, supports injective LSSS policies and has ciphertext delegation if Assumptions 1, 2, 3. hold.*

Since $\mathbb{G}$ is cyclic, it has unique subgroups of size p_1, p_2 and p_3 which we call $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}$ and $\mathbb{G}_{p_3}$ respectively.

Note that the proof of security requires significant divergence from [13] but the construction is a fairly straightforward adaptation. For our construction and security proof we will assume that the access policy (A, ρ) has A a $n \times l$ matrix where n and l are fixed for convenience and the obvious adaptation of the construction and proof suffices to cover the more general case where the dimensions are allowed to vary. We let the vector 1 stand as shorthand for $1 \circ 0 \circ 0 \ldots 0$ when the dimension is specified by context. Note that below we will assume the decryption algorithm has the public key as input, which was not part of our original definition. Our construction below can be modified to fit our formal definition by including the public key as part of all secret keys, however we produce it where they are separate to give the leanest instantiation.

Setup$(1^\lambda) \rightarrow (PK, MSK)$: Choose a bilinear group of order $N = p_1p_2p_3$ (three distinct primes) according to $\mathcal{G}(1^\lambda)$. Then choose $\alpha \leftarrow \mathbb{Z}_N$ and $g \leftarrow \mathbb{G}_{p_1}$ uniformly. For each $i \in \Omega$, $s_i \leftarrow \mathbb{Z}_N$ uniformly at random. Pick $X_3 \in \mathbb{G}_{p_3}$ uniformly with $X_3 \neq 1$ and set:

$$PK = (N, g, e(g, g)^\alpha, X_3, T_i = g^{s_i} \text{ for all } i \in \Omega) \quad , \quad MSK = (\alpha, PK).$$

KeyGen$(MSK, b, (A, \rho), U, PK)$. If α_U has not been generated yet, generate it uniformly from $\mathbb{Z}_N$ and store it. For each row of A_i of A choose a random $r_i \leftarrow \mathbb{Z}_N$ and $W_i, V_i \in \mathbb{G}_{p_3}$. If $b = 0$ let u be a random l dimensional vector over $\mathbb{Z}_N$ such that $1 \cdot u = \alpha_U$ otherwise, sample it subject to the restriction that $1 \cdot u = \alpha - \alpha_U$. For all $i \in [n]$ set:

$$K_i^{(1)} = g^{A_i \cdot u} T_{\rho(i)}^{r_i} W_i \quad , \quad K_i^{(2)} = g^{r_i} V_i$$

and return $SK_{U,(A,\rho)}^{(b)} = \{K_i^{(1)}, K_i^{(2)} : i \in [n]\}$.

Encrypt(PK, M, S). Choose $s \leftarrow \mathbb{Z}_N$ at random. Return:

$$CT_S = (C = Me(g,g)^{\alpha s}, C_0 = g^s, (C_i = T_i^s : i \in S)) .$$

Decrypt$(CT_S, SK^{(0)}_{U,(A,\rho)}, SK^{(1)}_{U,(B,\beta)}, PK)$. Let $\omega_i \in \mathbb{Z}_N$ be such that $\sum_i \omega_i A_i = 1$.

Label the components of $SK^{(0)}_{U,(A,\rho)}$ as $K_i^{(1)}, K_i^{(2)}$ for $i \in [n]$ and set:

$$\prod_{\rho(i)\in S} \frac{e(C_0, K_i^{(1)})^{\omega_i}}{e(C_{\rho(i)}, K_i^{(2)})^{\omega_i}} = \prod_{\rho(i)\in S} \frac{e(g,g)^{s\omega_i A_i \cdot u} e(g, T_{\rho(i)}^{sr_i\omega_i})}{e(g, T_{\rho(i)})^{sr_i\omega_i}} = e(g,g)^{s\alpha_U}$$

And using the identical procedure with $SK^{(1)}_{U,(B,\beta)}$ recovers $e(g,g)^{s(\alpha-\alpha_U)}$ allowing recovery of $e(g,g)^{s\alpha}$ and M as $C/e(g,g)^{s\alpha}$.

8.1 Removing State

Notice that in our construction above the only place that we use the fact that the key generation algorithm retains state between invocations is keeping track of the α_U values which are generated uniformly from $\mathbb{Z}_N$. It is possible to remove this requirement in a standard manner by using a PRF F by setting $\alpha_U = F(U)$. The security of this scheme follows from the security of the one we describe by the security of F as a PRF. For the proof of security we will stick to the above definition where α_U is generated uniformly and retained between invocations for ease of notation.

8.2 Proof of Security

We now define semi-functional ciphertexts and keys for our purposes. While the ciphertext modification is unchanged from the original scheme, the semi-functional key description is significantly changed. We will use the notation $X := f(X)$ to mean the value of a variable X is changed to be $f(X)$ (where the input to X is the old value of X). Throughout this section the notation $x \leftarrow X$ for a set X will mean x is uniformly sampled from X unless there is further qualification.

Semi-Functional Ciphertext. Let g_2 be a generator of $\mathbb{G}_{p_2}$ (the generation procedure is the same no matter which generator is chosen). Generation for a semi-functional ciphertext generates $\mathsf{Encrypt}(PK, M, S) = (C, C_0, (C_i : i \in S))$ and then generates $c \leftarrow \mathbb{Z}_N$ and $z_i \leftarrow \mathbb{Z}_N$ for all $i \in S$ and sets:

$$C_0 := C_0 g_2^c \quad , \quad C_i := C_i g_2^{z_i} \ \ \forall \ \ i \in S.$$

and returns $(C, C_0, (C_i : i \in S))$. This is equivalent to generating $s, c, z_i \leftarrow \mathbb{Z}_N$ uniformly for all $i \in S$ and setting:

$$C = Me(g,g)^{\alpha s} \ \ , \ \ C_0 = g^s g_2^c \ \ , \ \ C_i = T_i^s g_2^{z_i} \text{ for all } i \in S.$$

In general we will use the 'updating the values' notation used in the first definition above as we believes it gives the clearest intuition as to how the generation process is modified.

Semi-Functional Key. An identifier U has keys generated *semi-functionally* if the key generation process is modified as follows. Generate $\beta_{U,0}, \beta_{U,1} \leftarrow \mathbb{Z}_N$ uniformly at random once for this U (if this U has had a semi-functional key generated earlier, re-use those β values). A key $SK^{(b)}_{U,(A,\rho)}$ is generated semi-functionally by first calling the real generation procedure

$$\mathsf{KeyGen}(MSK, b, (A,\rho), U, PK)$$

then sampling a random vector $v \in \mathbb{Z}_N^l$ subject to the restriction that $v \cdot 1 = \beta_{U,b}$ and setting:

$$K_i^{(1)} := K_i^{(1)} \times g_2^{A_i \cdot v}$$

for all $i \in [n]$.

Definition 8.2. ($\text{GAME}_{\text{REAL}}$) *An adversary in* $\text{GAME}_{\text{REAL}}$ *is interacting with the actual functionality as described in* PIECEWISE KPABE SECURITY$_A$.

Definition 8.3. (GAME_0) *The response to the adversary's queries in* GAME_0 *will differ from* $\text{GAME}_{\text{REAL}}$ *only in the challenge ciphertext phase. In* GAME_0*, the challenge ciphertext will be generated as a semi-functional ciphertext.*

Lemma 8.4. *Let* $\epsilon_{\text{REAL}}, \epsilon_0$ *be the advantage of an adversary* A *in* $\text{GAME}_{\text{REAL}}$ *and* GAME_0 *respectively . If Assumption 1. holds then* $|\epsilon_{\text{REAL}} - \epsilon_0|$ *is negligible in* λ.

Proof. Let $(\mathbb{G}, g, X_3, T)$ be a uniform instance from Assumption 1. where T is either from $\mathbb{G}_{p_1 p_2}$ or $\mathbb{G}_{p_1}$ with equal probability, we will show how to reply to the adversary's queries given only this information such that if it comes from the first distribution we are responding as a uniform instance of $\text{GAME}_{\text{REAL}}$ and otherwise, we are responding as a uniform instance of GAME_0.

Having a generator of $\mathbb{G}_{p_3}$,X_3, and g suffices to reproduce the honest Setup and KeyGen queries completely by choosing $\alpha, s_i \leftarrow \mathbb{Z}_N$ uniformly at random for all $i \in \Omega$. The challenge ciphertext will be generated with the following modification:

On input M_0, M_1, S the algorithm will choose $b \leftarrow \{0,1\}$ at random. It will then set:

$$C = M_b e(g^\alpha, T) \ , \ C_0 = T \ , \ C_i = T^{s_i} \ , \ \forall i \in S$$

Then, if $T \in \mathbb{G}_{p_1}$ this is a properly distributed normal ciphertext but if $T \in \mathbb{G}_{p_1 p_2}$ the above is a property distributed semi-functional ciphertext (the $\mathbb{G}_{p_1}$ and $\mathbb{G}_{p_2}$ subgroups are orthogonal to each other in e), as desired since the value of any s_i value modulo p_2 is completely undetermined from the point of view of a distinguisher since it's only used as an exponent for elements in $\mathbb{G}_{p_1}$ in the public key. □

Definition 8.5. (GAME_k) *For this game, the keys corresponding to the first* k *identifiers that are queried by the adversary have keys generated semi-functionally and the challenge ciphertext is semi-functional.*

Notice that above we are not concerned with the strict order in which the adversary makes his queries, but instead with the first k unique identifiers he queries. This means that if an identifier was among the first k that A has made a key generation query to, all subsequent keys for this identifier are generated semi-functionally. We let q be an upper bound on the number of queries the adversary makes.

Lemma 8.6. *Let* ϵ_k *be the advantages of an adversary* A *in game* GAME_k*. For any* $k \in [q]$*, if Assumption 2. holds then* $|\epsilon_k - \epsilon_{k+1}|$ *is negligible in* λ.

This statement will require a sequence of hybrids. We define an adversary A to be Type 1. if for the k^{th} identifier that it queries a key for (call this identifier U) and any $\mathsf{KeyGen}(MSK, b, (A, \rho), U)$ query it makes in the security game with $b = 0$, $(A, \rho)(S^*) = 0$ where S^* is the attribute set of the challenge ciphertext. Similarly, we define A to be Type 2. if $(A, \rho)(S^*) = 0$ whenever $b = 1$. Note that while the adversary's queries must fall in one of the above categories in order for the security game to not automatically output 0, the actual adversary does not fall in either of these cases. However, we will use this analysis to conclude our result for general adversaries.

Definition 8.7. *We define the game $\mathcal{H}_i$ as follows. The challenge ciphertext and keys for the first $k-1$ identifiers queried are generated semi-functionally while the keys for the identifiers after and including the $k+1^{st}$ identifier are generated normally. The keys to the k^{th} identifier U are modified as follows:*

For the first $i-1$ queries to $\mathsf{KeyGen}(MSK, 0, (A, \rho), U)$, after generating the key from the normal distribution and then generate $u_2 \leftarrow \mathbb{Z}_N^l$ and set for all $j \in [n]$:

$$K_j^{(1)} := K_j^{(1)} \times g_2^{A_j \cdot u_2}$$

For the i^{th} query to $\mathsf{KeyGen}(MSK, 0, (A, \rho), U)$, generate the key from the normal distribution and then generate $u_2 \leftarrow \mathbb{Z}_N^l$, $\gamma_j \leftarrow \mathbb{Z}_N$ and set for all $j \in [n]$:

$$K_j^{(1)} := K_j^{(1)} \times g_2^{A_j \cdot u_2 + \gamma_j s_{\rho(j)}} \quad K_j^{(2)} := K_j^{(2)} g_2^{\gamma_j}$$

Following the i^{th} query, or to $\mathsf{KeyGen}(MSK, 1, (A, \rho), U)$ for the k^{th} identifier U, no modification is made to the normal key generation algorithm.

Definition 8.8. *Define the game $\mathcal{I}_i$ as $\mathcal{H}_i$ where the i^{th} query to $\mathsf{KeyGen}(MSK, 0, (A, \rho), U)$ is modified by setting $\gamma_i = 0$ rather than generating it uniformly over $\mathbb{Z}_N$.*

For the following analysis let ν_i denote the advantage of the adversary in the security game $\mathcal{H}_i$ and μ_i the advantage of the adversary in $\mathcal{I}_i$:

Lemma 8.9. *If A is of Type 1. then $|\mu_{i-1} - \nu_i|$ is negligible in λ if Assumption 2. holds.*

Proof. We will describe how to embed an instance of the security game from Assumption 2. into either $\mathcal{I}_{i-1}$ or $\mathcal{H}_i$ depending on the distribution chosen in the Assumption 2. experiment. In the security game for Assumption 2. we are given $\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3, T$ where T is either in $\mathbb{G}$ or $\mathbb{G}_{p_1p_3}$. Generate $\alpha \leftarrow \mathbb{Z}_N$ and $s_i \leftarrow \mathbb{Z}_N$ for each attribute $i \in \Omega$. As the public key, generate and send to the adversary:

$$PK = \{N, g, e(g,g)^\alpha, T_i = g^{s_i} \text{ for all } i \in \Omega\}$$

The challenge ciphertext with attribute set S will be generated as:

$$C = M_\beta e(g, g^s X_2)^\alpha \ , \ C_0 = g^s X_2 \ , \ C_i = (g^s X_2)^{s_i} \ \ \forall i \in S.$$

- **The first $k-1$ identifiers queried to $\mathsf{KeyGen}(MSK, b, U, (A, \rho))$.** If $\alpha_U, \beta_{U,0}, \beta_{U,1}$ has not yet been generated, choose them uniformly in $\mathbb{Z}_N$. For all such queries choose a random value $r_j \in \mathbb{Z}_N$, and uniformly generate $W_j, V_j \in \mathbb{G}_{p_3}$ (using X_3) for each row, and choose $u \in \mathbb{Z}_N^l$ uniformly subject to the restriction that if $b = 0$, $u \cdot 1 = \alpha_U$ and otherwise $u \cdot 1 = \alpha - \alpha_U$. Generate additionally $u_2 \leftarrow \mathbb{Z}_N^l$ so that $u_2 \cdot 1 = \beta_{U,b}$ and set:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_i} W_i (Y_2Y_3)^{A_j \cdot u_2} \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

- **The k^{th} identifier where $b = 0$** . If α_U has not yet been generated, choose it uniformly in $\mathbb{Z}_N$. The first $i-1$ times $\mathsf{KeyGen}(MSK, 0, (A, \rho), U)$ is called, choose u a random vector over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = \alpha_U$ and $u_2 \leftarrow \mathbb{Z}_N^l$, $V_j, W_j \leftarrow \mathbb{G}_{p_3}$, $r_j \leftarrow \mathbb{Z}_N$ for all $j \in [n]$ and set:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j (Y_2Y_3)^{A_j \cdot u_2} \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

  For the i^{th} query, choose u a random vector over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = \alpha_U$ and generate v randomly over $\mathbb{Z}_N^l$ so that $v \cdot 1 = 0$ and $V_j, W_j \leftarrow \mathbb{G}_{p_3}$, $r_j \leftarrow \mathbb{Z}_N$ for all $j \in [n]$ and set:

$$K_j^{(1)} = g^{A_j \cdot u} T^{A_j \cdot v} W_j T^{r_j s_{\rho(j)}} \quad , \quad K_j^{(2)} = T^{r_j} V_j$$

For queries after the i^{th} query choose u a random vector over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = \alpha_U$ and $V_j, W_j \leftarrow \mathbb{G}_{p_3}$, $r_j \leftarrow \mathbb{Z}_N$ for all $j \in [n]$ set:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

- **After the k^{th} identifier or the k^{th} identifier with $b = 1$.** If α_U has not yet been generated, choose it uniformly in $\mathbb{Z}_N$. Choose u a random vectors over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = \alpha_U$ if $b = 0$ or $u \cdot 1 = \alpha - \alpha_U$ if $b = 1$ and $V_j, W_j \leftarrow \mathbb{G}_{p_3}$, $r_i \leftarrow \mathbb{Z}_N$ for all $j \in [n]$ set:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

Note that when $T \in \mathbb{G}_{p_1 p_3}$ the above method is distributed identically to an instance of $\mathcal{I}_{i-1}$. However, when $T \in \mathbb{G}$ it is modified in that for the i^{th} query rather than each $K_j^{(1)}$ component being multiplied by $g_2^{A_j \cdot u_2 + r_j s_{\rho(j)}}$ for a randomly generated u_2 it is multiplied by $T_{\mathbb{G}_{p_2}}$ (the $\mathbb{G}_{p_2}$ part of T) raised to $A_j \cdot u' + r_j s_{\rho(j)}$ where u' is generated uniformly at random subject to the restriction that $1 \cdot u' = 0$ (notice that the fact that u' is used as an exponent in $\mathbb{G}_{p_1}$ does not affect this analysis as its value as an exponent in $\mathbb{G}_{p_2}$ is independent of its value as an exponent in $\mathbb{G}_{p_1}$). This is different from the actual generation procedure in $\mathcal{I}_i$ where u' is generated as a truly uniform vector, however we will show that these distributions are actually identical from the point of view of A because of the queries the adversary is allowed to make.

Notice that since $s_{\rho(i)}$ is not used as an exponent in $\mathbb{G}_{p_2}$ for attributes not in the challenge ciphertext, this is equivalent as when all j such that $\rho(j)$ is not in the challenge ciphertext is simply multiplied by a uniform element in $\mathbb{G}_{p_2}$ and all rows that do have $\rho(j)$ in the challenge ciphertext are multiplied by $T_{\mathbb{G}_{p_2}}$ raised to $A_i \cdot u' + r_i s_{\rho(j)}$, since 1 is not in the span of these A_i values, generating the challenge ciphertext by this procedure is equivalent to generating it by generating u' to be uniformly chosen vector as we demonstrate below.

We now argue that for a set of rows J, if $1 \notin \mathsf{Span}(A_j : j \in J)$, then the distributions on $(A_j \cdot u' : j \in J)$ is equivalent when either u' is chosen as a uniformly random l dimensional vector over $\mathbb{Z}_N$ or as one subject to the restriction that $u' \cdot 1 = 0$. This will suffice to prove that in the case that $T \in \mathbb{G}$ that the above generation process is equivalent to the process from $\mathcal{I}_i$.

Notice that for any $|J|$ dimensional vector $\vec{z}$, if we let S be the set of vectors $u'(\beta)$ such that $\vec{z}[j] = A_j \cdot u'$ for all $j \in J$ and $u' \cdot 1 = \beta$, then by using f an l dimensional vector orthogonal to all A_j such that $1 \cdot A_j \neq 0$, there is a one-to-one correspondence between $u'(\beta)$ and any other $u'(\beta')$. Then we have that the probability that $A_j \cdot u'$ takes a fixed set of values for all $j \in J$ restricted to the condition that $u' \cdot 1 = 0$ is $u'(0)/p_2^{l-1}$. Similarly, the probability it takes this set of values with no restrictions is $\sum_{\beta \in \mathbb{Z}_{p_2}} u'(\beta)/p_2^l$. By our previous observation on the equivalence of $u'(\beta)$ and $u'(\beta')$ for any $\beta \neq \beta'$ these quantities are equal, as desired. We note that a similar argument is presented in [13].

□

Lemma 8.10. *If A is of Type 1. then $|\mu_i - \nu_i|$ is negligible in λ if Assumption 2 holds.*

Proof. The only step that changes from the previous reduction is for the i^{th} query with $b = 0$ to the k^{th} identifier which is now generated as follows:

- **The k^{th} identifier**. For the i^{th} query where $b = 0$, choose u a random vector over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = \alpha_U$ and generate $v \leftarrow \mathbb{Z}_N^l$ and $V_j, W_j \leftarrow \mathbb{G}_{p_3}$, $r_i \leftarrow \mathbb{Z}_N$ for all $j \in [n]$ set:

$$K_j^{(1)} = g^{A_j \cdot u} (Y_2 Y_3)^{A_j \cdot v} W_j T^{r_j s_{\rho(j)}} \quad , \quad K_j^{(2)} = T^{r_j} V_j$$

The above is a uniform instance of $\mathcal{H}_i$ if $T \in \mathbb{G}$ and $\mathcal{I}_i$ if $T \in \mathbb{G}_{p_1 p_3}$ as desired.

□

Definition 8.11. *Define the game $\mathcal{J}$ as a modification of $\mathcal{I}_q$ as follows. For the k^{th} identifier U, rather than generating u_2 as a random vector for every* $\mathsf{KeyGen}(MSK, PK, 0, (A, \rho), U)$ *query, generate a fixed value $\beta_{U,0} \in \mathbb{Z}_N$ randomly for U (to be re-used across all such queries) and generate u_2 uniformly over $\mathbb{Z}_N^l$ subject to the restriction that $u_2 \cdot 1 = \beta_{U,0}$ rather than generating it completely uniformly.*

Let η be the advantage of the advantage of the adversary A in $\mathcal{J}$.

Lemma 8.12. *If A is of Type 1. then $|\mu_q - \eta|$ is negligible in λ if Assumption 2 holds.*

We remind the reader of the generation procedure in $\mathcal{I}_q$ and $\mathcal{J}$. In $\mathcal{I}_q$ and $\mathcal{J}$ the ciphertext and keys for the first $k-1$ identifiers are all semi-functional and the keys corresponding to the $k+1^{st}$ and following identifiers are generated according to the true distribution. In $\mathcal{I}_q$ and $\mathcal{J}$ the k^{th} identifier has keys generated as below:

- **The k^{th} identifier in $\mathcal{I}_q$.** For $b = 1$ the standard key generation algorithm is used. For $b = 0$, after the standard key generation algorithm, $u_2 \leftarrow \mathbb{Z}_N^l$ (new for every query to this identifier) and after the normal key generation, the K_j components are modified as:

$$K_j^{(1)} := K_j^{(1)} \times g_2^{A_j \cdot u_2}$$

- **The k^{th} identifier in $\mathcal{J}$.** For $b = 1$ the standard key generation algorithm is used. For $b = 0$, if $\beta_{U,0}$ has not yet been generated, it is generated uniformly from $\mathbb{Z}_N$. After the standard key generation algorithm u_2 is generated uniformly over $\mathbb{Z}_N^l$ subject to the restriction that $u_2 \cdot 1 = \beta_{U,0}$ and the K_j components are modified as:

$$K_j^{(1)} := K_j^{(1)} \times g_2^{A_j \cdot u_2}$$

For this we will define modified versions of $\mathcal{I}_i$ and $\mathcal{H}_i$. In both modifications, the net change is only to the $K_j^{(1)}$ components of the keys for the k^{th} identifier where $b = 0$. In $\mathcal{I}'_i$ and $\mathcal{H}'_i$ the values are all drawn as in $\mathcal{I}_i$ and $\mathcal{H}_i$ except that an additional value $\beta_{U,0}$ is chosen uniformly from $\mathbb{Z}_N$ once when the k^{th} identifier U is queried to the key generation algorithm and the below modification is made:

- **Queries to the k^{th} identifier with $b = 0$ for $\mathcal{I}'_i$ or $\mathcal{H}'_i$.** Generate the key as in the $\mathcal{I}_i$ or $\mathcal{H}_i$ respectively. Generate v uniformly over $\mathbb{Z}_N^l$ subject to the restriction that $v \cdot 1 = \beta_{U,0}$ and set for all $j \in [n]$:

$$K_j^{(1)} := K_j^{(1)} g_2^{A_j \cdot v}$$

Notice that $\mathcal{I}'_0 = \mathcal{H}'_0 = \mathcal{J}$ and $\mathcal{I}_q = \mathcal{I}'_q$ since the completely random u_2 factor subsumes the additional v factor introduced in $\mathcal{I}'_q$. Define similarly to before, ν'_i to be the advantage of A in $\mathcal{H}'_i$ and μ_i to be its advantage in $\mathcal{I}'_i$.

Lemma 8.13. *If A is of Type 1. then $|\mu'_{i-1} - \nu'_i|$ is negligible in λ if Assumption 2. holds.*

Proof. The proof proceeds identically to the proof for μ_{i-1} and ν_i with the following modification: At the beginning of the security game generate β uniformly from $\mathbb{Z}_N$. For all queries to the k^{th} identifier where $b = 0$, generate a uniform v such that $v \cdot 1 = \beta$ and after generating the $K_j^{(1)}$ component as in the previous proof, modify it as:

$$K_j^{(1)} := K_j^{(1)} (Y_2 Y_3)^{A_j \cdot v}$$

It's simple to check using the argument from the previous lemma that this provides the same indistinguishability guarantee between $\mathcal{I}'_{i-1}$ and $\mathcal{H}'_i$. □

Similarly, by reproducing the argument from the next lemma we get:

Lemma 8.14. *If A is of Type 1. then $|\mu'_i - \nu'_i|$ is negligible in λ if Assumption 2. holds.*

Combining these lemmas with the previous two allows us to conclude that $\mathcal{I}_0$ is indistinguishable from $\mathcal{I}'_0 = \mathcal{J}$. Let η be the advantage of A in $\mathcal{J}$. We can now conclude:

Corollary 8.15. *If A is of Type 1. then $|\epsilon_{k-1} - \eta|$ is negligible in λ if Assumption 2. holds.*

We now are ready to connect our hybrids to GAME_k:

Lemma 8.16. *If A is of Type 1. then $|\eta - \epsilon_k|$ is negligible in λ if Assumption 2 holds.*

Proof. At a high level, we will be using the $\mathbb{G}_{p_1}$ part of T as g^{α_U}. We generate the public key and challenge ciphertext as in the previous two reductions.

- **The first $k-1$ identifiers queried to KeyGen**$(MSK, b, U, (A, \rho))$. If $\alpha_U, \beta_{U,0}, \beta_{U,1}$ have not yet been generated, choose them uniformly in $\mathbb{Z}_N$. For all such queries pick $r_j \leftarrow \mathbb{Z}_N$, $W_j, V_j \leftarrow \mathbb{G}_{p_3}$ for each row, and choose $u \in \mathbb{Z}_N^l$ uniformly subject to the restriction that if $b = 0$ $u \cdot 1 = \alpha_U$ and otherwise $u \cdot 1 = \alpha - \alpha_U$. Generate additionally u_2 uniformly over $\mathbb{Z}_N^l$ so that $u_2 \cdot 1 = \beta_{U,b}$, and set for all $j \in [n]$:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j (Y_2 Y_3)^{A_j \cdot u_2} \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

- **The k^{th} identifier**.If $\beta_{U,0}$ has not yet been generated, choose it uniformly in $\mathbb{Z}_N$. For queries to KeyGen$(MSK, 0, (A, \rho), U)$. Generate u at random over $\mathbb{Z}_N^l$ so that $u \cdot 1 = 1$ and generate u_2 uniformly so that $u_2 \cdot 1 = \beta_{U,0}$ generate $V_j, W_j \leftarrow \mathbb{G}_{p_3}$ for all $j \in [n]$ and set:

$$K_j^{(1)} = T^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j (Y_2 Y_3)^{A_j \cdot u_2} \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

- For queries to KeyGen$(MSK, 1, (A, \rho), U)$, Generate u, u_2 at random over $\mathbb{Z}_N^l$ so that $u \cdot 1 = \alpha$ and $u_2 \cdot 1 = -1$ generate $V_j, W_j \leftarrow \mathbb{G}_{p_3}$ for all $j \in [n]$ and set:

$$K_j^{(1)} = g^{A_j \cdot u} T^{A_j \cdot u_2} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

- **After the k^{th} identifier**. If α_U has not yet been generate, choose it uniformly in $\mathbb{Z}_N$. Generate u at random over $\mathbb{Z}_N^l$ so that $u \cdot 1 = \alpha_u$ if $b = 0$ and $u \cdot 1 = \alpha - \alpha_U$ if $b = 1$. Generate $r_j \leftarrow \mathbb{Z}_N$, $V_j, W_j \leftarrow \mathbb{G}_{p_3}$ for all $j \in [n]$ and set:

$$K_j^{(1)} = g^{A_j \cdot u} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$

When $T \in \mathbb{G}_{p_1 p_3}$ the above is a uniform instance of $\mathcal{J}$ where the $\mathbb{G}_{p_1}$ part of T is implicitly set to be α_U for the k^{th} identifier U. Similarly, if $T \in \mathbb{G}$ it is a uniform instance of GAME_k where the $\mathbb{G}_{p_1}$ part of T is set to be α_U, the $\beta_{U,0}$ value for the k^{th} identifier U is actually set to be the Y_2 logarithm of the $\mathbb{G}_{p_2}$ part of T plus the $\beta_{U,0}$ value in the construction and $-\beta_{U,1}$ is implicitly set to be the Y_2 logarithm of the $\mathbb{G}_{p_2}$ part of T. □

Therefore we have that GAME_{k-1} is indistinguishable from GAME_k to Type 1. adversaries - and through a symmetric repetition of the above argument that it is also indistinguishable to Type 2. adversaries (where the $\mathcal{H}_i, \mathcal{I}_i, \mathcal{H}'_i, \mathcal{I}'_i$ hybrids now modify the generation procedure when $b = 1$). If there was a general adversary A that could distinguish between GAME_k and GAME_{k+1} it would be possible to create either a Type 1. or Type 2. adversary that could also distinguish by creating the two adversaries A_1 and A_2 where A_1 aborts is A ever makes a query that violates the condition of being Type 1. and guesses randomly and similarly for A_2. Therefore the above implies that GAME_k and GAME_{k+1} is indistinguishable for general adversaries. By a hybrid argument we have:

Corollary 8.17. *$|\epsilon_0 - \epsilon_q|$ is negligible in λ if Assumption 2. holds.*

We briefly remind the reader of the modified key and ciphertext generation in GAME_q.

- **Setup and the Public Key.** Choose a bilinear group of order $N = p_1p_2p_3$ and $\alpha \leftarrow \mathbb{Z}_N$ and $g \leftarrow \mathbb{G}_{p_1}$, $g_2 \leftarrow \mathbb{G}_{p_2}$ uniformly. For each $i \in \Omega$ pick $s_i \leftarrow \mathbb{Z}_N$, $X_3 \leftarrow \mathbb{G}_{p_3}$ uniformly and set:
$$PK = (N, e(g,g)^\alpha, X_3, T_i = g^{s_i})$$
- **Key Generation.** On a query to $\mathsf{KeyGen}(MSK, b, U, (A,\rho), PK)$ proceed as follows: If α_U or $\beta_{U,b}$ has not been generated yet, generate it and store it. Pick u and v random vectors over $\mathbb{Z}_N^l$ so that $u \cdot 1 = \alpha_U$ if $b = 0$ and $u \cdot 1 = \alpha - \alpha_U$ if $b = 1$ and v so that $v \cdot 1 = \beta_{U,b}$. Pick W_i, V_i randomly for all $i \in [n]$ from $\mathbb{G}_{p_3}$ and set, for all $i \in [n]$:
$$K_i^{(1)} = g^{A_i \cdot u} g_2^{A_i \cdot v} T_{\rho(i)}^{r_i} W_i \quad , \quad K_i^{(2)} = g^{r_i} V_i$$
- **The Challenge Ciphertext.** With challenge message M and attribute set S. Pick $s, c, z_i \leftarrow \mathbb{Z}_N$ at random for all $i \in S$ and return:
$$C = Me(g,g)^{\alpha s} \quad , \quad C_0 = g^s g_2^c \quad , \quad C_i = T_i^s g_2^{z_i} \forall i \in S$$

Definition 8.18. $\text{GAME}_{\text{FINAL}}$ *Let $\text{GAME}_{\text{FINAL}}$ be a modification of GAME_q where a random message is encrypted in the challenge phase rather than the challenge message.*

Lemma 8.19. *For any A, $|\epsilon_q - \epsilon_{\text{FINAL}}|$ is negligible if Assumption 3. holds.*

Proof. Recall that in the Assumption 3. security game the distinguisher is given $g, g^\alpha X_2, X_3, g^s Y_2, Z_2, T$ where $\alpha, s \leftarrow \mathbb{Z}_N$, $X_2, Y_2, Z_2 \leftarrow \mathbb{G}_{p_2}$, $X_3 \in \mathbb{G}_{p_3}$ and T is either $e(g,g)^{\alpha s}$ or randomly chosen from $\mathbb{G}_T$. Our reduction proceeds as follows:

- **Setup and the Public Key.** Generate the public key as:
$$PK = (N, g, X_3, e(g,g)^\alpha = e(g, g^\alpha X_2))$$
- **Private Key Queries.** On a query to generate $\mathsf{KeyGen}(MSK, b, (A,\rho), U)$ proceed as follows: If $\beta_{U,b}$ or α_U hasn't been chosen yet, generate it uniformly from $\mathbb{Z}_N$ and pick u, u_2 uniformly over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = 1$ and $u_2 \cdot 1 = \beta_{U,b}$ and pick V_j, W_j randomly in $\mathbb{G}_{p_3}$ for all $j \in [n]$. If $b = 0$ set for all $j \in [n]$:
$$K_j^{(1)} = g^{\sum_{j'=2}^{l} A_j[j'] u[j']} (g^{\alpha_U})^{A_j[1]} Z_2^{A_j \cdot u_2} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$
If $b = 1$ follow the same procedure but set:
$$K_j^{(1)} = g^{\sum_{j'=2}^{l} A_j[j'] u[j']} \left(g^{-\alpha_U} (g^\alpha X_2)\right)^{A_j[1]} Z_2^{A_j \cdot u_2} T_{\rho(j)}^{r_j} W_j \quad , \quad K_j^{(2)} = g^{r_j} V_j$$
- **The Challenge Ciphertext.** To generate the challenge ciphertext with messages (M_0, M_1) and attribute set S pick a random bit b and set:
$$C = M_b T \quad , \quad C_0 = g^s X_2 \quad , \quad C_i = (g^s X_2)^{s_i} \forall i \in S$$

In the case where $T = e(g,g)^{\alpha s}$ let us analyze the distribution of the exponent in $K_j^{(1)}, K_j^{(2)}$ when $b = 0$. In this case (if we call $X_2 = g_2^c$ and $Z_2 = g_2^d$) the above generation process is equivalent to generating a key from GAME_q where v is chosen so that the $\beta_{U,b}$ value is set to be $\beta_{U,b} + c$ in the new generation procedure. Similarly, if T is a random target group element this is an encryption of a randomly chosen message. □

As the advantage of any adversary in $\text{GAME}_{\text{FINAL}}$ is clearly negligible (it has no dependence on b) we have ϵ_{FINAL} is negligible which proves security by connecting our previous lemmas.

9 Revocable Storage CP-ABE

We now describe the requirement needed for a CP-ABE scheme to imply a revocable CP-ABE scheme and provide a construction. We first formally define security for a revocable CP-ABE scheme. Once again we begin by defining some oracles for use in the security game:

Security Game Oracles. Define the following oracles to use in the security game. These oracles are given access to (PK, MSK) that are generated at the beginning of the security game, and may have been modified since, at the time of the oracle's invocation.

1. The Secret Key Generation oracle $SK(\cdot, \cdot)$ takes as input (S, ID) and return $SK_{S,ID}$ generated from:
$$SK_{S,ID} \leftarrow \mathsf{KeyGen}(MSK, S, ID).$$
2. The Key Update Generation oracle $K(\cdot, \cdot)$ takes as input t and a revocation list rl and returns K_t:
$$K_t \leftarrow \mathsf{KeyUpdate}(MSK, t, rl)$$

The security definition now is similar to the key-policy case.

RCP-Security$_A(1^\lambda)$:

1. The challenger runs $\mathsf{Setup}(1^\lambda) \rightarrow (PK, MSK)$ and returns PK to A;
2. A is given oracle access to $SK(\cdot, \cdot)$, $K(\cdot, \cdot)$ until it signals the query phase is over;
3. After the query phase, A returns (M_0, M_1, P^*, t^*) to the challenger;
4. The challenger picks b a random bit and returns to A:
$$C_{P^*,t^*} \leftarrow \mathsf{Encrypt}(PK, M_b, P^*, t^*);$$
5. A is once again given oracle access to the three oracles above;
6. A returns a bit b'. The Experiment returns 1 if and only if $b' = b$ and the conditions below concerning the adversary's query history are satisfied.

The conditions placed on the adversary's queries is as follows: For any query, $SK(S, ID)$ such that $P^*(S) = 1$, $ID \in rl$ for every query $K(t, rl)$ with $t \geq t^*$.

Definition 9.1. *A Revocable CP-ABE scheme is secure if for any polynomial time adversary A the advantage of this adversary in the* RCP-Security *game:*

$$2 \Pr\left[\text{RCP-Security}_A(1^\lambda) = 1\right] - 1$$

is negligible in λ.

Definition 9.2. (Piecewise Key Generation) *A CP-ABE scheme is said to have* piecewise key generation *with attribute set Ω supporting policies in $\mathcal{P}$, message space $\mathbb{M}$ and identifier length $\mathcal{I}$ if key generation and encryption are modified to the following syntax:*

KeyGen takes as input the master key MSK, a bit b, a set $S \subset \Omega$ and an identifier $U \in \{0,1\}^{\mathcal{I}}$ and returns:

- $\mathbf{KeyGen}(MSK, b, S, U) \to K^{(b)}_{S,U}$.

Decrypt takes two outputs of KeyGen as input instead of one:

- $\mathbf{Decrypt}(C_S, K_0, K_1) \to M$

Definition. ***(Correctness)*** *A CP-ABE scheme with* piecewise key generation *is correct if for any $P \in \mathcal{P}$:*

- $(PK, MSK) \leftarrow \mathbf{Setup}(1^\lambda)$
- $C_P \leftarrow \mathbf{Encrypt}(PK, M, P)$
- $S, T \subset \Omega$ such that $P(S) = 1$ and $P(T) = 1$:

If $\mathsf{KeyGen}(MSK, 0, S, U) \to K^{(0)}_{S,U}$ and $\mathsf{KeyGen}(MSK, 1, T, U) \to K^{(1)}_{T,U}$, then,

$$\mathbf{Decrypt}(K^{(0)}_{S,U}, K^{(1)}_{T,U}, C_{(A,\rho)}) = M.$$

Security for a scheme with *piecewise key generation* now follows similarly to the KP-ABE definition:

Piecewise CPABE Security$_A(1^\lambda)$:

1. The challenger runs $\mathsf{Setup}(1^\lambda) \to (PK, MSK)$ and sends PK to A;
2. A makes queries of the type (b, S, U) for $b \in \{0,1\}, S \subset \Omega$ and $U \in \{0,1\}^{\mathcal{I}}$; The challenger runs $\mathsf{KeyGen}(MSK, b, S, U)$ and returns the key to A;
3. A signals the query phase is over and returns (M_0, M_1, P);
4. The challenger picks $b \xleftarrow{\$} \{0,1\}$ and returns $Enc(PK, M_b, P)$ to the adversary;
5. A has another query phase as previously;
6. A sends a bit b' to the challenger;
7. If for any U, $P(S) = P(T) = 1$ for some $S, T \subset \Omega$ s.t. A has queried $\mathsf{KeyGen}(MSK, 0, S, U)$ and $\mathsf{KeyGen}(MSK, 1, T, U)$ return 0.
8. If $b' = b$ return 1, otherwise return 0.

Definition 9.3. *A CP-ABE scheme with piecewise key generation is secure if for any polynomial time adversary A the advantage of this adversary in the* Piecewise CPABE Security *game:*

$$2\Pr\big[\text{Piecewise CPABE Security}_A(1^\lambda) = 1\big] - 1$$

is negligible in λ.

Revocable Storage CP-ABE from Piecewise CP-ABE with Delegation

Using a CP-ABE scheme with piecewise key generation $\mathcal{E}$ that supports elementary ciphertext manipulations and policies that include injective LSSS matrices, we will build a Revocable Storage CP-ABE scheme $\mathcal{F}$. First, we will split the attribute set of $\mathcal{E}$ as $\Omega \cup \Omega'$ where:

$$\Omega' = \{\omega_{i,b} : i \in [\log(T)], b \in \{0,1\}\}$$

The construction of $\mathcal{F}$ is as follows. Let the tree $\mathcal{U}$ be defined as before of depth $\mathcal{I}$ with all identifiers corresponding to leaf nodes and $\mathcal{U}(rl)$ defined as before, and $\mathcal{T}$ the tree with leaves numbered $[T]$ with the subsets $\mathcal{T}_t$ also defined as before. We now show how to handle the policy and delegation by constructing the 'time-policy' in a delegatable fashion from injective LSSS matrices. Once again we let $r = \log(T)$ and assume for notational convenience that T is a power of two.

Let B be the $2r \times r$ matrix such with 2×2 blocks of the all ones matrix cascading along the diagonal with all zeros elsewhere. Below is an example of this construction when $r = 3$.

$$B = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

Label the rows of the matrix B as $b_{1,0}, b_{1,1}, b_{2,0}, b_{2,1}, \ldots, b_{r,0}, b_{r,1}$ in descending order. The relevant property of the construction of B and this ordering that we will need is that the vector 1 will be in the span of a set of the rows S if and only if for each $i \in [r]$ either $b_{i,0} \in S$ or $b_{i,1} \in S$. We now define the labeling β as taking $\beta(b_{i,d}) \to \omega_{i,d}$ for all rows of B. Note that (B, β) now describe an LSSS policy that is satisfied by a set of attributes S if and only if for each $i \in [r]$, $\omega_{i,b} \in S$ for some $b \in \{0, 1\}$. Notice that (B, β) is injective.

We now describe the time policies corresponding to nodes of the tree. Recall each $y \in \mathcal{T}$ admits a natural string representation of length r where $*$ values are used to pad the description of the path from the root to this node to r bits. We describe the policy (B_y, β_y) now. For each index $i \in [r]$ if i is not $*$ we will eliminate the row $b_{i,\overline{y}[i]}$ from the rows of B (keeping the output of β_y the same as β on all non-deleted rows). As an example if $r = 3$ and $y = 01*$, the matrix B and labeling β_y (represented by the arrows below) are set to be:

$$B_y = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \quad \begin{matrix} \to \omega_{1,0} \\ \to \omega_{2,1} \\ \to \omega_{3,0} \\ \to \omega_{3,1} \end{matrix}$$

Notice that (B_y, β_y) is satisfied by a set of attributes S if and only if for all i such that $y[i] \neq *$, $\omega_{i,y[i]} \in S$ and for each i such that $y[i] = *$ either $\omega_{i,0}$ or $\omega_{i,1}$ is in S. Additionally, any scheme that allows elementary ciphertext manipulations can delegate any $(B_y, \beta_y) \to (B_{y'}, \beta_{y'})$ for any descendent y' of y (as shown in Section 5 by use of the Delete operation). Additionally, notice that $(B_y, \beta_y)(s_z) = 1$ (as defined in Section 7) if and only if y is an ancestor of z.

We now begin the description of our scheme. We use the notation $(A, \rho) \vee (B, \beta)$ for two LSSS policies to mean the LSSS policy that is made by sequentially adding the rows of B with assignment β to the matrix A (and padding both to the same length). Furthermore, if (A, ρ) and (B, β) are injective with disjoint ranges, $(A, \rho) \vee (B, \beta)$ is injective as well and if y' is a descendent of y, $(A, \rho) \vee (B_{y'}, \beta_{y'})$ can be delegated from $(A, \rho) \vee (B_y, \beta_y)$. Notice that in contrast to our use of $\vee$ in the definition, this policy may be satisfied by a set of attributes even if this set doesn't satisfy either of the policies individually, but in our application below we will assign attributes relevant to each half disjointly and therefore, a set will only satisfy the entire policy if it satisfies one of the two halves.

- **Setup**(1^λ): Return $\mathcal{E}.\ \mathsf{Setup}(1^\lambda) \to (PK, MSK)$.
- **KeyGen**(MSK, S, ID): For all $x \in \mathsf{Path}(ID)$ run $\mathcal{E}.\ \mathsf{KeyGen}(MSK, 0, S, x) \to SK^{(0)}_{S,x}$. Return:
$$SK^{(0)}_{S,ID} = \{SK_{S,x} : x \in \mathsf{Path}(ID)\}.$$

- **Encrypt**$(PK, M, (A, \rho), t)$: For each $y \in \mathcal{T}_t$, set:

$$C_{(A,\rho),y} = \mathcal{E}.\ \mathsf{Encrypt}(PK, M, (A, \rho) \vee (B_y, \beta_y)).$$

Return:

$$C_{(A,\rho),t} = \{C_{(A,\rho),y} : y \in \mathcal{T}_t\}.$$

- **KeyUpdate**(MSK, rl, t): For all $x \in \mathcal{U}(rl)$ set:

$$SK^{(1)}_{s_t,x} = \mathcal{E}.\ \mathsf{KeyGen}(MSK, 1, s_t, x) \text{ where } s_t = \{\omega_{i,t[i]} : i \in [r]\}$$

Return $K_t = \{SK^{(1)}_{t,x} : x \in \mathcal{U}(rl)\}$.

- **Decrypt**$(C_{(A,\rho),t}, SK_{S,ID}, K_{t'})$: If user $ID \notin rl$ when $K_{t'}$ was created there is some node of $x \in \mathcal{U}(rl) \cap \mathsf{Path}(ID)$. For this x there is:

$$SK^{(0)}_{S,x} \in SK^{(0)}_{S,ID} \text{ and } SK_{s_{t'},x} \in K_{t'}$$

Additionally if $t' \geq t$ this implies there is some $y \in \mathcal{T}_t$ such that y is an ancestor of t' and therefore $(B_y, \beta_y)(s_{t'}) = 1$. For this y, take $C_{(A,\rho),y} \in C_{(A,\rho),t}$ and return:

$$\mathcal{E}.\mathsf{Decrypt}(SK^{(0)}_{S,x}, SK_{s_{t'},x}, C_{(A,\rho),y})$$

If $(A, \rho)(S) = 1$ then $(A, \rho) \vee (B_y, \beta_y)(S) = (A, \rho) \vee (B_y, \beta_y)(s_{t'}) = 1$ which implies that decryption succeeds.

- **CTUpdate**$(PK, C_{(A,\rho),t})$: For all $x \in \mathcal{T}_{t+1}$ find $y \in \mathcal{T}_t$ such that y is an ancestor of x and such that there is a $C_{(A,\rho),y}$ component in $C_{(A,\rho),t}$. For all such x set:

$$C_{(A,\rho),x} = \mathcal{E}.\mathsf{Delegate}(PK, C_{(A,\rho),y}, (A, \rho) \vee (B_x, \beta_x))$$

And finally return:

$$C_{(A,\rho),t+1} = \{C_{(A,\rho),x} : x \in \mathcal{T}_{t+1}\}$$

Theorem 9.4. *If $\mathcal{E}$ is a secure CP-ABE with piecewise key generation that allows elementary ciphertext manipulations, $\mathcal{F}$ described above is a secure CP-ABE scheme with revocable storage supporting injective LSSS matrices.*

Proof. Let A be an adversary such that RCP-$\text{Security}_A(1^\lambda)$ is non-negligible and we will construct an A' such that Piecewise CP-$\text{ABE}_A(1^\lambda)$ is non-negligible. Once again, similar to the KP-ABE case we consider a modified security game where the adversary returns a tuple of policies $(P^*_1, P^*_2, \ldots, P^*_\rho)$ along with two messages (M_0, M_1) and a random bit b is chosen and the adversary receives an encryption of M_b under each of these policies. Security should hold as long as the identities for any key with sufficient credentials to decrypt any of the challenge policies is on every revocation list equal to or exceeding the challenge time.

A' initializes the Piecewise CP-ABE_A security game and forwards PK to A (note that the attribute set of the Revocable CP-ABE scheme that A is interacting with is Ω and therefore all keys and policies sent by A do not involve Ω'). To respond to a $SK(S, ID)$ query, A' queries $(0, S, x)$ to its key generation oracle for all $x \in \mathsf{Path}(ID)$, simulating the oracle of A. Similarly for all queries $K(t, rl)$, A' sends a query $(1, s_t, x)$ for all $x \in \mathcal{U}(rl)$ and combined them to answer the key update query of A.

To respond to the challenge query of A, which we call (M_0, M_1, P^*, t^*), A' sends as its challenge query (M_0, M_1) and the tuple of policies $P^* \vee (B_y, \beta_y)$ for all $y \in \mathcal{T}_{t^*}$ in the modified security game

we described above. It now remains to only show that for any $x \in \mathcal{U}$ and any $P^* \vee (B_y, \beta_y)$ queried in the challenge phase, A' does not query its oracle on two values $(0, S, x)$ and $(1, S', x)$ such that $P^* \vee (B_y, \beta_y)(S) = P^* \vee (B_y, \beta_y)(S') = 1$.

Fix any $y \in \mathcal{T}_t$, we will analyze each $P^* \vee (B_y, \beta_y)$ separately. Begin by considering an x such that no descendent leaf ID of this node is queried on a set $SK(S, ID)$ such that $P^*(S) = 1$. In this case, $(0, S, x)$ is only queried on sets by A' such that $P^*(S) = 0$ and therefore, since (B_y, β_y) does not depend on the attributes in Ω, $P^* \vee (B_y, \beta_y)(S) = 0$ as desired.

Next, consider an x such that a descendent leaf ID of x is queried on a set $SK(S, ID)$ such that $P^*(S) = 1$. Then, ID must be included on rl for all queries by A to $K(t, rl)$ where $t \geq t^*$. Therefore, for all the $(1, s_t, z)$ queries that A' makes, either $t < t^*$ (in which case $P^* \vee (B_y, \beta_y)(s_t) = 0$ because y is from $\mathcal{T}_{t^*}$ which does not contain an ancestor of t), or $z \neq x$ since no ancestor of ID is included on these queries when $ID \in rl$. Therefore, for all queries $(1, S, x)$ we have $P^* \vee (B_y, \beta_y)(S) = 0$, completing the proof. □

10 Piecewise CP-ABE Construction

In this section we give our construction of a CP-ABE scheme with piecewise key generation from Assumptions 1,2 and 3. Throughout this construction we assume the size of all queried policies are $n \times l$ for notational convenience but this can be extended in a straightforward manner.

Setup$(1^\lambda) \to (PK, MSK)$: Choose a bilinear group of order $N = p_1 p_2 p_3$ (from the definition of Assumptions 1, 2 and 3) according to $\mathcal{G}(1^\lambda)$. Then choose $\alpha, a \leftarrow \mathbb{Z}_N$ and $g \leftarrow \mathbb{G}_{p_1}$ uniformly. For each $i \in \Omega$, pick $s_i \leftarrow \mathbb{Z}_N$ and set $MSK = (\alpha, X_3)$ with $X_3 \leftarrow \mathbb{G}_{p_3}$ such that $X_3 \neq 1$:

$$PK = (N, g, g^a, e(g,g)^\alpha, \{T_i = g^{s_i} \text{ for all } i \in \Omega\})$$

KeyGen(MSK, b, S, U). If α_U has not been generated yet, select it uniformly from $\mathbb{Z}_N$; if it has been for a previous query, re-use this value. Generate $t_U \leftarrow \mathbb{Z}_N$ and $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for each $i \in S$. Let $\gamma = \alpha_U$ if $b = 0$ and $\alpha - \alpha_U$ if $b = 1$. Return $SK^{(b)}_{S,U} = (K_U, K_{S,U}, K^*_U)$ generated as:

$$K_U = g^{t_U} V \quad , \quad K_{S,U} = \{K_{U,i} = T_i^{t_U} W_i : i \in S\} \quad , \quad K^*_U = g^\gamma g^{a t_U} Z.$$

Encrypt$(PK, M, (A, \rho))$. Choose $v \leftarrow \mathbb{Z}^l_N$ and $r_i \leftarrow \mathbb{Z}_N$ for all $i \in [n]$, label the components as $v = (s, v_2, \ldots, v_n)$ and set:

$$C = M e(g,g)^{\alpha s} \quad , \quad C' = g^s$$

$$C_i = g^{a A_i \cdot v} T^{-r_i}_{\rho(i)} \quad , \quad D_i = g^{r_i} \text{ for all } i \in [n]$$

Return:

$$C_{(A,\rho)} = (C, C', (C_i, D_i : i \in [n]))$$

Decrypt$(C_{(A,\rho)}, SK^{(0)}_{S,U}, SK^{(1)}_{T,U})$. The decryption algorithm first computes $\omega_i \in \mathbb{Z}_N$ such that $\sum_{\rho(i) \in S} \omega_i A_i = 1$. Then, writing $SK^{(0)}_{S,U} = (K_U, K_{S,U} = (K_{i,U} : i \in S), K^*_U)$ It generates:

$$\prod_{\rho(i) \in S} (e(C_i, K_U) e(D_i, K_{U,\rho(i)}))^{\omega_i} = e(g,g)^{s a t_U}$$

$$e(K^*_U, C') = e(g^{\alpha_U}, g^s) e(g^{a t_U}, g^s)$$

Which allows us to recover $e(g^{\alpha_U}, g^s)$. Similarly, with $SK_{T,U}$ we can recover $e(g^{\alpha - \alpha_U}, g^s)$ which allows us to recover $e(g,g)^{s\alpha}$ and M as $C / e(g,g)^{s\alpha}$.

The property that the key generation algorithm must retain state between invocations above can be removed through the use of a PRF, as described in the KP-ABE case in Section 8.1.

10.1 Proof of Security

We now begin the proof of security. For this we once again define notions of semi-functionality for ciphertexts and keys.

Semi-Functional Ciphertext. For a semi-functional ciphertext, we add a multiplicative factor to some of the components according to the below algorithm. After computing the ciphertext honestly through $\mathsf{Encrypt}(PK, M, (A, \rho)) = (C, C', (C_i, D_i : i \in [n]))$, make the following modifications. First, generate $c \leftarrow \mathbb{Z}_N$ at random and set:

$$C' := C' g_2^c$$

Next generate a random vector $u \leftarrow \mathbb{Z}_N^l$ and for all $i \in [n]$ generate $\gamma_i \leftarrow \mathbb{Z}_N$ and set:

$$C_i := C_i \times g_2^{A_i \cdot u - \gamma_i s_{\rho(i)}} \quad , \quad D_i := D_i \times g_2^{\gamma_i}$$

The semi-functional ciphertext is now set as $C_{(A,\rho)} = (C, C', (C_i, D_i : i \in [n]))$.

Semi-Functional Key. An identifier U has keys generated *semi-functionally* if the following change is made to all queries to KeyGen. Generate $d_{U,0}, d_{U,1} \leftarrow \mathbb{Z}_N$ once for this U. A key $SK_{S,U}^{(b)}$ is generated semi-functionally by first calling the real generation procedure:

$$\mathsf{KeyGen}(MSK, b, S, U, pk) = (K_U, K_{S,U}, K_U^*)$$

and making the modification:

$$K_U^* := K_U^* \times g_2^{d_{U,b}}$$

Notice that for all queries to the same U for the same b value, the K_U^* component is offset by the same multiplicative factor in $\mathbb{G}_{p_2}$.

Definition 10.1. ($\text{GAME}_{\text{REAL}}$) *An adversary in* $\text{GAME}_{\text{REAL}}$ *is interacting with the actual functionality as described in* PIECEWISE CPABE SECURITY$_A$.

Definition 10.2. (GAME_0) *The response to the adversary's queries in* GAME_0 *will differ from* $\text{GAME}_{\text{REAL}}$ *only in the challenge ciphertext phase. In* GAME_0*, the challenge ciphertext will be generated as a semi-functional ciphertext.*

Throughout the rest of the proof we will let ϵ_i denote the advantage of A in GAME_i.

Lemma 10.3. *If Assumption 1. holds then* $|\epsilon_{\text{REAL}} - \epsilon_0|$ *is negligible in* λ.

Proof. Recall in Assumption 1. the challenger is given $g \leftarrow \mathbb{G}_{p_1}$ and $X_3 \leftarrow \mathbb{G}_{p_3}$ and T that either comes from $\mathbb{G}_{p_1 p_2}$ or from $\mathbb{G}_{p_1}$. We simulate either $\text{GAME}_{\text{Real}}$ or GAME_0 depending on the distribution T is drawn from as follows. Choose $a, \alpha \leftarrow \mathbb{Z}_N$ and$s_i \in \mathbb{Z}_N$ for all $i \in \Omega$. Then, sends A the public parameters:

$$PK = \{N, g, g^a, e(g,g)^\alpha, \{T_i = g^{s_i} : i \in \Omega\}\}$$

which is drawn from the correct distribution and internally set $MSK = \{\alpha, X_3\}$. This suffices to simulate honest key generation keys perfectly. The only difficulty lies in generating the challenge ciphertext. On input (M_0, M_1, P) for the challenge ciphertext, generate $b \leftarrow \{0, 1\}$ and sets:

$$C = M_b e(g^\alpha, T) \quad , \quad C' = T$$

We are implicitly setting the $\mathbb{G}_{p_1}$ part of T to be g^s. Then, pick a random vector $v \in \mathbb{Z}_N^n$ such that $v \cdot 1 = 1$ and generate $r_i \leftarrow \mathbb{Z}_N$ for all $i \in [n]$ and sets:

$$C_i = T^{a A_i \cdot v} T^{-r_i s_{\rho(i)}} \quad , \quad D_i = T^{r_i}$$

Notice that if $T = g^s$ then the above is a properly generated normal ciphertext where the v value in the real encryption algorithm is set to be $s \times v$ and the r_i values in the actual scheme are set to be $s \times r_i$ which are both drawing from the correct distribution. □

Definition 10.4. (GAME_k) *For this game, the keys corresponding to the first k identifiers that are queried by the adversary are generated semi-functionally and the challenge ciphertext is semi-functional.*

Lemma 10.5. *For any $k \in [q]$ if Assumption 2. holds, $|\epsilon_k - \epsilon_{k+1}|$ is negligible in λ.*

We will analyze two restricted types of adversaries separately and later show that we can use this analysis to conclude the final result. A is Type 1 if for the k^{th} identifier U it only queries a attribute sets S to $\mathsf{KeyGen}(MSK, 0, S, U)$ such that $(A^*, \rho)(S) = 0$ where (A^*, ρ) is the policy for the challenge ciphertext.

Similarly, A is Type 2 if for the k^{th} identifier U it only queries attribute sets S to $\mathsf{KeyGen}(MSK, 1, S, U)$ such that $(A^*, \rho)(S) = 0$.

We will show that for either type of A has advantage $|\epsilon_{k-1} - \epsilon_k|$ negligible. Note that this does not immediately imply the theorem statement as the actual A does not fall in either class, but we will use it to prove our lemma.

Definition 10.6. *We define the game $\mathcal{H}_i$ as follows. The challenge ciphertext and keys for the first $k-1$ identifiers queried are generated semi-functionally while the keys for the identifiers after and including the $k+1^{st}$ are generated normally. The keys for the k^{th} identifier are modified as follows:*

For the first $i-1$ queries to $\mathsf{KeyGen}(MSK, 0, S, U)$ generate $(K_U, K_{S,U}, K_U^*)$ from the normal distribution, generate $d_U \leftarrow \mathbb{Z}_N$ (new for each query) and set:

$$K_U^* := K_U^* \times g_2^{d_U}.$$

For the i^{th} query to $\mathsf{KeyGen}(MSK, 0, S, U)$ generate $(K_U, K_{S,U}, K_U^*)$ from the normal distribution and then generate $e_U, f_U \leftarrow \mathbb{Z}_N$ and set:

$$K_U := K_U \times g_2^{e_U} \quad , \quad K_{U,i} := K_{U,i} \times g_2^{e_U s_i} \quad , \quad K_U^* := K_U^* \times g_2^{f_U}$$

Following the i^{th} query, or to $\mathsf{KeyGen}(MSK, 1, (A, \rho), U)$ for the k^{th} identifier U, no modification is made to the normal key generation algorithm.

Definition 10.7. *Define $\mathcal{I}_i$ as a modification to $\mathcal{H}_i$ where instead of e_U being generated uniformly for the i^{th} query with $b = 0$ to the k^{th} identifier, set $e_U = 0$.*

Let ν_i be the advantage of A in the security game $\mathcal{H}_i$ and μ_i its advantage in $\mathcal{I}_i$.

Lemma 10.8. *If A is of Type 1. then $|\mu_{i-1} - \nu_i|$ is negligible in λ for any $i \in [q]$ if Assumption 2. holds.*

Proof. Let $g, X_1X_2, X_3, Y_2Y_3, N, T$ be generated from the indistinguishability instance of Assumption 2. - Recall then that either $T \leftarrow \mathbb{G}$ or $T \leftarrow \mathbb{G}_{p_1p_3}$. We generate the public key by generating $a, \alpha \leftarrow \mathbb{Z}_N$ and $s_i \leftarrow \mathbb{Z}_N$ for all $i \in \Omega$ and setting:

$$PK = (N, g, g^a, e(g,g)^\alpha, \{T_i = g^{s_i} \forall i \in \Omega\})$$

We first describe the generation of the semi-functional challenge ciphertext with policy (A, ρ):

- **Challenge Ciphertext Generation.** Generate the vectors u uniformly over $\mathbb{Z}_N^l$ subject to the restriction that $u \cdot 1 = a$ and generate $r_i \leftarrow \mathbb{Z}_N$ for each $i \in [n]$ set:

$$C = Me(g^\alpha, X_1X_2) \quad , \quad C' = X_1X_2$$

$$C_i = (X_1X_2)^{A_i u}(X_1X_2)^{-r_i s_{\rho(i)}} \quad , \quad D_i = (X_1X_2)^{r_i}$$

We next describe how to generate semi-functional keys for the first $k-1$ identifiers.

- **The first $k-1$ identifiers queried to KeyGen**(MSK, b, S, U'): If $\alpha_{U'}$ or $d_{U',b}$ has not been generated yet, generate it randomly from $\mathbb{Z}_N$. Generate $t_{U'} \leftarrow \mathbb{Z}_N$ and $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$ and set where $\gamma = \alpha_U$ if $b = 0$ and $\alpha - \alpha_U$ if $b = 1$:

$$K_{U'} = g^{t_{U'}} V \quad , \quad K_{U',S} = \{T_i^{t_{U'}} W_i : i \in S\} \quad K^*_{U'} = g^{\gamma} g^{a t_{U'}} Z (Y_2 Y_3)^{d_{U',b}}$$

- **The k^{th} identifier where $b = 0$**: Let U be the k^{th} identifier. For the first $i-1$ queries to $\mathsf{KeyGen}(MSK, 0, S, U)$ If α_U has not been generated yet, generate it randomly from $\mathbb{Z}_N$. Generate $t_U, d_U \leftarrow \mathbb{Z}_N$ and $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$ and set:

$$K_U = g^{t_U} V \quad , \quad K_{U,S} = \{T_i^{t_U} W_i : i \in S\} \quad K_V = g^{\alpha_U} g^{a t_U} Z (Y_2 Y_3)^{d_U}$$

  For the first i^{th} queries to $\mathsf{KeyGen}(MSK, 0, S, U)$: If α_U has not been generated yet, generate it randomly from $\mathbb{Z}_N$. Generate $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$ and set:

$$K_U = TV \quad , \quad K_{U,S} = \{T^{s_i} W_i : i \in S\} \quad K_V = g^{\alpha_U} T^a Z$$

  For the queries after the i^{th} or where $b = 1$ to $\mathsf{KeyGen}(MSK, b, S, U)$: If α_U has not been generated yet, generate it randomly from $\mathbb{Z}_N$. Generate $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$ and set where $\gamma = \alpha_U$ if $b = 0$ and $\alpha - \alpha_U$ if $b = 1$:

$$K_U = g^{t_U} V \quad , \quad K_{U,S} = \{T_i^{t_U} W_i : i \in S\} \quad K_V = g^{\gamma} g^{a t_U} Z$$

Generating the fully functional keys for the identifiers numbered $k' > k$ from the correct distribution is simple since we have the master secret key (α, X_3) as they are generated identically to the first $k-1$ identifiers where $d_{U,b}$ is set to be 0.

If $T \in \mathbb{G}_{p_1 p_3}$ the above is a uniform instance of $\mathcal{I}_{i-1}$. On the other hand, if $T \in \mathbb{G}$ then the above procedure is almost equivalent to the procedure from $\mathcal{H}_i$ except for the fact that the C_i components of the challenge ciphertext are multiplied by a new $X_2^{A_i \cdot u}$ factor where instead of u being generated uniformly at random, it is uniform subject to the restriction that $u \cdot 1 = a$. However, for all i such that $\rho(i) \notin T$ (where T is the attribute set for the i^{th} query to the k^{th} identifier with $b = 0$) notice the $\mathbb{G}_{p_2}$ components of C_i are independent from all other parts of the scheme (as for these values $s_{\rho(i)}$ is nowhere else used as an exponent in $\mathbb{G}_{p_2}$, note this statement requires the injectivity of ρ).

Therefore, the additional restriction that $u \cdot 1 = a$ is only present in the ciphertext components for i with $\rho(i) \notin S$ where S is the attribute set for the i^{th} query to the k^{th} identifier when $b = 0$. Since $1 \notin \mathsf{Span}(A_i : \rho(i) \in T)$ this generation procedure is identical to when u is generated completely uniformly (this argument is identical to that presented in the KP-ABE proof for indistinguishability between $\mathcal{I}_{i-1}$ and $\mathcal{H}_i$) which is the correct distribution. □

Lemma 10.9. *If A is Type 1. then $|\mu_i - \nu_i|$ is negligible in λ for any $i \in [q]$ if Assumption 2. holds.*

Proof. The only divergence from the above proof is for the i^{th} query with $b = 0$ to the k^{th} identifier which we give below:

- **The i^{th} query to the k^{th} identifier with $b = 0$**: **KeyGen**$(MSK, 0, S, U)$. If α_U has not been generated, sample it uniformly from $\mathbb{Z}_N$. Sample d_U uniformly at random from $\mathbb{Z}_N$ and $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$ and set:

$$K_U := TV \quad , \quad K_{U,S} := \{T^{s_i} W_i : i \in S\} \quad K^*_U := T^a g^{\alpha_U} Z (Y_2 Y_3)^{d_U}$$

Notice that the key generation above is actually independent of the $\mathbb{G}_{p_2}$ part of T^a (if $T \in \mathbb{G}$) to the additional re-randomization by $Y_2^{d_U}$ and therefore the additional analysis in the previous case is not needed since sampling u subject to the restriction $u \cdot 1 = a$ is equivalent to generating it randomly if a is not used as an exponent in $\mathbb{G}_{p_2}$ anywhere else in the scheme. It's therefore a simple observation that if $T \in \mathbb{G}_{p_1 p_3}$ the above is a uniform instance of $\mathcal{I}_i$ and if $T \in \mathbb{G}$ the above is a uniform instance in $\mathcal{H}_i$, as desired. □

Now, by a hybrid argument and the observation that $\mathcal{I}_0 = \text{GAME}_{k-1}$:

Corollary 10.10. *If A is of Type 1. then $|\mu_q - \epsilon_{k-1}|$ is negligible in λ if Assumption 2. holds.*

We still have not proven our desired result however as $\mathcal{I}_q$ differs from GAME_k in two ways. First, each KeyGen with $b = 0$ query to the k^{th} identifier has the $\mathbb{G}_{p_2}$ component of K_U^* re-randomized separately (whereas in a semi-functional identifier, the $\mathbb{G}_{p_2}$ component is constant for all such queries). Second, KeyGen queries with $b = 1$ to U should have the K_U^* component offset by a constant $g_2^{e_U}$ amount. For this we need another sequence of hybrids.

Definition 10.11. *Define $\mathcal{J}$ as a modification of $\mathcal{I}_q$ as follows. For the k^{th} identifier U, rather than generating $f_U \leftarrow \mathbb{Z}_N$ for each query $\mathsf{KeyGen}(MSK, 0, S, U)$, the value f_U is generated uniformly once for U and re-used across all such queries.*

Let η be the advantage of the adversary in $\mathcal{J}$.

Lemma 10.12. *If A is of Type 1. then $|\mu_q - \nu|$ is negligible in λ if Assumption 2. holds.*

To show this we use another series of games:

Definition 10.13. *We define $\mathcal{I}_i'$ and $\mathcal{H}_i'$ as modifications of $\mathcal{I}_i$ and $\mathcal{H}_i$ respectively where for the k^{th} identifier U an additional value $d_{U,0}$ is generated uniformly at random and all queries to U of the form $\mathsf{KeyGen}(MSK, 0, S, U)$ after being generated from $\mathcal{I}_i$ or $\mathcal{H}_i$ respectively are modified by:*

$$K_U^* := K_U^* \times g_2^{d_{U,0}}$$

Note that in the above definition $d_{U,0}$ is only generated once for U. We call the advantage of the adversary in $\mathcal{I}_i'$ and $\mathcal{H}_i'$, μ_i' and ν_i' respectively.

Notice first that $\mathcal{I}_q' = \mathcal{I}_q$ as the independent f_U factor in each of these queries in $\mathcal{I}_q$ completely subsumes the additional $g_2^{d_{U,0}}$ factor. Additionally, through repeating the proofs of the previous two lemmas with the only modification being that for the k^{th} identifier U, $d_{U,0}$ is generated and all responses to $\mathsf{KeyGen}(MSK, 0, S, U)$ are modified as $K_U^* := K_U^* \times g_2^{d_{U,0}}$, we can conclude the following two lemmas:

Lemma 10.14. *If A is of Type 1. then $|\mu_{i-1}' - \nu_i'|$ is negligible in λ for any $i \in [q]$ if Assumption 2. holds.*

and,

Lemma 10.15. *If A is of Type 1. then $|\mu_i' - \nu_i'|$ is negligible in λ for any $i \in [q]$ if Assumption 2. holds.*

Since $\mathcal{I}_0' = \mathcal{J}$, this allows us to combine these two lemmas and indistinguishability between $\mathcal{I}_q'$ and Game_{k-1} to conclude:

Corollary 10.16. *If A is of Type 1. then $|\eta - \epsilon_{k-1}|$ is negligible in λ if Assumption 2. holds.*

Now we have indistinguishability of GAME_{k-1} form $\mathcal{J}$ which is much closer to GAME_k. Like GAME_k, $\mathcal{J}$ has all $\mathsf{KeyGen}(MSK, 0, S, U)$ queries for the k^{th} identifier U have the K_U^* value offset by a constant amount in $\mathbb{G}_{p_2}$. However, the $\mathsf{KeyGen}(MSK, 1, S, U)$ values are not yet offset. For this, we will need one final lemma.

Lemma 10.17. *If A is of Type 1. then $|\epsilon_k - \eta|$ is negligible in λ if Assumption 2. holds.*

Proof. Let $g, X_1X_2, X_3, Y_2Y_3, N, T$ be generated from the indistinguishability instance of Assumption 2. - Recall then that either $T \leftarrow \mathbb{G}$ or $T \leftarrow \mathbb{G}_{p_1p_3}$. We generate the public key by generating $a, \alpha \leftarrow \mathbb{Z}_N$ and setting:

$$PK = (N, g, g^a, e(g, g)^\alpha, \{T_i = g^{z_i} \forall i \in \Omega\})$$

First we demonstrate how to simulate semi-functional challenge ciphertext generation (A, ρ):

- **Challenge Ciphertext Generation.** On a query $(M_0, M_1, (A, \rho))$, generate the vector $u \in \mathbb{Z}_N^l$ uniformly subject to the restriction that $u \cdot 1 = a$. Then choose $b \leftarrow \{0,1\}$ uniformly and $r_i \leftarrow \mathbb{Z}_N$ for all $i \in [n]$ and set:

$$C = M_b e(g^\alpha, X_1 X_2) \quad , \quad C' = X_1 X_2$$

$$C_i = (X_1 X_2)^{A_i \cdot u} (X_1 X_2)^{-r_i s_{\rho(i)}} \quad , \quad D_i = (X_1 X_2)^{r_i}$$

Next we show how to generate semi-functional keys for the first $k-1$ identifiers.

- **The first $k-1$ identifiers:** On a query $\mathbf{KeyGen}(MSK, b, S, U)$: If α_U or $d_{U,b}$ have not yet been generate them uniformly over $\mathbb{Z}_N$. Generate $t_U \leftarrow \mathbb{Z}_N$, $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$. Let $\gamma = \alpha_U$ if $b = 0$ and $\alpha - \alpha_U$ if $b = 1$ and set:

$$K_U = g^{t_U} V \quad , \quad K_{U,S} = \{T_i^{t_U} W_i : i \in S\} \quad K_U^* = g^{\alpha_U} g^{a t_U} Z (Y_2 Y_3)^{d_{U,b}}$$

The key for the identifiers following k^{th} are also generated according to the above distribution where $d_{U,b}$ is set to 0. We now detail the generation for the k^{th} identifier.

- **The k^{th} identifier:** On a query $\mathbf{KeyGen}(MSK, 0, S, U)$: If $d_{U,0}$ has not yet been generate it uniformly over $\mathbb{Z}_N$. Generate $t_U \leftarrow \mathbb{Z}_N$, $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$. Set:

$$K_U = g^{t_U} V \quad , \quad K_{U,S} = \{g^{t_U s_i} W_i : i \in S\} \quad K_U^* = T g^{a t_U} Z (Y_2 Y_3)^{d_{U,0}}$$

On a query $\mathbf{KeyGen}(MSK, 1, S, U)$: Generate $t_U \leftarrow \mathbb{Z}_N$, $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$. Set:

$$K_U = g^{t_U} V \quad , \quad K_{U,S} = \{g^{t_U s_i} W_i : i \in S\} \quad K_U^* = g^\alpha T^{-1} g^{a t_U} Z$$

Here if we write $T_{\mathbb{G}_{p_1}}$ (the $\mathbb{G}_{p_1}$ part of T) as g^t we are implicitly setting $\alpha_U = t$. If $T \in \mathbb{G}_{p_1 p_3}$ then the keys for the k^{th} identifier have $d_{U,0}$ as in $\mathcal{I}_0'$. Similarly, if $T \in \mathbb{G}_{p_1 p_2 p_3}$, $g_2^{d_{U,0}}$ is set to be the $g_2^{d_{U,0}}$ value generated in the above simulation plus the $\mathbb{G}_{p_2}$ part of T and $g_2^{d_{U,1}}$ is set to be the $\mathbb{G}_{p_2}$ part of T^{-1}. As these are both uniform, independent and reused across all queries to this identifier with the same value for b, this draws from the correct distribution for GAME_k as desired. □

Now combining all the lemmas we have shown we can conclude:

Lemma 10.18. *If A is of Type 1. then $|\epsilon_k - \epsilon_{k-1}|$ is negligible in λ for any $k \in [q]$ if Assumption 2. holds.*

Through a symmetric argument, the above also holds for Type 2. adversaries (where the $\mathcal{H}_i, \mathcal{I}_i, \mathcal{H}_i', \mathcal{I}_i'$ hybrids now modify the generation procedure when $b = 1$). If there was a general adversary A (not Type 1. or Type 2. that could distinguish between GAME_k and GAME_{k+1} notice that it would be possible to make two adversaries A_1 and A_2 such that A_1 is Type 1. and Type 2. using A (such that A_i aborts and guesses randomly is A ever makes a query to violate being Type i. that either A_1 or A_2 would have a non-negligible advantage if A did, violating the above lemma.

Therefore, GAME_k and GAME_{k+1} are indistinguishable for general adversaries and through a hybrid argument we have:

Corollary 10.19. *$|\epsilon_0 - \epsilon_q|$ is negligible in λ if Assumption 2. holds.*

Let $\text{GAME}_{\text{Final}}$ denote the game identical to GAME_q with the only exception that rather than choosing the challenge message to be M_c where $c \leftarrow \{0,1\}$, the message is chosen uniformly over $\mathbb{G}_T$.

Lemma 10.20. *If Assumption 3. holds, $|\epsilon_{\text{FINAL}} - \epsilon_q|$ is negligible.*

Proof. Recall that in Assumption 3. we are given $N, g, g^{\alpha}X_2, X_3, g^{s}Y_2, Z_2$ and T which either is $e(g,g)^{\alpha s}$ or a uniform element over $\mathbb{G}_T$. We now describe how to simulate either an instance of GAME_q or $\text{GAME}_{\text{Final}}$ depending on which distribution T comes from.

- To generate the public key, choose $a \leftarrow \mathbb{Z}_N$ and $z_i \leftarrow \mathbb{Z}_N$ for all $i \in \Omega$ and set:

$$PK = (N, g, g^a, e(g, g^{\alpha}X_2) = e(g,g)^{\alpha}, (T_i = g^{z_i} : i \in \Omega))$$

We next describe how to generate keys for semi-functional identifiers:

- **Semi-Functional Identifiers.** To queries to $\mathsf{KeyGen}(MSK, b, S, U)$ If $d_{U,b}$ or α_U has not been generated yet, sample it uniformly from $\mathbb{Z}_N$ and store it. Generate $t_U \leftarrow \mathbb{Z}_N$, $V, Z \leftarrow \mathbb{G}_{p_3}$ and $W_i \leftarrow \mathbb{G}_{p_3}$ for all $i \in S$. Set $G = g^{\alpha_U}$ if $b = 0$ and $G = g^{\alpha}X_2 g^{-\alpha_U}$ if $b = 1$ and set:

$$K_U = g^{t_U}V \quad , \quad K_{U,S} = \{T_i^{t_U}W_i \forall i \in S\} \quad , \quad K_U^* = Gg^{at_U}Z_2^{d_{U,b}}Z$$

- **The Ciphertext.** Generate $v \in \mathbb{Z}_N^l$ uniformly subject to the restriction that $v \cdot 1 = a$ and $r_i \leftarrow \mathbb{Z}_N$ for all $i \in [n]$ and $s \leftarrow \mathbb{Z}_N$ and set:

$$C = M_c T \quad , \quad C' = g^s Y_2$$

$$C_i = (g^s Y_2)^{A_i \cdot v}(g^s Y_2)^{-r_i s_{\rho(i)}} \quad , \quad D_i = (g^s Y_2)^{r_i}$$

When $T = e(g,g)^{\alpha}$ the above is a correctly distributed semi-functional ciphertext, whereas when T is a random element in $\mathbb{G}_T$ it is an encryption of a random message. Since ϵ_{FINAL} must be negligible, as the challenge bit c is completely independent of anything in the challenge ciphertext, this proves security of our scheme. □

Acknowledgements

We gratefully thank Thomas King and Daniel Manchala of Xerox for stimulating discussions regarding ABE with dynamic credentials. In particular, Thomas King and Daniel Manchala suggested to us that the problem of revoking access to stored data that had not yet been accessed could be of significant interest in practice, and their suggestion inspired us to consider this problem. We also thank the anonymous reviewers for their helpful comments on our writeup.

References

[1] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 116–129. Springer, 2000.

[2] William Aiello, Sachin Lodha, and Rafail Ostrovsky. Fast digital identity revocation (extended abstract). In *CRYPTO*, pages 137–152, 1998.

[3] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer, 1999.

[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[5] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *ACM Conference on Computer and Communications Security*, pages 417–426, 2008.

[6] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. *Advances in CryptologyEurocrypt 2003*, pages 646–646, 2003.

[7] Craig Gentry. Certificate-based encryption and the certificate revocation problem. In *EUROCRYPT*, pages 272–293, 2003.

[8] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*, page 98. ACM, 2006.

[9] Vipul Goyal. Certificate revocation using fine grained certificate space partitioning. In *Financial Cryptography*, pages 247–259, 2007.

[10] Vipul Goyal, Steve Lu, Amit Sahai, and Brent Waters. Black-box accountable authority identity-based encryption. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM Conference on Computer and Communications Security*, pages 427–436. ACM, 2008.

[11] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.

[12] A. Lewko and B. Waters. New Proof Methods for Attribute-Based Encryption: Achieving Full Security through Selective Techniques. *CRYPTO*, 2012.

[13] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010.

[14] Benoît Libert and Damien Vergnaud. Adaptive-id secure revocable identity-based encryption. In *CT-RSA*, pages 1–15, 2009.

[15] Tal Malkin, Daniele Micciancio, and Sara K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2002.

[16] S. Micali. Efficient certificate revocation. *LCS/TM 542b, Massachusetts Institute of Technology*, 1996.

[17] S. Micali. NOVOMODO: Scalable certificate validation and simplified PKI management. In *Proc. of 1st Annual PKI Research Workshop*, 2002.

[18] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. *Electronic Colloquium on Computational Complexity (ECCC)*, (043), 2002.

[19] M. Naor and K. Nissim. Certificate revocation and certificate update. *IEEE Journal on Selected Areas in Communications*, 18(4):561–560, 2000.

[20] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010.

[21] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM conference on Computer and communications security*, page 203. ACM, 2007.

[22] J. Qian and X. Dong. Fully secure revocable attribute-based encryption. *Journal of Shanghai Jiaotong University (Science)*, 16(4):490–496, 2011.

[23] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, volume 3494, pages 457–473. Springer, 2005.

[24] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *Public Key Cryptography*, pages 53–70, 2011.