Outsourcing the Decryption of ABE Ciphertexts

Matthew Green
Johns Hopkins University

Susan Hohenberger*
Johns Hopkins University

Brent Waters†
University of Texas at Austin

Abstract

Attribute-based encryption (ABE) is a new vision for public key encryption that allows users to encrypt and decrypt messages based on user attributes. For example, a user can create a ciphertext that can be decrypted only by other users with attributes satisfying ("Faculty" OR ("PhD Student" AND "Quals Completed")). Given its expressiveness, ABE is currently being considered for many cloud storage and computing applications. However, one of the main efficiency drawbacks of ABE is that the size of the ciphertext and the time required to decrypt it grows with the complexity of the access formula.

In this work, we propose a new paradigm for ABE that largely eliminates this overhead for users. Suppose that ABE ciphertexts are stored in the cloud. We show how a user can provide the cloud with a *single* transformation key that allows the cloud to translate *any* ABE ciphertext satisfied by that user's attributes into a (constant-size) El Gamal-style ciphertext, without the cloud being able to read any part of the user's messages.

To precisely define and demonstrate the advantages of this approach, we provide new security definitions for both CPA and replayable CCA security with outsourcing, several new constructions, an implementation of our algorithms and detailed performance measurements. In a typical configuration, the user saves significantly on both bandwidth and decryption time, without increasing the number of transmissions.

*Supported by NSF CAREER CNS-1053886, DARPA PROCEED, Air Force Research Laboratory, Office of Naval Research N00014-11-1-0470, a Microsoft Faculty Fellowship and a Google Faculty Research Award. 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.

†Supported by NSF CNS-0915361 and CNS-0952692, AFOSR Grant No: FA9550-08-1-0352, DARPA PROCEED, DARPA N11AP20006, Google Faculty Research Award, the Alfred P. Sloan Fellowship, and Microsoft Faculty Fellowship. 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.

1 Introduction

Traditionally, we have viewed encryption as a method for one user to encrypt data to another specific targeted party, such that only the target recipient can decrypt and read the message. However, in many applications a user might often wish to encrypt data according to some *policy* as opposed to specified set of users. Trying to realize such applications on top of a traditional public key mechanism poses a number of difficulties. For instance, a user encrypting data will need to have a mechanism which allows him to look up all parties that have access credentials or attributes that match his policy. These difficulties are compounded if a party's credentials themselves might be sensitive (e.g., the set of users with a TOP SECRET clearance) or if a party gains credentials well after data is encrypted and stored.

To address these issues, a new vision of encryption was put forth by Sahai and Waters [38] called Attribute-Based Encryption (ABE). In an ABE system, a user will associate an encryption of a message M with an function $f(\cdot)$, representing an access policy associated with the decryption. A user with a secret key that represents their set of attributes (e.g., credentials) S and will be able to decrypt a ciphertext associated with function $f(\cdot)$ if and only if $f(S) = 1$. Since the introduction of ABE there have been several other works proposing different variants [24, 7, 14, 36, 23, 42, 15, 28, 35] extending both functionality and refining security proof techniques. [1]

One property that all of these ABE systems have is that both the ciphertext size and time for decryption grow with the size of the access formula f. Roughly, current efficient ABE realizations are set in pairing-based groups where the ciphertexts require two group elements for every node in the formula and decryption will require

[1] A more general concept of functional encryption [11] allows for more general functions to be computed on the encrypted data and encompasses work such as searching on encrypted data and predicate encryption [10, 2, 12, 39, 27].

Scheme	ABE Type	Security Level	Model	Full CT Size	Full Decrypt Ops	Out CT Size	Out Dec Ops
Waters [42]	CP	CPA	-	$\|\mathbb{G}_T\|+(1+2\ell)\|\mathbb{G}\|$	$\leq(2+\ell)P+2\ell E_G$	-	-
§3.1	CP	CPA	-	$\|\mathbb{G}_T\|+(1+2\ell)\|\mathbb{G}\|$	$\leq(2+\ell)P+2\ell E_G$	$2\|\mathbb{G}_T\|$	E_T
§3.2	CP	RCCA	RO	$\|\mathbb{G}_T\|+(1+2\ell)\|\mathbb{G}\|+k$	$\leq(2+\ell)P+2\ell E_G+2E_T$	$2\|\mathbb{G}_T\|+k$	$3E_T$
GPSW [24]	KP	CPA	-	$\|\mathbb{G}_T\|+(1+s)\|\mathbb{G}\|$	$\leq(1+\ell)P+2\ell E_G$	-	-
§4.1	KP	CPA	-	$\|\mathbb{G}_T\|+(1+s)\|\mathbb{G}\|$	$\leq(1+\ell)P+2\ell E_G$	$2\|\mathbb{G}_T\|$	E_T
§4.2	KP	RCCA	RO	$\|\mathbb{G}_T\|+(1+s)\|\mathbb{G}\|+k$	$\leq(1+\ell)P+2\ell E_G+2E_T$	$2\|\mathbb{G}_T\|+k$	$3E_T$

Figure 1: Summary of ABE outsourcing results. Above s denotes the size of an attribute set, ℓ refers to an LSSS access structure with an $\ell \times n$ matrix, k is the message bit length in RCCA schemes, and P, E_G, E_T stand for the maximum time to compute a pairing, exponentiation in $\mathbb{G}$ and exponentiation in $\mathbb{G}_T$ respectively. We ignore non-dominant operations. All schemes are in the selective security setting. We discuss methods for moving to adaptive security in Section 5.1.

a pairing for each node in the satisfied formula. While conventional desktop computers should be able to handle such a task for typical formula sizes, this presents a significant challenge for users that manage and view private data on mobile devices where processors are often one to two orders of magnitude slower than their desktop counterparts and battery life is a persistent problem. Interestingly, in tandem there has emerged the ability for users to buy on-demand computing from cloud-based services such as Amazon's EC2 and Microsoft's Windows Azure.

Can cloud services be *securely* used to outsource decryption in Attribute-Based Encryption systems? A naive first approach would be for a user to simply hand over their secret key, SK, to the outsourcing service. The service could then simply decrypt all ciphertexts requested by the user and then transmit the decrypted data. However, this requires complete trust of the outsourcing service; using the secret key the outsourcing service could read any encrypted message intended for the user.

A second approach might be to leverage recent outsourcing techniques [20, 17] based on Gentry's [21] fully homomorphic encryption system. These give outsourcing for general computations and importantly preserve the privacy of the inputs so that the decryption keys and messages can remain hidden. Unfortunately, the overhead for these systems is currently impractical. Gentry and Halevi [22] showed that even for weak security parameters one "bootstrapping" operation of the homomorphic operation would take at least 30 seconds on a high performance machine (and 30 minutes for the high security parameter). Since one such operation would only count for a small constant number of gates in the overall computation, this would need to be repeated many times to evaluate an ABE decryption using the methods above.

Closer to practice, we might leverage recent techniques on secure outsourcing of pairings [16]. These techniques allow a client to outsource a pairing operation to a server. However, the solutions presented in [16] still require the client to compute multiple exponentiations in the target group for every pairing it outsources. These exponentiations can be quite expensive and the work of the client will still be proportional to the size of the policy f. Moreover, every pairing operation in the original protocol will trigger four pairings do be done by the proxy. Thus, the total workload is increased by a factor of at least four from the original decryption algorithm, and the client's bandwidth requirements may actually *increase*. Given these drawbacks, we aim for an ABE outsourcing system that is secure and imposes minimal overhead.

Our Contributions. We give new methods for efficiently and securely outsourcing decryption of ABE ciphertexts. The core change to outsourceable ABE systems is a modified Key Generation algorithm that produces two keys. The first key is a short El Gamal [19] type secret key that must be kept private by the user. The second is what we call a "transformation key", TK, that is shared with a proxy (and can be publicly distributed). If the proxy then receives a ciphertext CT for a function f for which the user's credentials satisfy, it is then able to use the key TK to transform CT into a simple and short El Gamal ciphertext CT$'$ of the same message encrypted under the user's key SK. The user is then able to decrypt with one simple exponentiation. Our system is secure against any malicious proxy. Moreover, the computational effort of the proxy is no more than that used to decrypt a ciphertext in a standard ABE system.

To achieve our results, we create what we call a new key blinding technique. At a high level, the new outsourced key generation algorithm will first run a key generation algorithm from an existing bilinear map based ABE scheme such as [24, 42]. Then it will choose a blinding factor exponent $z \in \mathbb{Z}_p$ (for groups of prime order p) and raise all elements to $z^{-1} \pmod p$. This will produce the transformation key TK, while the blinding factor z can serve as the secret key.

We show that we are able to adapt our outsourcing techniques to both the "Ciphertext-Policy" (CP-ABE) and "Key-Policy" (KP-ABE) types of ABE systems.[2] To

[2]CP-ABE systems behave as we outlined above where a ciphertext

Figure 2: Illustration of how ABE ciphertexts are fetched today.

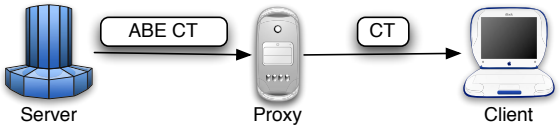

Figure 3: Outsourcing the Decryption: Illustration of how ABE ciphertexts could be transformed by a proxy into much shorter El Gamal-style ciphertexts.

achieve our KP-ABE and CP-ABE outsourcing systems we respectively apply our methodology to the constructions of Goyal et al. [24] and Waters [42]. To prove security of the systems we must show that they remain secure even in the presence of an attacker that acts as a user's proxy. Our first systems and proofs model semantic security for an attacker that tries to eavesdrop on the user. We then extend our systems and proofs to chosen ciphertext attacks where the attack might query the user's decryption routine on maliciously formed ciphertexts to compromise privacy. Our solutions in this setting apply the random oracle heuristic to achieve efficiency near the chosen plaintext versions.

Typical Usage Scenarios. We envision a typical usage scenario in Figures 2 and 3. Here a client sends a single transformation key *once* to the proxy, who can then retrieve (potentially large) ABE ciphertexts that the user is interested in and forward to her (small, constant-size) El Gamal-type ciphertexts. The proxy could be the client's mail server, or the ciphertext server and the proxy could be the same entity, as in a cloud environment.

The savings in bandwidth and local computation time for the client are immediate: a transformed ciphertext is always smaller and faster to decrypt than an ABE ciphertext of [24, 42] (for any policy size). *We emphasize in this useage scenario that the number of transmissions will be the same as in the prior (non-outsourced) solutions.* Thus, the power consumption can only improve with faster computations and smaller transmissions.

Implementation and Evaluation. To evaluate our outsourcing systems, we implemented the CP-ABE version and tested it in an outsourcing environment. Our implementation modified part of the libfenc [25] library, which includes a current CP-ABE implementation. We conducted our experiments on both an ARM-based mobile device and an Intel server to model the user device and proxy respectively.

Outsourcing decryption resulted in significant practical benefits. Decrypting on an ABE ciphertext containing 100 attributes, we found that without the use of a proxy the mobile device would require about 30 seconds of computation time and drain a significant amount of the device's battery. When we applied our outsourcing technique, decrypting the ciphertext took 2 seconds on our Intel server and approximately 60 milliseconds on the mobile device itself.

To demonstrate compatibility with existing infrastructure, we constructed a re-usable platform for outsourcing decryption using the Amazon EC2 service. Our proxy is deployed as a public Amazon Machine Image that can be programmatically instantiated by any application requiring acceleration.

In addition to the core benefits of outsourcing, we discovered other collateral advantages. In existing ABE implementations [6, 25] much of the decryption code is dedicated to determining how a policy is satisfied by a key and executing the corresponding pairing computations of decryption. In our outsourcing solution, most of this code is pushed into the untrusted transformation algorithm, leaving only a much smaller portion on the user's device. This has two advantages. First, the amount of decryption code that needs to reside on a resource constrained user device will be smaller. Actually, all bilinear map operations can be pushed outside. Second, this partitioning will dramatically decrease the size of the trusted code base, removing thousands of lines of complex parsing code. Even without using outsourcing, this partitioning of code is useful.

Related Work: Proxy Re-Encryption. In this work, we show how to delegate (in a true offline sense) the ability to transform an ABE ciphertext on message m into an El Gamal-style ciphertext on the same m, without learning anything about m. This is similar to the concept of *proxy re-encryption* [8, 4] where an untrusted proxy is given a re-encryption key that allows it to transform an encryption under Alice's key of m into an encryption under Bob's key of the same m, without allowing the proxy to learn anything about m.

2 Background

We first give the security definitions for ABE with outsourcing. We then give background information on bilinear maps. Finally, we provide formal definitions for

is associated with a boolean access formula f and a user's key is a set of attributes x, where a user can decrypt if $f(x) = 1$. KP-ABE is useful in applications where we want to have the mirror image semantics where the attributes x are associated with a ciphertext and an access formula f with the key.

access structures and relevant background on Linear Secret Sharing Schemes (LSSS), as taken from [42].

Types of ABE. We consider two distinct varieties of Attribute-Based Encryption: Ciphertext-Policy (CP-ABE) and Key-Policy (KP-ABE). In CP-ABE an *access structure* (policy) is embedded into the ciphertext during encryption, and each decryption key is based an some attribute set S. KP-ABE inverts this relationship, embedding S into the ciphertext and a policy into the key.[3] We capture both paradigms in a generalized ABE definition.

2.1 Access Structures

Definition 1 (Access Structure [5]) *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B,C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An* access structure *(respectively, monotone access structure) is a collection (resp., monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1,P_2,\ldots,P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets.

In our context, the role of the parties is taken by the attributes. Thus, the access structure $\mathbb{A}$ will contain the authorized sets of attributes. We restrict our attention to monotone access structures. However, it is also possible to (inefficiently) realize general access structures using our techniques by defining the "not" of an attribute as a separate attribute altogether. Thus, the number of attributes in the system will be doubled. From now on, unless stated otherwise, by an access structure we mean a monotone access structure.

2.2 ABE with Outsourcing

Let S represent a set of attributes, and $\mathbb{A}$ an access structure. For generality, we will define (I_{enc}, I_{key}) as the inputs to the encryption and key generation function respectively. In a CP-ABE scheme $(I_{enc}, I_{key}) = (\mathbb{A}, S)$, while in a KP-ABE scheme we will have $(I_{enc}, I_{key}) = (S, \mathbb{A})$. A CP-ABE (resp. KP-ABE) scheme with outsourcing functionality consists of five algorithms:

Setup(λ, U)**.** The setup algorithm takes security parameter and attribute universe description as input. It outputs the public parameters PK and a master key MK.

Encrypt$(\mathrm{PK}, M, I_{enc})$**.** The encryption algorithm takes as input the public parameters PK, a message M, and an access structure (resp. attribute set) I_{enc}. It outputs the ciphertext CT.

KeyGen$_{out}(\mathrm{MK}, I_{key})$**.** The key generation algorithm takes as input the master key MK and an attribute set (resp. access structure) I_{key} and outputs a private key SK and a transformation key TK.

Transform$(\mathrm{TK}, \mathrm{CT})$**.** The ciphertext transformation algorithm takes as input a transformation key TK for I_{key} and a ciphertext CT that was encrypted under I_{enc}. It outputs the partially decrypted ciphertext CT$'$ if $S \in \mathbb{A}$ and the error symbol $\perp$ otherwise.

Decrypt$_{out}(\mathrm{SK}, \mathrm{CT}')$**.** The decryption algorithm takes as input a private key SK for I_{key} and a partially decrypted ciphertext CT$'$ that was originally encrypted under I_{enc}. It outputs the message M if $S \in \mathbb{A}$ and the error symbol $\perp$ otherwise.[4]

Why RCCA security? We describe a security model for ABE that support outsourcing. We want a very strong notion of security. The traditional notion of security against adaptive chosen-ciphertext attacks (CCA) is a bit too strong since it does not allow any bit of the ciphertext to be altered, and the purpose of our outsourcing is to compress the size of the ciphertext. We thus adopt a relaxation due to Canetti, Krawczyk and Nielsen [13] called *replayable* CCA security, which allows modifications to the ciphertext provided they cannot change the underlying message in a meaningful way.

RCCA Security Model for ABE with Outsourcing. Figure 4 describes a generalized RCCA security game for both KP-ABE and CP-ABE schemes with outsourcing. We define the *advantage* of an adversary $\mathscr{A}$ in this game as $\Pr[b' = b] - \frac{1}{2}$.

Definition 2 (RCCA-Secure ABE with Outsourcing) *A CP-ABE or KP-ABE scheme with outsourcing is* RCCA-*secure (or secure against* replayable chosen-ciphertext attacks*) if all polynomial time adversaries have at most a negligible advantage in the RCCA game defined above.*

CPA Security. We say that a system is *CPA*-secure (or secure against *chosen-plaintext attacks*) if we remove the Decrypt oracle in both Phase 1 and 2.

Selective Security. We say that a CP-ABE (resp. KP-ABE) system is *selectively* secure if we add an Init stage before Setup where the adversary commits to the challenge value I^*_{enc}.

[3] More intuitively, CP-ABE is often suggested as a means to implement role-based access control, where the user's key attributes correspond the long-term roles and ciphertexts carry an access policy. Key-Policy ABE is more appropriate in applications where ciphertexts may be tagged with attributes (*e.g.,* relating to message content), and each user's access to these ciphertexts determined by a policy in their decryption key. For more on applications, see *e.g.,* [37].

[4] Note that we can implement the standard (non-outsourced) ABE Decrypt algorithm by combining Transform and Decrypt$_{out}$.

Setup. The challenger runs the Setup algorithm and gives the public parameters, PK to the adversary.

Phase 1. The challenger initializes an empty table T, an empty set D and an integer $j = 0$. Proceeding adaptively, the adversary can repeatedly make any of the following queries:

- Create(I_{key}): The challenger sets $j := j+1$. It runs the outsourced key generation algorithm on I_{key} to obtain the pair $(\mathrm{SK}, \mathrm{TK})$ and stores in table T the entry $(j, I_{key}, \mathrm{SK}, \mathrm{TK})$. It then returns to the adversary the transformation key TK.
  Note: Create can be repeatedly queried with the same input.
- Corrupt(i): If there exists an i^{th} entry in table T, then the challenger obtains the entry $(i, I_{key}, \mathrm{SK}, \mathrm{TK})$ and sets $D := D \cup \{I_{key}\}$. It then returns to the adversary the private key SK. If no such entry exists, then it returns $\perp$.
- Decrypt(i, CT): If there exists an i^{th} entry in table T, then the challenger obtains the entry $(i, I_{key}, \mathrm{SK}, \mathrm{TK})$ and returns to the adversary the output of the decryption algorithm on input $(\mathrm{SK}, \mathrm{CT})$. If no such entry exists, then it returns $\perp$.

Challenge. The adversary submits two equal length messages M_0 and M_1. In addition the adversary gives a value I^*_{enc} such that for all $I_{key} \in D$, $f(I_{key}, I^*_{enc}) \neq 1$. The challenger flips a random coin b, and encrypts M_b under I^*_{enc}. The resulting ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restrictions that the adversary cannot

- trivially obtain a private key for the challenge ciphertext. That is, it cannot issue a Corrupt query that would result in a value I_{key} which satisfies $f(I_{key}, I^*_{enc}) = 1$ being added to D.
- issue a trivial decryption query. That is, Decrypt queries will be answered as in Phase 1, except that if the response would be either M_0 or M_1, then the challenger responds with the special message **test** instead.

Guess. The adversary outputs a guess b' of b.

Figure 4: Generalized RCCA Security game for CP- and KP-ABE with outsourcing functionality. For CP-ABE we define the function $f(I_{key}, I_{enc})$ as $f(S, \mathbb{A})$ and for KP-ABE it is defined as $f(\mathbb{A}, S)$. In either case the function f evaluates to 1 iff $S \in \mathbb{A}$.

2.3 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map with the properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ are both efficiently computable.

The schemes we present in this work are provably secure under the Decisional Parallel BDHE Assumption [42] and the Decisional Bilinear Diffie-Hellman assumption (DBDH) [9] in bilinear groups. For reasons of space we will omit a definition of these assumptions here, and refer the reader to the cited works.

2.4 Linear Secret Sharing Schemes

We will make essential use of linear secret-sharing schemes. We adapt our definitions from those in [5]:

Definition 3 (Linear Secret-Sharing Schemes (LSSS)) *A secret-sharing scheme Π over a set of parties $\mathscr{P}$ is called* linear *(over $\mathbb{Z}_p$) if*

1. *The shares of the parties form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix M with ℓ rows and n columns called the share-generating matrix for Π. There exists a function ρ which maps each row of the matrix to an associated party. That is for $i = 1, \ldots, \ell$, the value $\rho(i)$ is the party associated with row i. When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to party $\rho(i)$.*

It is shown in [5] that every linear secret sharing-scheme according to the above definition also enjoys the

linear reconstruction property, defined as follows: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1,2,\ldots,\ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$. It is shown in [5] that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

Like any secret sharing scheme, it has the property that for any unauthorized set $S \notin \mathbb{A}$, the secret s should be information theoretically hidden from the parties in S.

Note on Convention. We use the convention that vector $(1,0,0,\ldots,0)$ is the "target" vector for any linear secret sharing scheme. For any satisfying set of rows I in M, we will have that the target vector is in the span of I.

For any unauthorized set of rows I the target vector is not in the span of the rows of the set I. Moreover, there will exist a vector w such that $w \cdot (1,0,0\ldots,0) = -1$ and $w \cdot M_i = 0$ for all $i \in I$.

Using Access Trees. Some prior ABE works (e.g., [24]) described access formulas in terms of binary trees. Using standard techniques [5] one can convert any monotonic boolean formula into an LSSS representation. An access tree of ℓ nodes will result in an LSSS matrix of ℓ rows.

3 Outsourcing Decryption for Ciphertext-Policy ABE

3.1 A CPA-secure Construction

Our CP-ABE construction is based on the "large universe" construction of Waters [42], which was proven to be selectively CPA-secure under the Decisional q-parallel BDHE assumption for a challenge matrix of size $\ell^* \times n^*$, where $\ell^*, n^* \leq q$.[5] The Setup, Encrypt and (non-outsourced) Decrypt algorithms are identical to [42]. To enable outsourcing we modify the KeyGen algorithm to output a transformation key. We also define a new Transform algorithm, and modify the decryption algorithm to handle outputs of Encrypt as well as Transform. We present the full construction in Figure 5.

Discussion. For generality, we defined the transformation key TK as being created by the master authority. However, we observe that our outsourcing approach above is actually backwards compatible with existing deployments of the Waters system. In particular, one can see that any existing user with her own Waters SK can create a corresponding outsourcing pair $(\mathrm{SK}', \mathrm{TK}')$ by rerandomizing with a random value z.

[5] By "large universe", we mean a system that allows for a super-polynomial number of attributes.

Theorem 3.1 *Suppose the large universe construction of Waters [42, Appendix C] is a selectively CPA-secure CP-ABE scheme. Then the CP-ABE scheme of Figure 5 is a selectively CPA-secure outsourcing scheme.*

Note that the Waters scheme of [42] was proven secure under the Decisional q-parallel BDHE assumption. Due to space constraints, we omit a proof of Theorem 3.1. However, we observe that the proof techniques are quite similar to those used for the RCCA-secure variant we present in the next section.

3.2 An RCCA-secure Construction

We now extend our CPA-secure system to achieve the stronger RCCA-security guarantee. To do so, we borrow some techniques from Fujisaki and Okamoto [18], who (roughly) showed how to transform a CPA-secure encryption scheme into a CCA-secure encryption scheme in the random oracle model. Here we relax to RCCA-security and have the additional challenge of preserving the decryption outsourcing capability.

The Setup and KeyGen algorithms operate exactly as in the CPA-secure scheme, except the public key additionally includes the description of hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$. We now describe the remaining algorithms.

Encrypt$_{rcca}(\mathrm{PK}, \mathscr{M} \in \{0,1\}^k, (M,\rho))$ The encryption algorithm selects a random $R \in \mathbb{G}_T$ and then computes $s = H_1(R, \mathscr{M})$ and $r = H_2(R)$. It then computes $(C_1, D_1), \ldots, (C_\ell, D_\ell)$ as in the CPA-secure construction of Figure 5 (except that s is no longer chosen randomly as part of $\vec{v}$). The ciphertext is published as $\mathrm{CT} =$

$$C = R \cdot e(g,g)^{\alpha s},\ C' = g^s,\ C'' = \mathscr{M} \oplus r,$$
$$(C_1, D_1), \ldots, (C_\ell, D_\ell)$$

along with a description of access structure (M,ρ).

Transform$_{rcca}(\mathrm{TK}, \mathrm{CT})$**.** The transformation algorithm recovers the value $e(g,g)^{s\alpha/z}$ as before. It outputs the partially decrypted ciphertext CT' as $(C, C'', e(g,g)^{s\alpha/z})$.

Decrypt$_{rcca}(\mathrm{SK}, \mathrm{CT})$**.** The decryption algorithm takes as input a private key $\mathrm{SK} = (z, \mathrm{TK})$ and a ciphertext CT. If the ciphertext is not partially decrypted, then the algorithm first executes $\mathrm{Transform}_{out}(\mathrm{TK}, \mathrm{CT})$. If the output is $\perp$, then this algorithm outputs $\perp$ as well. Otherwise, it takes the ciphertext (T_0, T_1, T_2) and computes $R = T_0 / T_2^z$, $\mathscr{M} = T_1 \oplus H_2(R)$, and $s = H_1(R, \mathscr{M})$. If $T_0 = R \cdot e(g,g)^{\alpha s}$ and $T_2 = e(g,g)^{\alpha s/z}$, it outputs $\mathscr{M}$; otherwise, it outputs the error symbol $\perp$.

Setup(λ, U). The setup algorithm takes as input a security parameter and a universe description U. To cover the most general case, we let $U = \{0,1\}^*$. It then chooses a group $\mathbb{G}$ of prime order p, a generator g and a hash function F that maps $\{0,1\}^*$ to $\mathbb{G}$.[a] In addition, it chooses random exponents $\alpha, a \in \mathbb{Z}_p$. The authority sets $\text{MSK} = (g^\alpha, \text{PK})$ as the master secret key. It publishes the public parameters as:

$$\text{PK} = \quad g,\ e(g,g)^\alpha,\ g^a,\ F$$

Encrypt$(\text{PK}, \mathscr{M}, (M, \rho))$ The encryption algorithm takes as input the public parameters PK and a message $\mathscr{M}$ to encrypt. In addition, it takes as input an LSSS access structure (M, ρ). The function ρ associates rows of M to attributes. Let M be an $\ell \times n$ matrix. The algorithm first chooses a random vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$. These values will be used to share the encryption exponent s. For $i = 1$ to ℓ, it calculates $\lambda_i = \vec{v} \cdot M_i$, where M_i is the vector corresponding to the ith row of M. In addition, the algorithm chooses random $r_1, \ldots, r_\ell \in \mathbb{Z}_p$. The ciphertext is published as $\text{CT} =$

$$C = \mathscr{M} \cdot e(g,g)^{\alpha s},\ C' = g^s,$$
$$(C_1 = g^{a\lambda_1} \cdot F(\rho(1))^{-r_1},\ D_1 = g^{r_1}), \ldots, (C_\ell = g^{a\lambda_\ell} \cdot F(\rho(\ell))^{-r_\ell},\ D_\ell = g^{r_\ell})$$

along with a description of (M, ρ).

KeyGen$_{out}(\text{MSK}, S)$ The key generation algorithm runs $\text{KeyGen}(\text{MSK}, S)$ to obtain $\text{SK}' = (\text{PK}, K' = g^\alpha g^{at'}, L' = g^{t'}, \{K'_x = F(x)^{t'}\}_{x \in S})$. It chooses a random value $z \in \mathbb{Z}_p^*$. It sets the transformation key TK as

$$\text{PK}, \quad K = K'^{1/z} = g^{(\alpha/z)} g^{a(t'/z)} = g^{(\alpha/z)} g^{at}, \quad L = L'^{1/z} = g^{(t'/z)} = g^t, \quad \{K_x\}_{x \in S} = \{K_x'^{1/z}\}_{x \in S}$$

and the private key SK as (z, TK).

Transform$_{out}(\text{TK}, \text{CT})$ The transformation algorithm takes as input a transformation key $\text{TK} = (\text{PK}, K, L, \{K_x\}_{x \in S})$ for a set S and a ciphertext $\text{CT} = (C, C', C_1, \ldots, C_\ell)$ for access structure (M, ρ). If S does not satisfy the access structure, it outputs $\perp$. Suppose that S satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. The transformation algorithm computes

$$e(C', K) / \left(e(\textstyle\prod_{i \in I} C_i^{\omega_i}, L) \cdot \prod_{i \in I} e(D_i^{\omega_i}, K_{\rho(i)}) \right) =$$
$$e(g,g)^{s\alpha/z} e(g,g)^{ast} / \left(\textstyle\prod_{i \in I} e(g,g)^{ta\lambda_i \omega_i} \right) = e(g,g)^{s\alpha/z}$$

It outputs the partially decrypted ciphertext CT' as $(C, e(g,g)^{s\alpha/z})$, which can be viewed as the El Gamal ciphertext $(\mathscr{M} \cdot G^{zd}, G^d)$ where $G = e(g,g)^{1/z} \in \mathbb{G}_T$ and $d = s\alpha \in \mathbb{Z}_p$.

Decrypt$_{out}(\text{SK}, \text{CT})$ The decryption algorithm takes as input a private key $\text{SK} = (z, \text{TK})$ and a ciphertext CT. If the ciphertext is not partially decrypted, then the algorithm first executes $\text{Transform}_{out}(\text{TK}, \text{CT})$. If the output is $\perp$, then this algorithm outputs $\perp$ as well. Otherwise, it takes the ciphertext (T_0, T_1) and computes $T_0 / T_1^z = \mathscr{M}$.

Notice that if the ciphertext is already partially decrypted for the user, then she need only compute one exponentiation and no pairings to recover the message.

[a] See Waters [42] for details on how to implement this hash in the standard model. For our purposes, one can think of F as a random oracle.

Figure 5: A CPA-secure CP-ABE outsourcing scheme based on the large-universe construction of Waters [42, Appendix C].

Theorem 3.2 *Suppose the large universe construction of Waters [42, Appendix C] is a selectively CPA-secure CP-ABE scheme. Then the outsourcing scheme above is selectively RCCA-secure in the random oracle model for large message spaces.*[6]

We present a proof of Theorem 3.2 in Appendix A.

4 Outsourcing Decryption for Key-Policy ABE

4.1 A CPA-secure Construction

We now present an outsourcing scheme based on the large universe KP-ABE construction due to Goyal, Pandey, Sahai and Waters [24].[7] The Setup and Encrypt algorithms are identical to [24]. We modify KeyGen to output a transformation key, introduce a Transform algorithm, and then modify the decryption algorithm to handle outputs of Encrypt as well as Transform. The full construction is presented in Figure 6.

Theorem 4.1 *Suppose the GPSW KP-ABE scheme [24] is selectively CPA-secure. Then the KP-ABE scheme of Figure 6 is a selectively CPA-secure outsourcing scheme.*

Discussion. As in the previous construction, we defined the transformation key TK as being created by the master authority. We again note that our outsourcing approach above is actually backwards compatible with existing deployments of the GPSW system.

Due to restrictions on space, we leave the proof of security to the full version of this work [26].

4.2 An RCCA-secure construction

We now extend our above results, which only hold for CPA-security, to the stronger RCCA-security guarantee. Once again, we accomplish this using the techniques from Fujisaki and Okamoto [18]. The Setup and KeyGen algorithms operate exactly as before, except the public key additionally includes the value $e(g,h)^{\alpha}$ (which was already computable from existing values) and the description of hash functions $H_1 : \{0,1\}^* \to \mathbb{Z}_p$ and $H_2 : \{0,1\}^* \to \{0,1\}^k$.

Encrypt$_{rcca}$(PK, $\mathscr{M} \in \{0,1\}^k, S$)**.** The encryption algorithm chooses a random $R \in \mathbb{G}_T$. It then computes $s = H_1(R, \mathscr{M})$ and $r = H_2(R)$. For each $x \in S$ it generates C_x as in the CPA-secure scheme. The ciphertext is published as CT =

$$C = R \cdot e(g,h)^{\alpha s},\ C' = g^s,\ C'' = r \oplus \mathscr{M},\ \{C_x\}_{x \in S}$$

along with a description of S.

Transform$_{rcca}$(TK, CT)**.** The transformation algorithm recovers the value $e(g,h)^{s\alpha/z}$ as before. It outputs the partially decrypted ciphertext CT′ as $(C, C'', e(g,h)^{s\alpha/z})$.

Decrypt$_{rcca}$(SK, CT)**.** The decryption algorithm takes as input a private key SK = (z, TK) and a ciphertext CT. If the ciphertext is not partially decrypted, then the algorithm first executes Transform$_{out}$(TK, CT). If the output is $\perp$, then this algorithm outputs $\perp$ as well. Otherwise, it takes the ciphertext (T_0, T_1, T_2) and computes $R = T_0 / T_2^z$, $\mathscr{M} = T_1 \oplus H_2(R)$, and $s = H_1(R, \mathscr{M})$. If $T_0 = R \cdot e(g,h)^{\alpha s}$ and $T_2 = e(g,h)^{\alpha s/z}$, it outputs $\mathscr{M}$; otherwise, it outputs the error symbol $\perp$.

Theorem 4.2 *Suppose the construction of GPSW [24] is a selectively CPA-secure KP-ABE scheme. Then the outsourcing scheme above is selectively RCCA-secure in the random oracle model for large message spaces.*

See the footnote on Theorem 3.2 for a definition and discussion of "large message spaces". We present a proof of Theorem 4.2 in the full version [26] of this work.

5 Discussion

5.1 Achieving Adaptive Security

The systems we presented were proven secure in the selective model of security. We briefly sketch how we can adapt our techniques to achieve ABE systems that are provably secure in the adaptive model.[8]

Recently, the first ABE systems that achieved adaptive security were proposed by Lewko *et al.* [28] using the techniques of Dual System Encryption [41]. Since the underlying structure of the KP-ABE and CP-ABE schemes presented by Lewko *et al.* is almost respectively identical to the underlying Goyal *et al.* [24] and Waters [42] systems we use, it is possible to adapt our construction techniques to these underlying constructions.[9]

[6] The security of this scheme follows for large message spaces; e.g., k-bit spaces where $k \geq \lambda$, the security parameter. To obtain a secure scheme for smaller message spaces, replace C'' with any CPA-secure symmetric encryption of $\mathscr{M}$ using key $H_2(R)$ and let the range of H_2 be the key space of this symmetric scheme. Since the focus of this work is on efficiency, we'll typically be assuming large enough message spaces and therefore opting for the quicker XOR operation.

[7] This construction was originally described using access trees; here we generalize it to LSSS access structures.

[8] We briefly note that it is simple to prove adaptive security of our schemes in the generic group model like Bethencourt, Sahai, and Waters [7]. Here we are interested in proofs under non-interactive assumptions.

[9] The main difference in terms of the constructions is that the systems proposed by Lewko *et al.* are set in composite order groups where the "core scheme" sits in one subgroup. The primary novelty of their work is in developing adaptive proofs of security for ABE systems.

Setup(λ, U). The setup algorithm takes as input a security parameter and a universe description U. To cover the most general case, we let $U = \{0,1\}^*$. It then chooses a group $\mathbb{G}$ of prime order p, a generator g and a hash function F that maps $\{0,1\}^*$ to $\mathbb{G}$.[a] In addition, it chooses random values $\alpha \in \mathbb{Z}_p$ and $h \in \mathbb{G}$. The authority sets $\text{MSK} = (\alpha, \text{PK})$ as the master secret key. The public key is published as

$$\text{PK} = \quad g, \quad g^{\alpha}, \quad h, \quad F$$

Encrypt$(\text{PK}, \mathscr{M}, S)$. The encryption algorithm takes as input the public parameters PK, a message $\mathscr{M}$ to encrypt, and a set of attributes S. It chooses a random $s \in \mathbb{Z}_p$. The ciphertext is published as $\text{CT} = (S, C)$ where

$$C = \mathscr{M} \cdot e(g,h)^{\alpha s},\ C' = g^s,\ \{C_x = F(x)^s\}_{x \in S}.$$

KeyGen$_{out}(\text{MSK}, (M, \rho))$. Parse $\text{MSK} = (\alpha, \text{PK})$. The key generation algorithm runs $\text{KeyGen}((\alpha, \text{PK}), (M, \rho))$ to obtain $\text{SK}' = (\text{PK}, (D_1' = h^{\lambda_1} \cdot F(\rho(1))^{r_1'}, R_1' = g^{r_1'}), \ldots, (D_\ell', R_\ell'))$. Next, it chooses a random value $z \in \mathbb{Z}_p$, computes the transformation key TK as below, and outputs the private key as (z, TK). Denoting r_i'/z as r_i, TK is computed as:

$$\text{PK}, \quad (D_1 = D_1'^{1/z} = h^{\lambda_1/z} \cdot F(\rho(1))^{r_1},\ R_1 = R_1'^{1/z} = g^{r_1}),\ \ldots\ , (D_\ell = D_\ell'^{1/z},\ R_\ell = R_\ell'^{1/z})$$

Transform$_{out}(\text{TK}, \text{CT})$. The transformation algorithm takes as input a transformation key $\text{TK} = (\text{PK}, (D_1, R_1), \ldots, (D_\ell, R_\ell))$ for access structure (M, ρ) and a ciphertext $\text{CT} = (C, C', \{C_x\}_{x \in S})$ for set S. If S does not satisfy the access structure, it outputs $\perp$. Suppose that S satisfies the access structure and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in S\}$. Then, let $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ be a set of constants such that if $\{\lambda_i\}$ are valid shares of any secret s according to M, then $\sum_{i \in I} \omega_i \lambda_i = s$. The transformation algorithm computes

$$e(C', \prod_{i \in I} D_i^{\omega_i}) / \left(\prod_{i \in I} e(R_i, C_{\rho(i)}^{\omega_i}) \right) \quad = \quad e(g^s, \prod_{i \in I} h^{\lambda_i \omega_i / z} \cdot F(\rho(i))^{r_i \omega_i}) / \left(\prod_{i \in I} e(g^{r_i}, F(\rho(i))^{s \omega_i}) \right)$$

$$= e(g,h)^{s\alpha/z} \cdot \prod_{i \in I} e(g^s, F(\rho(i))^{r_i \omega_i}) / \left(\prod_{i \in I} e(g^{r_i}, F(\rho(i))^{s \omega_i}) \right) \quad = \quad e(g,h)^{s\alpha/z}$$

It outputs the partially decrypted ciphertext CT' as $(C, e(g,h)^{s\alpha/z})$, which can be viewed as the El Gamal ciphertext $(\mathscr{M} \cdot G^{zd}, G^d)$ where $G = e(g,h)^{1/z} \in \mathbb{G}_T$ and $d = s\alpha \in \mathbb{Z}_p$.

Decrypt$_{out}(\text{SK}, \text{CT})$. The decryption algorithm takes as input a private key $\text{SK} = (z, \text{TK})$ and a ciphertext CT. If the ciphertext is not partially decrypted, then the algorithm first executes $\text{Transform}_{out}(\text{TK}, \text{CT})$. If the output is $\perp$, then this algorithm outputs $\perp$ as well. Otherwise, it takes the ciphertext (T_0, T_1) and computes $T_0 / T_1^z = \mathscr{M}$.

[a] Goyal *et al.* [24] give a standard model instantiation for F using an n-wise independent hash function (in the exponents) with the restriction that any ciphertext can contain at most n attributes. For our purposes, one can think of F as a random oracle.

Figure 6: A CPA-secure KP-ABE outsourcing scheme based on the large-universe construction of Goyal, Pandey, Sahai and Waters [24].

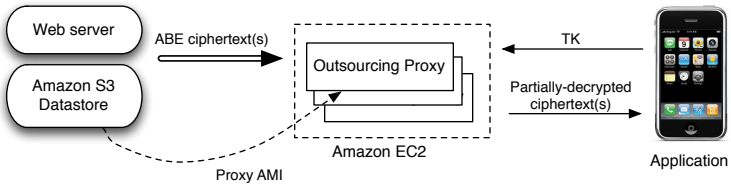

Figure 7: Architecture and data flow for our cloud-based outsourcing proxy. An application programmatically instantiates one or more instances of the outsourcing proxy, which is loaded from a public Amazon Machine Image (AMI) in the S3 storage cloud. Next the application uploads a transform key TK to the proxy, and subsequently instructs the proxy to obtain ciphertexts from remote web servers or from locations within the S3 storage cloud. The proxy transforms the ciphertexts and returns the partially-decrypted result to the application, which completes decryption to obtain a plaintext. We emphasize that the setup step including uploading the transformation key only needs to be done once; subsequently, many decryption steps can follow. In an alternative configuration (not shown) the application can also upload ABE ciphertexts to the proxy from its local storage. We note the first configuration conflates the ciphertext delivery and partial decryption and thus requires no additional transmissions relative to non outsourcing solutions. The alternative will require an round trip for each outsourcing operation.

One might hope that the proof of adaptive security could be a black box reduction to the adaptively secure schemes of Lewko *et al.* Unfortunately, this seems infeasible. Consider any direct black box reduction to the security of the underlying scheme. When the attacker makes a query to some transformation key, the reduction algorithm has two options. First, it could ask the security game for the underlying ABE system for a private key. Yet, it might turn out that the key both is never corrupted and is capable of decryption for the eventual challenge ciphertext. In this case the simulator will have to abort. A second option is for the reduction algorithm not to ask for such a key, but fill in the transformation key itself. However, if that user's key is later corrupted it will be difficult for the reduction to both ask for such a private key *and* match it to the published transformation key.

Accordingly, to prove security one needs to make a direct Dual-System encryption type proof. The proof would go along the lines of Lewko *et al.*, with the exception that in the hybrid stage of the proof *all* private keys and transformational keys will be set (one by one) to be semi-functional including those that could decrypt the eventual challenge ciphertext. In the Lewko *et al.* proof giving a private key that could decrypt the challenge ciphertext would undesirably result in the simulator producing observably incorrect correlations between the challenge ciphertext and keys. However, if we only give out the transformation part of such a key (and keep the whole private key hidden) then this correlation will remain hidden. This part of the argument is somewhat similar to the work of Lewko, Rouselakis, and Waters [29], who show that in their leakage resilient ABE scheme if only part of a private key is leaked such a correlation will be hidden.

5.2 Checking the Transformation

In the description of our systems a proxy will be able to transform any ABE ciphertext into a short ciphertext for the user. While the security definitions show that an attacker will not be able to learn an encrypted message, there is no guarantee on the transformation's correctness. In some applications a user might want to request the transformation of a particular ciphertext and (efficiently) check that the transformation was indeed done correctly (assuming the original ciphertext was valid). It is easy to adapt our RCCA systems to such a setting. Since decryption results in recovery of the ciphertext randomness, one can simply add a tag to the ciphertext as $H'(r)$, where H' is a different hash function modeled as a random oracle and r is the ciphertext randomness. On recovery of r the user can compute $H'(r)$ and make sure it matches the tag.

6 Performance in Practice

To validate our results, we implemented the CPA-secure CP-ABE of Section 3 as an extension to the libfenc Attribute Based Encryption library [25]. We then used this as a building block for a platform for accelerating ABE decryption through cloud-based computing resources.

The core of our solution is a virtualized outsourcing "proxy" that runs in the Amazon Elastic Compute Cloud (EC2). Our proxy exists as a machine image that can be programmatically instantiated by any application that requires assistance with ABE decryption. As we demonstrate below, this proxy is particularly useful for accelerating decryption on constrained devices such as mobile phones. However, the system can be used in any application where significant numbers of ABE decryptions must be performed, *e.g.,* in large-scale search op-

erations.[10] The use of on-demand computing is particularly well-suited to our outsourcing techniques, since we do not require trusted remote servers or long-term storage of secrets.

System Architecture. Figure 7 illustrates the architecture of our outsourcing platform. The proxy is stored in Amazon's S3 datastore as a public Amazon Machine Image (AMI), which wraps a standard Linux/Apache distribution along with the code needed to execute the Transform algorithm. Applications can remotely instantiate the proxy and upload a TK corresponding to a particular ABE decryption key.[11] Depending on the use case, they can either push ciphertexts to the proxy for transformation, or direct the proxy to retrieve ABE ciphertexts from remote locations such as the web or the Amazon S3 storage cloud. The latter technique is helpful when accessing remotely-held records on a mobile device, since the proxy transformation dramatically reduces the mobile device's bandwidth requirements vs. downloading and decrypting each ABE ciphertext locally. This can significantly enhance device battery life.

6.1 Performance: Microbenchmarks

To evaluate the performance of our CPA-secure CP-ABE outsourcing scheme in isolation (without confounding factors such as network lag, file I/O, etc.) we conducted a series of microbenchmarks using the libfenc implementation. For consistency, we ran these tests on two dedicated hardware platforms: a 3GHz Intel Core Duo platform with 4GB of RAM running 32-bit Linux Kernel version 2.6.32, and a 412MHz ARM-based iPhone 3G with 128MB of RAM running iOS 4.0.[12] We instantiated the ABE schemes using a 224-bit MNT elliptic curve from the Stanford Pairing-Based Crypto library [30].[13]

The existing libfenc implementation implements the Waters scheme using a Key Encapsulation variant. For backwards compatibility, we adopted this approach in our implementation as well. Herein, the ciphertext carries a symmetric session key k that is computed at encryption time as $k = H(e(g,g)^{\alpha s})$. The element $C = \mathscr{M} \cdot e(g,g)^{\alpha s}$ is omitted from the ciphertext, and any data payload must be carried via a separate symmetric encryption under k. The practical impact of this approach is that the ABE ciphertexts (and partially-decrypted ciphertexts) are shortened by one element of $\mathbb{G}_T$.

Experimental setup. Both decryption time and ciphertext size in the CP-ABE scheme depend on the complexity of the ciphertext's policy. To capture this in our experiments, we first generated a collection of 100 distinct ciphertext policies of the form (A_1 AND A_2 AND ... AND A_N), where each A_i is an attribute, for values of N increasing from 1 to 100. In each case we constructed a corresponding decryption key that contained the N attributes necessary for decryption. This approach ensures that the decryption procedure depends on all N components of the ciphertext and is a reasonable sample of a complex policy.

To obtain our baseline results, we encapsulated a random 128-bit symmetric key under each of these 100 different policies, then decrypted the resulting ABE ciphertext using the normal (non-outsourced) Decrypt algorithm.[14] To smooth any experimental variability, we repeated each of our experiments 100 times on the Intel device (due to the time consuming nature of the experiments, we repeated the test only 30 times on the ARM device) and averaged to obtain our decryption timings. Figure 8 shows the size of the resulting ciphertexts as a function of N, along with the measured decryption times on our Intel and ARM test platforms.

Next, we evaluated the algorithms by generating a Transform Key (TK) from the appropriate N-attribute ABE decryption key and applying the Transform algorithm to the ABE ciphertext using this key.[15] Finally we decrypted the resulting transformed ciphertext. Figure 8 shows the time required for each of those operations.

Discussion. As expected, the ABE ciphertext size and decryption/transform time were linear in the complexity of the ciphertext's policy (N). However, our results illustrate the surprisingly high constants. Encrypting under a 100-component ciphertext policy produced an unwieldy 25KB of ABE ciphertext. The relatively fast Intel processor required nearly 2 full seconds to decrypt this value. By comparison, the same machine can perform a 1024-bit RSA decryption in 1.7 *milliseconds*.[16]

The results were more dramatic on the mobile device. Decrypting a 100-component ciphertext policy on the

[10] Indeed, since cloud computing platforms support the creation of multiple proxy instances, servers can rapidly scale their outsourcing capability up and down to meet demand.

[11] The proxy requires only one TK to decrypt an unlimited number of ciphertexts. However, a proxy can be shared by multiple users, each with their own TK.

[12] Note that our tests were single-threaded, and thus used resources from only a single core of the Intel processor. In all cases we conducted our timing experiments with accessible background services disabled, and with the mobile device connected to a power source.

[13] Although we define our schemes in the symmetric bilinear group setting, the MNT curve choice required that we implement the scheme in asymmetric groups with a pairing of the form $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. As a result we assigned various elements of the ciphertext and key to the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with the aim of minimizing ciphertext size.

[14] Note that for this experiment we did not employ any symmetric encryption, hence all times and ciphertext sizes refer to the ABE key encapsulation ciphertext.

[15] We used the "backwards-compatible" key generation approach described in Section 3.1 to derive a TK from a standard ABE decryption key, rather than having the PKG generate the TK directly. This allowed us to retain compatibility with the existing CP-ABE implementation.

[16] Measured with OpenSSL 1.0 [40].

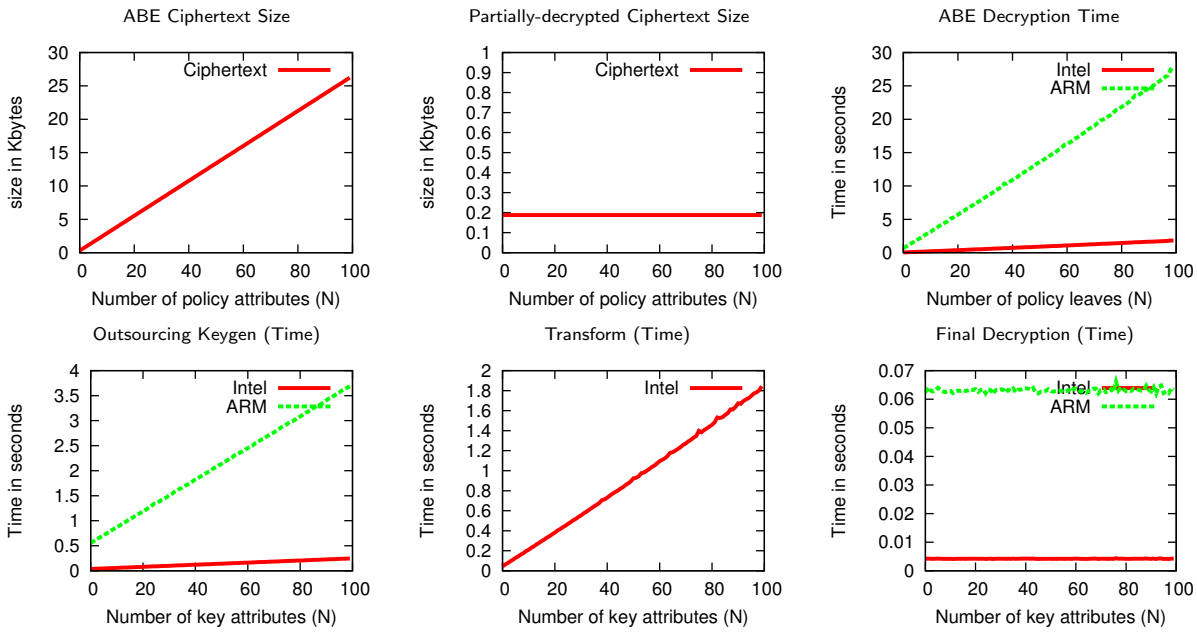

Figure 8: Microbenchmark results for our CP-ABE scheme with outsourcing. Timing results are provided for both Intel and ARM platforms. Key generation times represent the time to convert a standard ABE decryption key into an outsourcing key, using the "backwards-compatible" approach described in Section 3.1. "Final decryption" refers to the decryption of a partially-decrypted ciphertext. Note that we present the Transform timing results for the Intel platform only, since we view this as the more likely outsourcing platform. Intel (resp. ARM) timings represent the average of 100 (resp. 30) test iterations.

ARM processor required nearly *30 seconds* of sustained computation. Even at lower policy complexities, our results seem problematic for implementers looking to deploy unassisted ABE on limited computing devices.

Outsourcing substantially reduced both ciphertext size and the time needed to decrypt the partially-decrypted ciphertext. Each partially-decrypted ciphertext was a fixed 188 bytes in size, regardless of the original ciphertext's CP-ABE policy. Furthermore, the final decryption process required only 4ms on the Intel processor and a manageable 60ms on ARM.[17] Thus, it appears that outsourcing can provide a noticeable decryption time advantage for ciphertexts with 10 or more attributes.

Other Implementation Remarks. There are several optimizations and tradeoffs one might explore that could impact both the performance of the existing ABE scheme and our outsourced scheme. We chose to use the PBC library due to its use in the libfenc system and its simple API. However, PBC does not include all of the latest optimizations discussed in the research literature. Other future optimizations could include the use of multi-pairings for decryption. We emphasize that while using such optimizations to the existing ABE systems could give some performance improvements, they will not improve the size of ABE ciphertexts. Furthermore, decryption time will still be linear in the size of the satisfied formula, whereas our outsourcing technique transforms the final decryption step to a short El-Gamal-type ciphertext.

A note on policy complexity. The reader might assume that 50- or 100-component policies are rare in practice. In fact, we observed that it is relatively easy to arrive at highly complex policies in typical use cases. This is particularly true when using policies that contain integer comparison operators, *e.g.,* "AGE $<$ 30". The libfenc library implements integer comparison operators using the technique of Bethencourt et al. [7]: prior to encryption, each comparison operator is converted into a boolean policy circuit composed of OR and AND gates, and the resulting policy is applied to the ciphertext. Comparing an attribute to a fixed n-bit integer adds approximately n components to the policy. For example, without special optimizations, a restriction window involving a Unix time value ($x <$ KEY_CREATION_TIME $< y$) increases the policy size by approximately 64 components.

[17] We conducted our experiments on the CPA-secure version of our scheme. The primary performance differences in the RCCA version are an extra exponentiation in $\mathbb{G}_T$ and some additional bytes.

Operation	local-only *(sec)*	local+web *(sec/kb)*	proxy *(sec/kb)*	proxy+web *(sec/kb)*
New proxy instantiation	·	·	93.4 sec	93.4 sec
Restart existing proxy instance	·	·	45 sec	45 sec
Generate & set 70-element transform key	·	·	2.9 sec	2.9 sec
Decryption:				
((DOCTOR OR NURSE) AND INSTITUTION)	1.1s	1.2s/1.1k	.2s/1.4k	.2s/0.4k
(DOCTOR AND TIME $>$ 1262325600 AND TIME $<$ 1267423200)	17.3s	17.3s/22.8k	1.2s/23.2k	1.2s/0.4k

Figure 9: Some average performance results for the proxy-enhanced iHealthEHR application running on our iPhone 3G. From left to right, "local-only" indicates device-local decryption and storage of ciphertexts, "local+web" indicates that ciphertexts were downloaded from a web server and decrypted at the device. "proxy" indicates local ciphertext storage with proxy outsourcing. "proxy+web" indicates that ciphertexts were obtained from the web *via* the proxy. Where relevant we provide both timings *and* total bandwidth transferred (up+down) from the device. Note that proxy launch times exhibit some variability depending on factors outside of our control.

6.2 Performance: Mobile Example

To validate our ideas in a real application, we incorporated outsourcing into the iPhone viewer component of iHealthEHR [3], an experimental system for distributing Electronic Health Records (EHRs). Since EHRs can contain highly sensitive data, iHealthEHR uses CP-ABE to perform end-to-end encryption of records from the origination point to the viewing device. Distinct ciphertext policies may be applied to each node in an individual's health record (e.g., to admit special permissions for psychiatric records). iHealthEHR supports both local and cloud-based storage of records.

We modified the iPhone application to remotely instantiate our outsourcing proxy on startup, using a "small" server instance within Amazon's storage cloud.[18] In our experiments we found that the first EC2 instantiation required anywhere from 1-3 minutes, presumably depending on the system's load. However, once the proxy was launched, it could be left running indefinitely and shared by many different users with different TKs, or — when not in use — paused and brought back to full operation in as little as 30 seconds (with an average closer to 45 seconds). During this startup interval we set the application to locally process all decryption operations. Once the proxy signaled its availability, the application pushed a TK to it via HTTP, and outsourced all further decryption operations.

To evaluate the performance implications, we conducted experiments on the system with outsourcing enabled and disabled, considering four likely usage scenarios. In the first scenario (local-only), we conducted device-local decryption on ciphertexts stored locally in the device's Flash memory. In the second scenario (local+web) we downloaded ciphertexts from a web server, then decrypted them locally at the device. In the third scenario (proxy), we stored ciphertexts locally and then *uploaded* them to the proxy for transformation. In the final scenario (proxy+web) ciphertexts were retrieved from a web server by the proxy, then Transformed before being sent to the device. In each case we measured the time required to decrypt, along with the total bandwidth transmitted and received by the device (excepting the local-only case, which did not employ the network connection). The results are summarized in Figure 9.

[18] According to Amazon's documentation, a small EC2 instance provides "the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor" and 1.7GB of RAM, at a cost of USD $0.085/hr. [1].

7 Hardening ABE Implementations

Thus far we described outsourcing solely as a means to improve decryption *performance*. In certain cases outsourcing can also be used to enhance security. By way of motivation, we observe that ABE implementations tend to be relatively complex compared to implementations of other public-key encryption schemes. For example, libfenc's policy handling components alone comprise nearly 3,000 lines of C code, excluding library dependencies. It has been observed that the number of vulnerabilities in a software product tends to increase in proportion to the code's complexity [34].

It is common for designers to mitigate software issues by sandboxing vulnerable processes *e.g.,* [33], or through techniques that isolate security-sensitive functions within a process [32]. McCune *et al.* recently proposed TrustVisor [31], a specialized hypervisor designed to protect and isolate security-sensitive "Pieces of Application Logic" (PALs) from less sensitive code.

We propose outsourcing as a tool to harden ABE implementations in platforms with code isolation. For example, in a system equipped with TrustVisor, implementers can embed the relatively simple key generation and Decrypt$_{out}$ routines in security-sensitive code (*e.g.,* a TrustVisor PAL) and use outsourcing to push the remaining calculations into non-sensitive code. This not

only reduces the size of the sensitive code base, it also simplifies parameter validation for the PAL (since the partially-decrypted ABE ciphertext is substantially less complex than the original). We refer to this technique as "self-outsourcing" and note that it can also be used in systems containing hardware security modules (*e.g.,* cryptographic smart cards). Moreover, based on our experiments of Section 6, we estimate that this approach will have a minimal impact on performance.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

[1] Amazon EC2 FAQs. `http://aws.amazon.com/ec2/faqs/`, November 2010.

[2] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *CRYPTO*, pages 205–222, 2005.

[3] Joseph A. Akinyele, Christoph U. Lehmann, Matthew Green, Matthew W. Pagano, Zachary N. J. Peterson, and Aviel D. Rubin. Self-protecting electronic medical records using Attribute-Based Encryption. Cryptology ePrint Archive, Report 2010/565, 2010. Available from `http://eprint.iacr.org/`.

[4] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. In *NDSS*, pages 29–43, 2005.

[5] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[6] John Bethencourt. Ciphertext-policy attribute-based encryption library. Available from `http://acsc.cs.utexas.edu/cpabe`, May 2010.

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

[8] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.

[9] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[10] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

[12] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[13] Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, pages 565–582, 2003.

[14] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

[15] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[16] Benoît Chevallier-Mames, Jean-Sébastien Coron, Noel McCullagh, David Naccache, and Michael Scott. Secure delegation of elliptic-curve pairing. In *CARDIS*, pages 24–35, 2010.

[17] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, pages 483–501, 2010.

[18] Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO '99*, volume 1666, pages 537–554, 1999.

[19] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[20] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.

[21] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

[22] Craig Gentry and Shai Halevi. Implementing Gentry's fully-homomorphic encryption scheme. In *EUROCRYPT*, pages 129–148, 2011.

[23] Vipul Goyal, Abishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP*, pages 579–591, 2008.

[24] 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.

[25] Matthew Green, Ayo Akinyele, and Michael Rushanan. libfenc: The Functional Encryption Library. Available from `http://code.google.com/p/libfenc`.

[26] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of ABE ciphertexts, 2011. The full version of this paper is available from the Cryptology ePrint Archive.

[27] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[28] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.

[29] Allison Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

[30] Ben Lynn. The Stanford Pairing Based Crypto Library. Available from `http://crypto.stanford.edu/pbc`.

[31] Jonathan M. McCune, Yanlin Li, Ning Qu, Zongwei Zhou, Anupam Datta, Virgil D. Gligor, and Adrian Perrig. TrustVisor: Efficient TCB Reduction and Attestation. In *IEEE Symposium on Security and Privacy*, pages 143–158, May 2010.

[32] Jonathan M. McCune, Bryan Parno, Adrian Perrig, Michael K. Reiter, and Arvind Seshadri. Minimal tcb code execution (extended abstract). In *IEEE Symposium on Security and Privacy*, pages 267–272, 2007.

[33] Elinor Mills. Chrome OS security: 'Sandboxing' and auto updates. eWeek., 2009.

[34] Subhas C. Misra and Virendra C. Bhavsar. Relationships between selected software measures and latent bug-density: guidelines for improving quality. In *ICCSA'03*, pages 724–732, 2003.

[35] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[36] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.

[37] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security*, pages 99–112, 2006.

[38] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[39] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[40] The OpenSSL Project v1.0. OpenSSL: The open source toolkit for SSL/TLS. `www.openssl.org`, April 2010.

[41] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

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

A Proof of Theorem 3.2

Proof. Suppose there exists a polynomial-time adversary $\mathscr{A}$ that can attack our scheme in the selective RCCA-security model for outsourcing with advantage ε. We build a simulator $\mathscr{B}$ that can attack the Waters scheme of [42, Appendix C] in the selective CPA-security model with advantage ε minus a negligible amount. In [42] the Waters scheme is proven secure under the decisional q-parallel BDHE assumption.

Init. The simulator $\mathscr{B}$ runs $\mathscr{A}$. $\mathscr{A}$ chooses the challenge access structure (M^*, ρ^*), which $\mathscr{B}$ passes on to the Waters challenger as the structure on which it wishes to be challenged.

Setup. The simulator $\mathscr{B}$ obtains the Waters public parameters $\text{PK} = g, e(g,g)^{\alpha}, g^{a}$ and a description of the hash function F. It sends these to $\mathscr{A}$ as the public parameters.

Phase 1. The simulator $\mathscr{B}$ initializes empty tables T, T_1, T_2, an empty set D and an integer $j = 0$. It answers the adversary's queries as follows:

- Random Oracle Hash $H_1(R, \mathscr{M})$: If there is an entry $(R, \mathscr{M}, s)$ in T_1, return s. Otherwise, choose a random $s \in \mathbb{Z}_p$, record $(R, \mathscr{M}, s)$ in T_1 and return s.
- Random Oracle Hash $H_2(R)$: If there is an entry (R, r) in T_2, return r. Otherwise, choose a random $r \in \{0,1\}^k$, record (R, r) in T_2 and return r.
- Create((S)): $\mathscr{B}$ sets $j := j + 1$. It now proceeds one of two ways.
  - If S satisfies (M^*, ρ^*), then it chooses a "fake" transformation key as follows: choose a random $d \in \mathbb{Z}_p$ and run $\text{KeyGen}((d, \text{PK}), S)$ to obtain SK'. Set $\text{TK} = \text{SK}'$ and set $\text{SK} = (d, \text{TK})$. Note that the pair (d, TK) is not well-formed, but that TK is properly distributed if d was replaced by the unknown value $z = \alpha/d$.
  - Otherwise, it calls the Waters key generation oracle on S to obtain the key $\text{SK}' = (\text{PK}, K', L', \{K'_x\}_{x \in S})$. (Recall that in the non-outsourcing CP-ABE game, the Create and Corrupt functionalities are combined in one oracle.) The algorithm chooses a random value $z \in \mathbb{Z}_p$ and sets the transformation key TK as $(\text{PK}, \quad K = K'^{1/z}, L = L'^{1/z}, \quad \{K_x\}_{x \in S} = \{K'^{1/z}_x\}_{x \in S})$ and the private key as (z, TK).

  Finally, store $(j, S, \text{SK}, \text{TK})$ in table T and return TK to $\mathscr{A}$.
- Corrupt(i): $\mathscr{A}$ cannot ask to corrupt any key corresponding to the challenge structure (M^*, ρ^*). If there exists an ith entry in table T, then $\mathscr{B}$ obtains the entry $(i, S, \text{SK}, \text{TK})$ and sets $D := D \cup \{S\}$. It then returns SK to $\mathscr{A}$, or $\perp$ if no such entry exists.
- Decrypt(i, CT): Without loss of generality, we assume that all ciphertexts input to this oracle are already partially decrypted. Recall that both $\mathscr{B}$ and $\mathscr{A}$ have access to the TK values for all keys created, so either can execute the transformation operation. Let $\text{CT} = (C_0, C_1, C_2)$ be associated with structure (M, ρ). Obtain the record $(i, S, \text{SK}, \text{TK})$ from table T. If it is not there or $S \notin (M, \rho)$, return $\perp$ to $\mathscr{A}$.
  If key i does not satisfy the challenge structure (M^*, ρ^*), proceed as follows:
  1. Parse $\text{SK} = (z, \text{TK})$. Compute $R = C_0 / C_2^z$.
  2. Obtain the records $(R, \mathscr{M}_i, s_i)$ from table T_1. If none exist, return $\perp$ to $\mathscr{A}$.
  3. If in this set, there exists indices $y \neq x$ such that $(R, \mathscr{M}_y, s_y)$ and $(R, \mathscr{M}_x, s_x)$ are in table T_1, $\mathscr{M}_y \neq \mathscr{M}_x$ and $s_y = s_x$, then $\mathscr{B}$ aborts the simulation.
  4. Otherwise, obtain the record (R, r) from table T_2. If it does not exist, $\mathscr{B}$ outputs $\perp$.
  5. For each i, test if $C_0 = R \cdot e(g,g)^{\alpha s_i}$, $C_1 = \mathscr{M}_i \oplus r$ and $C_2 = e(g,g)^{\alpha s_i / z}$.
  6. If there is an i that passes the above test, output the message $\mathscr{M}_i$; otherwise, output $\perp$. (Note: at most one value of s_i, and thereby one index i, can satisfy the third check of the above test.)

  If key i does satisfy the challenge structure (M^*, ρ^*), proceed as follows:
  1. Parse $\text{SK} = (d, \text{TK})$. Compute $\beta = C_2^{1/d}$.
  2. For each record $(R_i, \mathscr{M}_i, s_i)$ in table T_1, test if $\beta = e(g,g)^{s_i}$.
  3. If zero matches are found, $\mathscr{B}$ outputs $\perp$ to $\mathscr{A}$.
  4. If more than one matches are found, $\mathscr{B}$ aborts the simulation.
  5. Otherwise, let $(R, \mathscr{M}, s)$ be the sole match. Obtain the record (R, r) from table T_2. If it does not exist, $\mathscr{B}$ outputs $\perp$.
  6. Test if $C_0 = R \cdot e(g,g)^{\alpha s}$, $C_1 = \mathscr{M} \oplus r$ and $C_2 = e(g,g)^{ds}$.
  7. If all tests pass, output $\mathscr{M}$; else, output $\perp$.

Challenge. Eventually, $\mathscr{A}$ submits a message pair $(\mathscr{M}_0^*, \mathscr{M}_1^*) \in \{0,1\}^{2 \times k}$. $\mathscr{B}$ acts as follows:

1. $\mathscr{B}$ chooses random "messages" $(\mathscr{R}_0, \mathscr{R}_1) \in \mathbb{G}_T^2$ and passes them on to the Waters challenger to obtain a ciphertext $\text{CT} = (C, C', \{C_i\}_{i \in [1,\ell]})$ under (M^*, ρ^*).
2. $\mathscr{B}$ chooses a random value $C'' \in \{0,1\}^k$.
3. $\mathscr{B}$ sends to $\mathscr{A}$ the challenge ciphertext $\text{CT}^* = (C, C', C'', \{C_i\}_{i \in [1,\ell]})$.

Phase 2. The simulator $\mathscr{B}$ continues to answer queries as in Phase 1, except that if the response to a Decrypt query would be either $\mathscr{M}_0^*$ or $\mathscr{M}_1^*$, then $\mathscr{B}$ responds with the message **test** instead.

Guess. Eventually, $\mathscr{A}$ must either output a bit or abort, either way $\mathscr{B}$ ignores it. Next, $\mathscr{B}$ searches through tables T_1 and T_2 to see if the values $\mathscr{R}_0$ or $\mathscr{R}_1$ appear as the first element of any entry (i.e., that $\mathscr{A}$ issued a query of the form $H_1(\mathscr{R}_i, \cdot)$ or $H_2(\mathscr{R}_i)$.) If neither or both values appear, $\mathscr{B}$ outputs a random bit as its guess. If only value $\mathscr{R}_b$ appears, then $\mathscr{B}$ outputs b as its guess.

This ends the description of the simulation. Due to space limitations, our analysis of this simulation appears in the full version of this work [26].

□