Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption

Amit Sahai[1,⋆], Hakan Seyalioglu[2,⋆⋆], and Brent Waters[3,⋆⋆⋆]

[1] Department of Computer Science, UCLA
[2] Department of Mathematics, UCLA
[3] Department of Computer Science, University of Texas at Austin

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. Our main result is obtained by pairing two contributions:

- We first ask how a third party who is not trusted with secret key information can process a ciphertext to disqualify revoked users from decrypting data encrypted in the past. Our core tool is a new procedure called *ciphertext delegation* that allows a ciphertext to be 're-encrypted' to a more restrictive policy using only public information.
- Second, we study the problem of revocable attribute-based encryption. We provide the first fully secure construction by modifying an attribute-based encryption scheme due to Lewko et al. [9] and prove security in the standard model.

We then combine these two results for a new approach for revocation on stored data. Our scheme allows a storage server to update stored ciphertexts to disqualify revoked users from accessing data that was encrypted before the user's access was revoked while key update broadcasts can dynamically revoke selected users.

⋆ 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.

R. Safavi-Naini and R. Canetti (Eds.): CRYPTO 2012, LNCS 7417, pp. 199–217, 2012.

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 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) [18]. However, this fails to address the problem that the credentials of a user may change with time. This problem motivated the study of revocation [3] where a 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 study 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

[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.

upkeep as the database is handling constant two way communication in our system and is therefore modeled as the most vulnerable party.

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, all stored files (no matter how old) immediately become inaccessible to this user 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).

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 [8,18,19,9].

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 credentials was considered by Boldyreva et al. [3] 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].

[2] A related work [17] proposes a 'fully secure Revocable ABE scheme' under a significantly different model than the one presented here.

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*. This requirement is similar to the standard ABE requirement but allows for an adversary to build his key 'piece by piece'. Many existing proof methods extend with only minor modifications to prove piecewise security of existing schemes. We show that variants of the transformation method of Boldyreva et al. [3] 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 [9] 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.

Related Work. Originally proposed by Sahai and Waters [18], *attribute-based encryption* [8,16,19,9,15] has been an active research area in cryptography in part since it is a primitive with interesting functional applications [7,3] and can be implemented efficiently [2]. 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 $P(U) = \mathsf{True}$. While the problem of delegating a key to a more restrictive key has been considered [8], 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 [11,14,1,12,13,5,6], identity based encryption [3,10] and attribute-based encryption [17]. At a high level, our revocable storage results can be seen as taking methods from forward secure encryption [4] 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.

Roadmap. We briefly give an organizational outline to the paper. We begin with an introduction to preliminary notions and notation including formally defining attribute-based encryption and revocable ABE in Section 2. In Section 3 we define the piecewise key generation requirement that implies a revocable ABE scheme in a black-box fashion. In Section 4 we define revocable storage. Ciphertext delegation is defined and studied in Section 5. We give a technical lemma to efficiently handle time in our final construction in Section 6. We combine all our previously introduced tools to give our main construction of a revocable storage ABE scheme from an ABE scheme with piecewise key generation and ciphertext delegation in Section 7. We finally give a construction of an ABE scheme with piecewise key generation and ciphertext delegation in Section 8.

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 use $[i, j]$ to denote the set of all integers from i to j inclusive or $[i]$ as shorthand for $[1, i]$. 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. We now briefly give the syntax for the two central notions for this chapter.

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.

Definition 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**$(1^\lambda) \rightarrow (PK, MSK)$: Setup takes as input the security parameter and outputs the public key and master secret key.
- **KeyGen**$(MSK, P) \rightarrow SK_P$: Key generation outputs a secret key with policy $P \in \mathcal{P}$.
- **Encrypt**$(PK, M, S) \rightarrow C_S$: Encrypts a message $M \in \mathbb{M}$ under the attribute set $S \subseteq \Omega$.
- **Decrypt**$(SK_P, C_S) \rightarrow M$: Decryption successfully recovers the encrypted message if and only if $P(S) = 1$ (i.e. 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). In a **Ciphertext-Policy ABE** (CP-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. In these conference proceedings, we only detail the KP-ABE version of our results and delay the definition of revocable storage CP-ABE and our construction to the full version.

Revocable Attribute-Based Encryption. A revocable attribute-based encryption scheme [3] 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. As in previous work, we will assume the key generation procedure is *stateful* for these constructions (it can store information used to create other keys in internal state denoted by σ).

Definition 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**$(1^\lambda) \to (PK, MSK, \sigma)$: Setup takes as input the security parameter and outputs the public key, master secret key and initializes $\sigma = \emptyset$.
- **KeyGen**$(MSK, P, ID, \sigma) \to (SK_{P,ID}, \sigma)$: Key generation outputs a secret key with policy $P \in \mathcal{P}$ for the user $ID \in \{0,1\}^{\mathcal{I}}$ and updates the state σ.
- **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**$(MSK, rl, t, \sigma) \to (K_t, \sigma)$: The update phase takes as input a revocation list rl (a set of strings in $\{0,1\}^{\mathcal{I}}$) and the master secret key, outputs the key update information for time t and updates the state σ.
- **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).

Security Game Oracles. Define the following oracles to use in the security game. These oracles are given access to (PK, MSK, σ) 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 (P, ID) and returns $SK_{P,ID}$ generated from: $(SK_{P,ID}, \sigma) \leftarrow \mathsf{KeyGen}(MSK, P, ID, \sigma)$.
2. The Key Update Generation oracle $K(\cdot, \cdot)$ takes as input t and a revocation list rl and returns K_t generated from: $(K_t, \sigma) \leftarrow \mathsf{KeyUpdate}(MSK, t, rl, \sigma)$.

Note that for both oracles σ is not sent to the adversary but is used to update the current σ value of the scheme. 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.

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

1. The challenger runs $\mathsf{Setup}(1^\lambda) \to (PK, MSK, \sigma)$ and sends 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;
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 conditions placed on the adversary's queries are as follows: For any query, $SK(P, ID)$ such that $P(S^*) = 1$, $ID \in rl$ for every query $K(t, rl)$ with $t \geq t^*$.

Informally this corresponds to the fact that every user with sufficient credentials to decrypt the challenge ciphertext should be revoked by time t^* for the message to remain hidden.

Definition 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:*

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

is negligible in λ.

3 Revocable Attribute-Based Encryption

The Revocable ABE and IBE constructions given by Boldyreva et al. [3] 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. [9] 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. We find that 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. For the rest of this section we will assume key generation is allowed to be stateful (as captured with σ in the Revocable ABE definition) but will omit the state being updated as part of the syntax of the key generation algorithm from this point on for notational simplicity.

Definition 4. (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 $U \in \{0,1\}^{\mathcal{I}}$:

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

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

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

We now define correctness of the scheme:

Definition 5. (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 $\mathsf{KeyGen}(MSK, 0, P_0, U) \rightarrow K^{(0)}_{P_0,U}$, $\mathsf{KeyGen}(MSK, 1, P_1, U) \rightarrow K^{(1)}_{P_1,U}$, then,

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

The definition for security for a scheme with *piecewise key generation* is now 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 $\mathsf{Setup}(1^\lambda) \rightarrow (PK, MSK)$ 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. Challenger picks b a random bit, sends $\mathsf{Encrypt}(PK, M_b, S^*)$ 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 6. *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:*

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

is negligible in λ.

3.1 Revocability from Piecewise KP-ABE

To apply our results to revocability 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 [9], we will state our result if the KP-ABE scheme supports access policy formatted as LSSS matrices (as in [9]).

LSSS Matrices. 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 1. *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 *with an attribute set of size* $\omega - 2\log(T)$.

In Section 7 we prove a stronger statement: In language that will be introduced shortly, we will prove that a KP-ABE scheme with piecewise key generation and cipertext delegation implies a Revocable-Storage KP-ABE scheme. The construction presented in Section 5 when the underlying scheme does not have ciphertext delegation fulfills the requirement of being a Revocable KP-ABE scheme and therefore we defer the construction to that section. In Section 8 we give a modification of the fully secure ABE scheme due to Lewko et al. [9] that has piecewise key generation.

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 ciphertext. 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 [4] which is used to manage keys, to managing ciphertexts.

Definition 7. (Revocable Storage) *A Revocable KP-ABE scheme has Revocable Storage if it has the following additional algorithm:*

- **CTUpdate**$(C_{S,t}, PK) \rightarrow C_{S,t+1}$: Ciphertext updating 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, 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 strong requirement is motivated further in the following section.

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 trivial inefficient ways to satisfy this requirement assuming a standard 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. Note delegation is required to produce a new encryption of the same message that is independent of the randomness and access policy of 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.

5.1 KP-ABE Ciphertext Delegation

For monotone access policies (as are generally considered in the literature [8,9,18]), 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 8. (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 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 simple analysis also holds for [9,18] which allows us to conclude:

Theorem 2. *The KP-ABE schemes in [8,9,18] have ciphertext delegation.*

In the full version of this paper we also give full delegation procedures for a variety of existing CP-ABE schemes whose access structures are either built on threshold trees [18,2] or LSSS matrices [19,9]. For this analysis we define

certain operations that we call *elementary ciphertext manipulations* that can be performed on ciphertexts which all the schemes we mentioned above allow and show how any CP-ABE scheme that allows elementary ciphertext manipulations actually allows robust delegation operations on the underlying ciphertext. We cite one of our results which follows as a special case of our more general analysis. A formal definition of threshold trees is present either in the full version of this paper or in either of the cited works[3].

Theorem 3. *The CP-ABE schemes [18,2]) allow the following delegation: 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 threshold $n_x = 1$ (In other words, splitting an edge into two, connected by a node x).
2. Increasing a threshold $n_x \rightarrow n_x + 1$ while optionally adding another descendent leaf y to x assigned to an arbitrary attribute.
3. Deleting a subtree.

6 Managing the Time Structure Efficiently

In this section we will give the main technical lemma needed to achieve efficient delegation. 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 similar to that used by Canetti et al. [4] in the context of forward secure encryption. 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.

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

[3] Informally, a threshold tree is an access structure represented by a tree where leaves are labeled with attributes and each internal node is labeled with an integer threshold. To see if the tree evaluates to true on a set of attributes, first the leaves corresponding to these attributes are labeled true and all others are labeled false. Then an internal node on the second level is labeled true only if a number of its descendents equal to or exceeding its threshold are labeled true. After this process is recursed through all of the tree, if the root evaluates to true, the threshold tree is satisfied.

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. This theorem is also implicit in the work of [4] but we provide a proof in the full version for containment. 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.

Theorem 4. *There are (efficiently computable) subsets of $V(\mathcal{Q})$ (the node set of $\mathcal{Q}$ where $\mathcal{Q}$ is a tree of depth r), $\{\mathcal{T}_i : i \in [q]\}$ 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 an informal 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 some node of 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 update ciphertexts. Property 3. guarantees that this process can be done efficiently.

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 5. *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 [3]) with the identities $\{0,1\}^{\mathcal{I}}$ labeling the leaves. For a set of leaves V, the function $\mathcal{U}(V)$ returns 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 [3]. 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:

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

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 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 4.

- **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 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 of RKP-ABE Security. 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 6. *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. [9]:

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

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

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 $D=(\mathbb{G},\mathbb{G}_T,N,g,g^\alpha X_2,X_3,g^sY_2,Z_2)$, then:

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

is negligible in λ.

We now provide the construction that fulfils the requirements put forth in the body of the paper. We prove this theorem in the full version of this paper.

Theorem 7. *The KP-ABE scheme given below has piecewise key generation, supports 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. We let the vector 1 stand as shorthand for $1 \circ 0 \circ 0 \ldots 0$ when the dimension is specified by context. Below we will give the decryption algorithm the public key as input, which was not part of our original definition, but can be easily adapted to our definition by including the public key as part of the secret key.

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 and store it. For each row A_i of A choose a uniform $r_i \leftarrow \mathbb{Z}_N$ 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:

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

Decrypt$(CT_S, SK_{U,(A,\rho)}^{(0)}, SK_{U,(B,\beta)}^{(1)}, PK)$. Take ω_i with $\sum_i \omega_i A_i = 1$.

Label the components of $SK_{U,(A,\rho)}^{(0)}$ 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 $SK_{U,(B,\beta)}^{(1)}$ recovers $e(g,g)^{s(\alpha-\alpha_U)}$ allowing recovery of $e(g,g)^{s\alpha}$ and M as $C/e(g,g)^{s\alpha}$.

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. Aiello, W., Lodha, S.P., Ostrovsky, R.: Fast Digital Identity Revocation (Extended Abstract). In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 137–152. Springer, Heidelberg (1998)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334 (2007)
3. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: ACM CCS, pp. 417–426 (2008)
4. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
5. Gentry, C.: Certificate-Based Encryption and the Certificate Revocation Problem. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 272–293. Springer, Heidelberg (2003)
6. Goyal, V.: Certificate Revocation Using Fine Grained Certificate Space Partitioning. In: Dietrich, S., Dhamija, R. (eds.) FC 2007 and USEC 2007. LNCS, vol. 4886, pp. 247–259. Springer, Heidelberg (2007)
7. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: CCS, pp. 427–436 (2008)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: CCS, pp. 89–98 (2006)
9. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
10. Libert, B., Vergnaud, D.: Adaptive-ID Secure Revocable Identity-Based Encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 1–15. Springer, Heidelberg (2009)
11. Micali, S.: Efficient certificate revocation. LCS/TM 542b, Massachusetts Institute of Technology (1996)
12. Micali, S.: NOVOMODO: Scalable certificate validation and simplified PKI management. In: Proc. of 1st Annual PKI Research Workshop (2002)

13. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. ECCC (043) (2002)
14. Naor, M., Nissim, K.: Certificate revocation and certificate update. IEEE Journal on Selected Areas in Communications 18(4), 561–570 (2000)
15. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
16. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: CCS, p. 203 (2007)
17. Qian, J., Dong, X.: Fully secure revocable attribute-based encryption. Journal of Shanghai Jiaotong University (Science) 16(4), 490–496 (2011)
18. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
19. Waters, B.: Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)