Online/Offline Attribute-Based Encryption

Susan Hohenberger[1] and Brent Waters[2]

Johns Hopkins University and University of Texas at Austin

Abstract. Attribute-based encryption (ABE) is a type of public key encryption that allows users to encrypt and decrypt messages based on user attributes. For instance, one can encrypt a message to any user satisfying the boolean formula ("crypto conference attendee" AND "PhD student") OR "IACR member". One drawback is that encryption and key generation computational costs scale with the complexity of the access policy or number of attributes. In practice, this makes encryption and user key generation a possible bottleneck for some applications.
To address this problem, we develop new techniques for ABE that split the computation for these algorithms into two phases: a preparation phase that does the vast majority of the work to encrypt a message or create a secret key *before* it knows the message or the attribute list/access control policy that will be used (or even the size of the list or policy). A second phase can then rapidly assemble an ABE ciphertext or key when the specifics become known. This concept is sometimes called "online/offline" encryption when only the message is unknown during the preparation phase; we note that the addition of unknown attribute lists and access policies makes ABE significantly more challenging.
One motivating application for this technology is mobile devices: the preparation work can be performed while the phone is plugged into a power source, then it can later rapidly perform ABE operations on the move without significantly draining the battery.

1 Introduction

Attribute-Based Encryption (ABE) was introduced by Sahai and Waters [20] as a more expressive form of encryption where one can encrypt according to some policy. For example, in a large corporate setting one might encrypt data to the policy of ("Procurement" AND "Manager") OR "Accounting". There are two main flavors of ABE. In Key-Policy ABE [10], a key is associated with a boolean formula ϕ and a ciphertext with a set S of attributes. One can decrypt iff the set S satisfies the formula ϕ. Alternatively, in Ciphertext-Policy ABE the roles are flipped; a key is associated with a set of attributes and the ciphertext with an access formula.

One challenge in building systems that use Attribute-Based Encryption is that the added functionality may come with a significant cost compared to standard public key cryptography. Consider a Key-Policy ABE system. Here the encryption time will scale with the number of attributes assigned to the ciphertext and key generation time will scale with the size of the boolean formula

ascribed to a user's private key. These costs could impact several applications. If the encryption algorithm is run on a mobile device, encryption time and battery power are of large importance. In other applications, authority servers that generate users' private keys may become a bottleneck. In both of these scenarios, *an exacerbating factor is that the cost for operations may vary widely between each ciphertext and key; thus forcing a system to provision for a load that matches a worst case scenario.* See [4, 18, 23] for further ABE performance cost details.

In this work, we aim to mitigate this problem by introducing methods for online/offline encryption and key generation in Attribute-Based Encryption. By moving the majority of the cost of an encryption and key generation into an offline phase, a system will be able to smooth the computational (and power) demand over a longer range of time, and thus only need the resources to handle the average case load.

Applications for this Technology One motivating application for splitting the work this way is that a mobile device could be programmed to automatically do ABE preparation work whenever it is plugged into a power source, and then when it is unplugged, ABE ciphertexts could be rapidly formed with a significant reduction in battery consumption.

Another potential advantage of splitting work this way is that in some applications the online and offline work can be performed in different devices. One might perform the offline work for several encryptions on a high-end server and store these intermediate ciphertexts on a sensor device such that the small device never needs to perform a full encryption. In other applications, for security reasons a designer might wish to limit the number of outward facing servers that have access to the master secret key (or equivalent). Using online/offline techniques he could have several servers performing offline operations, but relatively fewer required for the final online step to generate a user's private key. While a corrupted offline server (without the master secret) could not break the system, in collusion it could produce outputs that would allow an eventual key holder to do so. Therefore, application of this idea would require further analysis and techniques to mitigate this scenario.

Background on Online/Offline Cryptography Even, Goldreich and Micali [9] initiated online/offline techniques for signatures and Shamir and Tauman [22] introduced a general method using chameleon hash functions. In the context of signatures, one would like to perform most of the work for signing a message in the offline phase, but without knowing what the message to be signed is. Later in the online phase the signer will learn the message and given the offline work should be able to sign it relatively quickly.

The focus of our investigation is on moving encryption computation offline. In the basic encryption setting, the job is to perform most of the work for encryption offline, before the message is known. This is one of the reasons that stream ciphers, such as RC4, are sometimes preferred over certain block ciphers, because they operate by generating a pseudorandom string (which can be done offline) and then XORing it with the plaintext (in the online phase).

Let's next consider the task of moving encryption computation offline for Identity-Based Encryption (IBE), where neither the message nor the recipient's identity is known during the offline phase. Guo et al. [12] give an offline encryption system for Identity-Based Encryption (and other works [17, 16, 8, 21] proposed different variants). We illustrate the main idea as a KEM[1] variant of the Boneh-Boyen [5] IBE system. In the offline phase, one will create a ciphertext by encrypting to a random identity $x \in \mathbb{Z}_p$ with randomness $s \in \mathbb{Z}_p$. The resulting BB-type ciphertext will have the form $C_1 = g^s, C_2 = (u^x h)^s$ and the encapsulated key will be $e(g,g)^{\alpha s}$, where the bilinear group description $\mathbb{G}$ of order p and $g, u, h, e(g,g)^{\alpha}$ are in the public parameters. The offline algorithm will store these ciphertext components as well as remember x and s; these together will consist of what we call an *intermediate ciphertext*. In the online phase, the encryptor will learn that she wishes to encrypt to a certain identity $\mathcal{I} \in \mathbb{Z}_p$. To do this, she simply adds a small "correction factor" $r \cdot (\mathcal{I} - x) \in \mathbb{Z}_p$ to the ciphertext components C_1, C_2. The computation only takes one multiplication and subtraction in $\mathbb{Z}_p$. A modified decryption algorithm with the correct private key can then extract the required symmetric key. We note that treating the system as a Key Encapsulation Mechanism allows us to separate the issues of learning the identity in the online phase versus learning the message in the online phase.

The Challenge for ABE From the above description, one can see that the correction techniques critically rely on there being well-known algebraic relationships between the Boneh-Boyen hashes of different identities. Unfortunately, these do not exist in most initial ABE systems [10, 6, 24] as an attribute for string x would typically be represented as either a random group element h_x in the parameters or as the result of a (random oracle modeled) hash function $H(x)$. A second challenge is that the size and structure of ciphertext descriptors is more complex in ABE systems. For instance, in a KP-ABE system the number of attributes associated with a ciphertext may vary widely between each encryption. If one encrypts to a small number in each offline stage, the intermediate ciphertext may be not useable. If one encrypts to a large or maximum number in each offline phase, it can result in much wasted work. Using offline computation efficiently becomes a challenge in this setting. For ciphertext-policy ABE, finding a good solution is more challenging as the "unknown" is an complex access structure.

Our Contributions We develop new techniques for online/offline ABE encryption and key generation that tackle these challenges. The first non-trivial task is to identity ABE constructions that have the required algebraic structure to enable online/offline computation. Unfortunately, most existing schemes do not. However, a few do. We first identified the recent "large universe" construction of Lewko and Waters [14] as a candidate base scheme due to its algebraic structure

[1] A key encapsulation mechanism, where the public key ciphertext encapsulates a symmetric key which could later be used to symmetrically encrypt the plaintext.

that appears amenable to adding correction factors.[2] We finally decided to use a recent more efficient prime-order variant due to Rouselakis and Waters [19]. (We are not aware of any other ABE schemes that can support a similarly efficient online/offline tradeoff.)

We begin by designing online/offline encryption algorithms for Key-Policy ABE. For our first construction we assume a set number of attributes that will be associated with each ciphertext. In this setting we develop a correction technique for the KP-ABE [19] system. We prove security by directly reducing to the security of [19]. This has the advantage of simplicity in that we do not need to revisit the guts of the prior proof. In addition, we will automatically inherenit any future improvements in the proof for the underlying scheme.

For reasons, discussed above assuming a fixed number of attributes per ciphertext is undesirable. To this end we come up with a method of "pooling" work done offline. In this system an encryptor will continuously create offline ciphertext pieces and add these to a pool. When the encryption algorithm later needs to encrypt to a set S of attributes, it grabs $|S|$ pieces from the pool connecting each one to a single attribute from S. The work per attribute is dominated by one multiplication in $\mathbb{Z}_p$. We describe this as a "connect and correct" approach.

We extend our offline encryption approach to the more complex case of Ciphertext-Policy ABE. The challenge here is that a CP-ABE ciphertext is associated with a Linear Secret Sharing Scheme (LSSS) matrix. Again, we develop a pooling technique. However, in this application for each row of the matrix M given online, we will need to correct each ciphertext component to an LSSS share in the exponent and to the corresponding attribute. Finally, we show how online/offline key generation can be derived from our encryption techniques. We observe a symmetry between CP-ABE encryption and KP-ABE key generation that allows us to develop an online/offline pair of algorithms for the latter.

Combining with Outsourcing for ABE We make a brief detour here to discuss how the results of this work might be combined with prior ABE results to make a practical overall system.

In 2011, Green, Hohenberger and Waters [11] presented a solution for outsourcing the decryption of ABE ciphertexts. That is, they assumed that ABE ciphertexts might be stored in the cloud. They then showed how a user can provide the cloud with a *single* translation key that allows the cloud to translate *any* ABE ciphertext satisfied by that user's attributes into a very short El Gamal-style ciphertext, without the cloud being able to read any part of the user's messages. These transmitted ciphertexts are short (saving on bandwidth and receiving time), but also quick to decrypt (with roughly one or two exponentiations). Thus, the ability to outsource decryption to the cloud allows a mobile device to quickly decrypt an ABE-encrypted message.

[2] Interestingly, [14] aimed for a large universe construction in the standard model and thus our use of the schemes's additional structure is a byproduct of removing the random oracles.

Conversely, the results of this work allow a mobile device to quickly *encrypt* an ABE-encrypted message. These two results could be combined into one system, where a mobile device would be fully ABE operational while drastically reducing the computational costs for both decryption (with the help of the cloud) and encryption (with the help of a preparation phase while the phone charges). We believe that creative solutions of this sort can be implemented transparently, but will provide noticeably better performance for users.

2 Definitions for Online/Offline ABE

We work in the key encapsulation mechanism (KEM) setting, where the attribute-based ciphertext hides a symmetric session key that can then be used to symmetrically encrypt data of arbitrary length. The goal in the online/offline setting is to allow as much precomputation of attribute-based ciphertext as possible *without* knowing the intended access policy (ciphertext-policy) or set of attributes (key-policy). We refer the reader to [13] for a review of access structures, linear secret sharing schemes (LSSS) and related conventions.

Definition 1 (Online/Offline Attribute-Based KEM Specification). *Let S represent a set of attributes and $\mathbb{A}$ an access structure. For generality, we will define (I_{key}, I_{enc}) as the inputs to the extract and online encryption functions respectively. In a KP-ABE scheme $(I_{key}, I_{enc}) := (\mathbb{A}, S)$, while in a CP-ABE scheme, we have $(I_{key}, I_{enc}) := (S, \mathbb{A})$. We define the function f as follows:*

$$f(I_{key}, I_{enc}) := \begin{cases} 1 & \text{if } I_{enc} \in I_{key} \text{ in KP-AB setting} \\ 1 & \text{if } I_{key} \in I_{enc} \text{ in CP-AB setting} \\ 0 & \text{otherwise.} \end{cases}$$

An online/offline KP-AB (resp., CP-AB) key-encapsulation mechanism for access structure space $\mathcal{G}$ is a tuple of the following algorithms:

Setup$(\lambda, U) \to (\mathrm{PK}, \mathrm{MK})$**.** *The setup algorithm takes as input a security parameter λ and a universe description U, which defines the set of allowed attributes in the system. It outputs the public parameters* PK *and the master secret key* MK*.*

Extract$(\mathrm{MK}, I_{key}) \to \mathrm{SK}$**.** *The extract algorithm takes as input the master secret key* MK *and an access structure (resp., set of attributes) I_{key} and outputs a private key* SK *associated with the attributes.*

Offline.Encrypt$(\mathrm{PK}) \to \mathrm{IT}$**.** *The offline encryption algorithm takes as input the public parameters* PK *and outputs an intermediate ciphertext* IT*.*

Online.Encrypt$(\mathrm{PK}, \mathrm{IT}, I_{enc}) \to (\mathsf{key}, \mathrm{CT})$ *The online encryption algorithm takes as input the public parameters* PK*, an intermediate ciphertext* IT *and a set of attributes (resp., access structure) I_{enc} and outputs a session key* key *and a ciphertext* CT*.*

Decrypt$(\mathrm{SK}, \mathrm{CT}) \to \mathsf{key}$**.** *The decryption algorithm takes as input a private key* SK *for* I_{key} *and a ciphertext* CT *associated with* I_{enc} *and decapsulates ciphertext* CT *to recover a session key* key *if* S *satisfies* $\mathbb{A}$ *or the error message* $\perp$ *otherwise.*

For a fixed universe description U *and* $\lambda \in \mathbb{N}$*, the KP-AB correctness property requires that for all* $(\mathrm{PK}, \mathrm{MK}) \in \mathsf{Setup}(\lambda, U)$*, all* $S \subseteq U$*, all* $\mathbb{A} \in \mathcal{G}$*, all* $\mathrm{SK} \in \mathsf{Extract}(\mathrm{MK}, \mathbb{A})$*, if* $(\mathsf{key}, \mathrm{CT}) \in \mathsf{Online.Encrypt}(\mathrm{PK}, \mathsf{Offline.Encrypt}(\mathrm{PK}), S)$ *and if* S *satisfies* $\mathbb{A}$*, then* $\mathsf{Decrypt}(\mathrm{SK}, \mathrm{CT})$ *outputs* key*. CP-AB correctness is defined analogously, with the last inputs to* Extract *and* Online.Encrypt *reversed.*

Security Model for Online/Offline AB-KEM Let $\Pi = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Offline.Encrypt}, \mathsf{Online.Encrypt}, \mathsf{Decrypt})$ be an AB-KEM for access structure space $\mathcal{G}$, and consider the following experiment for an adversary $\mathcal{A}$, parameter λ and attribute universe U:

The Online/Offline AB-KEM experiment $\mathsf{OO\text{-}ABKEM\text{-}Exp}_{\mathcal{A},\Pi}(\lambda, U)$:

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 counter $j = 0$. Proceeding adaptively, the adversary can repeatedly make any of the following queries:

- Create(I_{key}): The challenger sets $j := j + 1$. It runs the key generation algorithm on I_{key} to obtain the private key SK and stores in table T the entry $(j, I_{key}, \mathrm{SK})$.
  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 $(i, I_{key}, \mathrm{SK})$ 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 $(i, I_{key}, \mathrm{SK})$ 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 gives a challenge value I^*_{enc} such that for all $I_{key} \in D$, $f(I_{key}, I^*_{enc}) \neq 1$. The challenger runs the algorithm $\mathsf{Online.Encrypt}(\mathrm{PK}, \mathsf{Offline.Encrypt}(\mathrm{PK}), I^*_{enc})$ to obtain $(\mathsf{key}^*, \mathrm{CT}^*)$. It then randomly selects a bit b. If $b = 0$, it returns $(\mathsf{key}^*, \mathrm{CT}^*)$ to the adversary. If $b = 1$, it selects a random session key R in the session key space and returns (R, CT^*).

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 decryption query on the challenge ciphertext CT^*.

Guess. The adversary outputs a guess b' of b. The output of the experiment is 1 if and only if $b = b'$.

Definition 2 (Online/Offline AB-KEM Security). *An online/offline AB-KEM Π is CCA-secure (or secure against* chosen-ciphertext attacks*) for attribute universe U if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists a negligible function negl such that:*

$$\Pr[\mathsf{OO\text{-}ABKEM\text{-}Exp}_{\mathcal{A},\Pi}(\lambda, U) = 1] \leq \frac{1}{2} + negl(\lambda).$$

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 system is *selectively* secure if we add an Init stage before Start where the adversary outputs the challenge I^*_{enc} (instead of waiting until Challenge).

3 A KP-ABE Scheme with Online/Offline Encryption

We now show how to extend the unbounded KP-ABE scheme of Rouselakis and Waters [19, Appendix C] to be an online/offline system. We will work in a key encapsulation mechanism (KEM) model as specified in Defintion 2, so that we can focus on preparing for an unknown attribute set. Any plaintext can be encrypted in a hybrid manner during the online phase by a symmetric cipher keyed with the encapsulated key. We first show a simple system that assumes a bound P on the maximum number of attributes that can be used to encrypt a ciphertext. We show how to remove this bound in Section 3.2.

Setup(λ, U) The setup algorithm takes in a security parameter λ and a universe U of attributes. chooses a bilinear group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. It also chooses random generators $g, h, u, w \in \mathbb{G}$ and picks a random exponent $\alpha \in \mathbb{Z}_p$. It then sets the keys as:

$$\text{PK} = (\mathbb{G}, p, g, h, u, w, e(g,g)^\alpha), \qquad \text{MSK} = (\text{PK}, \alpha).$$

We assume that the universe of attributes can be encoded as elements in $\mathbb{Z}_p$.

Extract$(\text{MSK}, (M, \rho))$ The extract algorithm takes as input the master secret key MSK and an LSSS access structure (M, ρ). Let M be an $\ell \times n$ matrix. The function ρ associates rows of M to attributes. The algorithm initially chooses random values $y_2, \ldots, y_n \in \mathbb{Z}_p$. It then computes ℓ shares of the master secret key as $(\lambda_1, \lambda_2, \ldots, \lambda_\ell) := M \cdot (\alpha, y_2, \ldots, y_n)^T$ (where T denotes the transpose). It then picks ℓ random exponents $t_1, t_2, \ldots, t_\ell \in \mathbb{Z}_p$. For $i = 1$ to ℓ, it computes

$$K_{i,0} := g^{\lambda_i} w^{t_i} \; K_{i,1} := \left(u^{\rho(i)} h\right)^{-t_i} \; K_{i,2} := g^{t_i}.$$

The private key is $\text{SK} := ((M, \rho), \{K_{i,0}, K_{i,1}, K_{i,2}\}_{i \in [1,\ell]})$.

Offline.Encrypt(PK) The offline encryption algorithm takes in the public parameters only. Here we describe the basic system which assumes a maximum bound of P attributes will be associated with any ciphertext. We describe more advanced variations in Section 3.2. The algorithm first picks a random $s \in \mathbb{Z}_p$ and computes

$$\mathsf{key} := e(g,g)^{\alpha s} \; C_0 := g^s.$$

Next, for $j = 1$ to P, it chooses random $r_j, x_j \in \mathbb{Z}_p$ and computes

$$C_{j,1} := g^{r_j} \; C_{j,2} := (u^{x_j} h)^{r_j} w^{-s}.$$

One can view this as encrypting for a random attribute x_j, where this will be corrected in the online phase. The work done in the offline phase is roughly equivalent to the work of the regular encryption algorithm in [19, Appendix C].

The intermediate ciphertext is $\mathrm{IT} := (\mathsf{key}, C_0, \{r_j, x_j, C_{j,1}, C_{j,2}\}_{j \in [1,P]})$.

Online.Encrypt(PK, IT, S) The online encryption KEM algorithm takes as input the public parameters, an intermediate ciphertext IT, and a set of attributes $S = (A_1, A_2, \ldots, A_{k \leq P})$. For $j = 1$ to k, it computes $C_{j,3} := (r_j \cdot (A_j - x_j)) \mod p$. Intuitively, this will correct to the proper attributes. It sets the ciphertext:

$$\mathrm{CT} := (S, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1,k]}).$$

The encapsulated key is key. The dominant cost is one multiplication in $\mathbb{Z}_p$ per attribute in S.

Decrypt(SK, CT) The decryption algorithm in the KEM setting recovers the encapsulated key. It takes as input a ciphertext $\mathrm{CT} = (S, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1,k]})$ for attribute set S and a private key $\mathrm{SK} = ((M, \rho), \{K_{i,0}, K_{i,1}, K_{i,2}\}_{i \in [1,\ell]})$ for access structure (M, ρ). If S does not satisfy this access structure, then the algorithm issues an error message. Otherwise, it sets $I := \{i : \rho(i) \in S\}$ and computes constants $w_i \in \mathbb{Z}_p$ such that $\sum_{i \in I} w_i \cdot M_i = (1, 0, \ldots, 0)$, where M_i is the i-th row of the matrix M. Then it then recovers the encapsulated key by calculating $\mathsf{key} :=$

$$\prod_{i \in I} \left(e(C_0, K_{i,0}) \cdot e(C_{j,1}, K_{i,1}) \cdot e(C_{j,2} \cdot u^{C_{j,3}}, K_{i,2}) \right)^{w_i} = e(g,g)^{\alpha s} \qquad (1)$$

where j is the index of the attribute $\rho(i)$ in S (it depends on i). This does not increase the number of pairing operations over [19, Appendix C], although it adds $|I|$ exponentiations.

Correctness If the attribute set S of the ciphertext is authorized, we have that $\sum_{i \in I} w_i \lambda_i = \alpha$. Therefore, key:

$$
\begin{aligned}
&:= \prod_{i \in I} \left(e(C_0, K_{i,0}) \cdot e(C_{j,1}, K_{i,1}) \cdot e(C_{j,2} \cdot u^{C_{j,3}}, K_{i,2}) \right)^{w_i} \\
&= \prod_{i \in I} (e(g^s, g^{\lambda_i} w^{t_i}) \cdot e(g^{r_j}, (u^{\rho(i)} h)^{-t_i}) \cdot e((u^{x_j} h)^{r_j} w^{-s} \cdot u^{r_j(\rho(i) - x_j)}, g^{t_i}))^{w_i} \\
&= \prod_{i \in I} (e(g,g)^{s \lambda_i} \cdot e(g,w)^{s t_i} \cdot e(g,u)^{-r_j t_i \rho(i)} \cdot \\
&\qquad\qquad e(g,h)^{-r_j t_i} \cdot e(g,u)^{\rho(i) r_j t_i} \cdot e(g,h)^{r_j t_i} \cdot e(g,w)^{-s t_i})^{w_i} \\
&= \prod_{i \in I} e(g,g)^{s w_i \lambda_i} = e(g,g)^{s \alpha}.
\end{aligned}
$$

Recall that in the symmetric setting $e(g,u) = e(u,g)$, for all $g, u \in \mathbb{G}$, although this scheme can operate in an asymmetric setting with small alterations.

3.1 Proof of Selective Security

Discussion on Security. We shortly show that the security of our online/offline system can be directly based on the security of the underlying Rouselakis-Waters [19, Appendix C] system. The Rouselakis-Waters system that we reduce security to is selectively secure based on a "q-type" assumption in prime order groups. We remark that our techniques appear to be equally ammenable to transforming the Lewko-Waters [15] system to an online/offiline system. The Lewko-Waters system is proven selectively secure from a static assumption in composite order groups. If such a transformation were done (as well as a reduction to their scheme), the new scheme would inherit those assumptions.

In [10, Section 9], Goyal et al. discuss how to combine delegation in their ABE systems with the techniques of Canneti-Halevi-Katz [7] to build a CCA secure ABE scheme from a CPA one. We believe that a similar delegation structure exists in our schemes, so that similar techniques would likely work out (although we do not work out the details here).

Theorem 1. *The above online/offline KP-AB-KEM scheme is selectively CPA-secure with respect to Definition 2 assuming that the scheme of Rouselakis and Waters [19, Appendix C] is a selectively CPA-secure KP-ABE system.*

Proof. To prove the theorem, we will show that any PPT attacker $\mathcal{A}$ with a non-negligible advantage in the OO-ABKEM-Exp experiment against the above scheme, which we will denote $\Pi_{OO} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Offline.Encrypt}, \mathsf{Online.Encrypt}, \mathsf{Decrypt})$, can be used to break the selective CPA-security of the Rouselakis-Waters scheme, which we will denote $\Pi_{RW} = (\mathsf{Setup}_{RW}, \mathsf{Extract}_{RW}, \mathsf{Encrypt}_{RW}, \mathsf{Decrypt}_{RW})$, with a PPT simulator $\mathcal{B}$.

The simulator plays the challenger and interacts with $\mathcal{A}$ in OO-ABKEM-Exp with security parameter λ and the universe of attributes set to $U = \mathbb{Z}_p$.

Initialization Initially, $\mathcal{B}$ receives an attribute set $S^* = \{A_1^*, A_2^*, \ldots, A_k^*\} \subseteq U$ from $\mathcal{A}$ and gives it to the RW challenger.

Setup Next, $\mathcal{B}$ receives the public parameters $\text{PK} = (\mathbb{G}, p, g, h, u, w, e(g,g)^\alpha)$ from the RW challenger and passes them to $\mathcal{A}$ unchanged.

Phase 1 The secret keys are the same in both schemes, so any key generation request from $\mathcal{A}$ is passed to the RW challenger to obtain the key.

Challenge $\mathcal{B}$ chooses two distinct, random messages m_0, m_1 in the RW message space and sends them to its RW challenger, and receives back a challenge ciphertext $\text{CT}^*_{RW} = (S^*, C, C_0, \{C_{j,1}, C_{j,2}\}_{j \in [1,|S^*|]})$. Here C is the encrypted message times $e(g,g)^{\alpha s}$, $C_0 = g^s$ and for each attribute $A_j \in S^*$, we have $C_{j,1} = g^{r_j}$ and $C_{j,2} = (u^{A_j} h)^{r_j} w^{-s}$.

It then selects random values $z_1, \ldots, z_{|S|} \in \mathbb{Z}_p$ and computes the ciphertext CT^*_{OO} as (S^*, C_0) followed by

$$C^*_{j,1} := C_{j,1} = g^{r_j} \; C^*_{j,2} := C_{j,2} \cdot u^{-z_j} = (u^{A_j} h)^{r_j} w^{-s} u^{-z_j} \; C^*_{j,3} := z_j.$$

To see why this is a correctly formed ciphertext, one needs to recall the third pairing of equation 1, where one must compute $e(C^*_{j,2} \cdot u^{C^*_{j,3}}, K_{i,2})$, as well as observe that the ciphertext is randomized to have the proper distribution. The z_j blinding will cancel out in this step. Next, $\mathcal{B}$ guess which message was encrypted $\tau_{\mathcal{B}} \in \{0,1\}$ and computes $\mathsf{key}_{guess} := C/m_{\tau_{\mathcal{B}}}$. Finally, $\mathcal{B}$ then sends to $\mathcal{A}$ the tuple $(\mathsf{key}_{guess}, \text{CT}^*_{OO})$.

Phase 2 $\mathcal{B}$ proceeds as in Phase 1.

Guess Eventually, $\mathcal{A}$ outputs a bit $\tau_{\mathcal{A}}$. If $\tau_{\mathcal{A}} = 0$ (meaning that $\mathcal{A}$ guesses that key_{guess} is the key encapsulated by CT^*_{OO}), then $\mathcal{B}$ outputs $\tau_{\mathcal{B}}$. If $\tau_{\mathcal{A}} = 1$ (meaning that $\mathcal{A}$ guesses that key_{guess} is a random key), then $\mathcal{B}$ outputs $1 - \tau_{\mathcal{B}}$. The distribution for $\mathcal{A}$ is perfect. Thus, if $\mathcal{A}$ has advantage ϵ in the OO-ABKEM-Exp experiment, then $\mathcal{B}$ breaks the RW KP-ABE system with the same probability.

3.2 A More Advanced System: Pooling Attributes for an Unbounded System

Previously, we presented a system that imposed a bound of P attributes associated with any ciphertext. We presented P as if it was a system-wide bound for all ciphertexts, for simplicity. A slightly less naive solution would involve creating a set of intermediate ciphertexts prepared for different sizes of attribute sets, and then pulling the "right-sized IT" off-the-shelf during the online phase (e.g., create one IT for a set of size 1, another for a set of size 2, etc.). However, these approaches could prove wasteful, as certain ITs may be created and stored without being used.

Pooling Construction. Instead, we introduce the idea of "pooling" to eliminate waste during the offline phase. The intermediate ciphertext is now comprised of two logical types of objects: a main module and an attribute module. During the offline phase(s), an arbitrary number of main and attribute modules are independently created. During the online phase for attribute set S, one main module and $|S|$ attribute modules will be consumed. The critical feature of this approach is that any attribute module can be attached to any main module. The online phase uses exactly what it needs, and any modules left in the pool can be used on subsequent ciphertexts.

Specifically, during Offline.Encrypt, a main module is computed as follows. It picks a random $s \in \mathbb{Z}_p$ and sets $\mathrm{IT}_{main} := (\mathsf{key}, C_0, C_w)$, where these values are computed as

$$\mathsf{key} := e(g,g)^{\alpha s}\ C_0 := g^s\ C_w := w^{-s}.$$

During Offline.Encrypt, an attribute module is computed as follows. It picks a random $r, x \in \mathbb{Z}_p$ and sets $\mathrm{IT}_{att} := (r, x, C'_1, C'_2)$, where these values are computed as

$$C'_1 := g^r\ C'_2 := (u^x h)^r.$$

During Online.Encrypt for an attribute set S, the algorithm selects any one main module $\mathrm{IT}_{main} := (\mathsf{key}, C_0, C_w)$ and any $|S|$ attribute modules $\mathrm{IT}_{att,j} := (r_j, x_j, C'_{j,1}, C'_{j,2})$ available in the pool. Finally, it computes CT as $(S, C_0, \{C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1,|S|]})$, where

$$C_{j,1} := C'_{j,1} = g^{r_j}\ C_{j,2} := C'_{j,2} \cdot C_w = (u^{x_j} h)^{r_j} \cdot w^{-s}\ C_{j,3} := r_j \cdot (A_j - x_j).$$

The encapsulated key is key.

Security Discussion. The dominant cost in the online encryption algorithm is 2 modular multiplications per attribute in S. To formally capture the pooling model, the specification and security definition in Section 2 would need to be expanded to have the Offline.Encrypt algorithm keep state (e.g., the pool) between iterations and to pass this state into Online.Encrypt as well. Since pooling does not impact the structure or distribution of the final ciphertexts over Section 3 and the adversary in the security experiment only views final ciphertexts, it is relatively straightforward to prove the selective security of the pooling scheme.

4 A CP-ABE Scheme with Online/Offline Encryption

We now turn our attention to developing online/offline CP-ABE systems. This is intuitively harder than KP-ABE, because the structure of ciphertext is more complex. We must now be able to create an intermediate ciphertext in the offline phase that can be quickly be translated to a ciphertext for a hitherto unknown access structure. To do this, we will use and extend the basic "correction" and

pooling concepts introduced for KP-ABE. Our online/offline system is based on the unbounded CP-ABE scheme of Rouselakis and Waters [19, Section 4], where again it takes a special algebraic structure to make this work, which most other CP-ABE systems do not appear to have. As before, we are working in the KEM model. We'll first show a simple system that assumes a bound P on the maximum number of rows in an LSSS access structure that will be used to encrypt. We will subsequently discuss how to remove this bound.

Setup(λ, U) The setup algorithm chooses a bilinear group $\mathbb{G}$ of prime order $p \in \Theta(2^\lambda)$. It also chooses random generators $g, h, u, v, w \in \mathbb{G}$ and picks a random exponent $\alpha \in \mathbb{Z}_p$. It then sets the keys as:

$$\text{PK} = (\mathbb{G}, p, g, h, u, v, w, e(g,g)^\alpha), \qquad \text{MSK} = (\text{PK}, \alpha).$$

Again, we will view the attribute universe as consisting of elements in $\mathbb{Z}_p$.

Extract(MSK, S) The extract algorithm takes as input the master secret key MSK and an attribute set $S = \{A_1, A_2, \ldots, A_k\} \subseteq \mathbb{Z}_p$. The algorithm chooses random values $r, r_1, r_2, \ldots, r_k \in \mathbb{Z}_p$. It then computes $K_0 := g^\alpha w^r, K_1 := g^r$, and for $i = 1$ to k, it computes

$$K_{i,2} := g^{r_i} \; K_{i,3} := \left(u^{A_i} h\right)^{r_i} v^{-r}.$$

The private key is $\text{SK} := (S, K_0, K_1, \{K_{i,2}, K_{i,3}\}_{i \in [1,k]})$.

Offline.Encrypt(PK) The offline encryption algorithm takes in the public parameters only. Here we describe the basic system which assumes a maximum bound of P rows in any LSSS access structure used in a ciphertext. We describe more advanced variations in Section 4.1. The algorithm first picks a random $s \in \mathbb{Z}_p$ and computes

$$\mathsf{key} := e(g,g)^{\alpha s} \; C_0 := g^s.$$

Next, for $j = 1$ to P, it chooses random $\lambda'_j, x_j, t_j \in \mathbb{Z}_p$ and computes

$$C_{j,1} := w^{\lambda'_j} v^{t_j} \; C_{j,2} := (u^{x_j} h)^{-t_j} \; C_{j,3} := g^{t_j}.$$

One can view this as encrypting for a random attribute x_j with a random "share" λ'_j of s, where this will be corrected in the online phase. We remark that the work done in the offline phase is roughly equivalent to the work of the regular encryption algorithm in [19, Section 4].

Intermediate ciphertext is $\text{IT} := (\mathsf{key}, s, C_0, \{\lambda'_j, t_j, x_j, C_{j,1}, C_{j,2}, C_{j,3}\}_{j \in [1,P]})$.

Online.Encrypt$(\text{PK}, \text{IT}, (M, \rho))$ The online encryption KEM algorithm takes as input the public parameters, an intermediate ciphertext IT, and an LSSS access structure (M, ρ), where M is an $\ell \times n$ matrix and $\ell \leq P$. It picks random $y_2, \ldots, y_n \in \mathbb{Z}_p$, sets the vector $\boldsymbol{y} = (s, y_2, \ldots, y_n)^T$ (where T denotes the transpose of the matrix) and computes a vector of shares of s as $(\lambda_1, \ldots, \lambda_\ell)^T = M\boldsymbol{y}$.

For $j = 1$ to ℓ, it computes

$$C_{j,4} := \lambda_j - \lambda'_j \; C_{j,5} := t_j \cdot (\rho(j) - x_j).$$

Intuitively, this will correct to the proper attributes and shares of s. It sets the ciphertext as:

$$\mathrm{CT} := ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j \in [1,k]}).$$

The encapsulated key is key. The dominant cost is one multiplication in $\mathbb{Z}_p$ per row of M.

Decrypt$(\mathrm{SK}, \mathrm{CT})$ The decryption algorithm in the KEM setting recovers the encapsulated key. It takes as input a ciphertext $\mathrm{CT} = ((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j \in [1,k]})$ for access structure (M, ρ) and a private key $\mathrm{SK} = (S, \{K_{i,0}, K_{i,1}, K_{i,2}\}_{i \in [1,\ell]})$ for access structure (M, ρ). If S does not satisfy this access structure, then the algorithm issues an error message. Otherwise, it sets $I := \{i : \rho(i) \in S\}$ and computes constants $w_i \in \mathbb{Z}_p$ such that $\sum_{i \in I} w_i \cdot M_i = (1, 0, \ldots, 0)$, where M_i is the i-th row of the matrix M. Then it then recovers the encapsulated key by calculating $\mathsf{key} := e(g, g)^{\alpha s} =$

$$\frac{e(C_0, K_0)}{e(w^{\sum_{i \in I} C_{i,4} w_i}, K_1) \cdot \prod_{i \in I} (e(C_{i,1}, K_1)} \cdot \frac{1}{e(C_{i,2} \cdot u^{C_{i,5}}, K_{j,2}) \cdot e(C_{i,3}, K_{j,3}))^{w_i}} \quad (2)$$

where j is the index of the attribute $\rho(i)$ in S (it depends on i). We note that this decryption algorithm adds one pairing operation and $|I|+1$ exponentiations over [19, Appendix C]. Alternatively, one could re-arrange the equation for no additional pairings at the cost of $2|I|$ exponentiations.

In the full version [13], we show correctness and prove the below theorem.

Theorem 2. *The above online/offline CP-AB-KEM scheme is selectively CPA-secure with respect to Definition 2 assuming that the scheme of Rouselakis and Waters [19, Section 4] is a selectively CPA-secure CP-ABE system.*

4.1 Pooling Attributes for an Unbounded Ciphertext-Policy System

In the previous section, we presented an online/offline system that imposed a bound of P rows on any LSSS access matrix associated with any ciphertext. As introduced in Section 3.2, we now show how to remove this bound by creating a "pool" from which to draw ready-made ciphertext components. As before, the intermediate ciphertext is comprised of two logical types of objects: a main module and an attribute module. During the offline phase(s), an arbitrary number of main and attribute modules are independently created. During the online phase for LSSS access structure (M, ρ), one main module and ℓ attribute modules will be consumed, where M is an $\ell \times n$ matrix. Any attribute module can be attached to any main module.

Specifically, during Offline.Encrypt, a main module is computed as follows. It picks a random $s \in \mathbb{Z}_p$ and sets $\mathrm{IT}_{main} := (\mathsf{key}, C_0)$, where these values are computed as

$$\mathsf{key} := e(g,g)^{\alpha s}\ C_0 := g^s.$$

During Offline.Encrypt, an attribute module is computed as follows. It picks a random $\lambda, x, t \in \mathbb{Z}_p$ and sets $\mathrm{IT}_{att} := (\lambda, x, t, C_1, C_2, C_3)$, where these values are computed as

$$C_1 := w^{\lambda} v^t\ C_2 := (u^x h)^t\ C_3 := g^t.$$

During Online.Encrypt for an LSSS access structure (M, ρ), where M is an $\ell \times n$ matrix, the algorithm selects any one main module $\mathrm{IT}_{main} := (\mathsf{key}, C_0)$ and any ℓ attribute modules $\mathrm{IT}_{att,j} := (\lambda_j, x_j, t_j, C_{j,1}, C_{j,2}, C_{j,3})$ available in the pool. It picks random $y_2, \ldots, y_n \in \mathbb{Z}_p$, sets the vector $\boldsymbol{y} = (s, y_2, \ldots, y_n)^T$ (where T denotes the transpose of the matrix) and computes a vector of shares of s as $(\lambda_1, \ldots, \lambda_\ell)^T = M\boldsymbol{y}$.

Finally, it computes CT as $((M, \rho), C_0, \{C_{j,1}, C_{j,2}, C_{j,3}, C_{j,4}, C_{j,5}\}_{j \in [1,\ell]})$, where

$$C_{j,4} := \lambda_j - \lambda'_j\ C_{j,5} := t_j \cdot (\rho(j) - x_j).$$

The encapsulated key is key. The dominant cost in the online encryption algorithm is one modular multiplication per row in M. The security discussion at the end of Section 3.2 applies here as well.

5 Online/Offline ABE Key Generation

Private key generation in ABE systems requires the master secret key MSK. This key is so valuable that any organization granting keys might do well to store it on only a small number of well-guarded servers. At the same time, this could create a bottleneck in systems with many users, especially when private keys are reissued each time period for revocation purposes. In this section, we discuss how the key generation operation in the KP-ABE system of Section 3 and the CP-ABE system of Section 4 can operate in an online/offline fashion as well. Thus, the bulk of the key generation work can be performed by servers that are truly *offline* (or otherwise well secured). These pre-computations can be passed to the online servers, where incoming requests can be processed quickly.

In the KP-ABE setting, a private key embeds an LSSS access structure, whereas in the CP-ABE setting, the private key embeds a set of attributes. We will borrow ideas from the prior two sections to deal with these objects, where again we can employ both the "correct and connect" and "pooling" concepts.

To capture online/offline key generation, one needs to replace the Extract algorithm with an offline algorithm that takes in the MK and produces a intermediate private key (or pool of private key parts) and an online algorithm that takes in this intermediate key (or pool) together with an access structure and then produces the private key. The security experiment is essentially unchanged except that the Create oracle (called in Phases 1 and 2) now calls Offline.Extract and Online.Extract in sequence to create a private key.

5.1 Online/Offline Key Generation for KP-ABE Keys

The Setup and encryption algorithms remain the same as Section 3. We present a pooling solution, and because the structure of the private keys change, so must the decryption algorithm.

Offline.Extract(MSK) There are no "main" key modules. A "row" module is computed by selecting random $\lambda', x, t \in \mathbb{Z}_p$ and outputting $I_{row} := (\lambda', x, t, K_0, K_1, K_2)$ where $K_0 := g^{\lambda'} w^t$, $K_1 := (u^x h)^{-t}$ and $K_2 := g^t$.

Online.Extract(*pool*, (M, ρ)) Let M be an $\ell \times n$ matrix. The algorithm initially chooses random values $y_2, \ldots, y_n \in \mathbb{Z}_p$. It then computes ℓ shares of the master secret key as $(\lambda_1, \lambda_2, \ldots, \lambda_\ell) := M \cdot (\alpha, y_2, \ldots, y_n)$. Next select any ℓ row modules from the pool. For $i = 1$ to ℓ, set $K_{i,3} := \lambda_i - \lambda_i'$ and $K_{i,4} := t_i \cdot (\rho(i) - x_i)$. The private key is $\mathrm{SK} := ((M, \rho), \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}, K_{i,4}\}_{i \in [1,\ell]})$. The dominant cost is one multiplication per row of M.

Decrypt(SK, CT) Using the prior steps and notation, it recovers the encapsulated $\mathsf{key} := \prod_{i \in I} \left(e(C_0, K_{i,0} \cdot g^{K_{i,3}}) \cdot e(C_{j,1}, K_{i,1} \cdot u^{K_{i,4}}) \cdot e(C_{j,2} \cdot u^{C_{j,3}}, K_{i,2}) \right)^{w_i} = e(g,g)^{\alpha s}$. This adds $2|I|$ exponentiations over the construction in Section 3.

5.2 Online/Offline Key Generation for CP-ABE Keys

The CP-ABE system in Section 4 can be extended in a similar manner. In that system, there will be a "main" key module which contains K_0, K_1 and $K_v := v^{-r}$. The attribute modules are identical to those of Section 3.2 and the keys are assembled as in the online phase of 3.2. The decryption equation is then $\mathsf{key} := e(C_0, K_0)/D$, where $D = e(w^{\sum_{i \in I} C_{i,4} w_i}, K_1) \cdot \prod_{i \in I} (e(C_{i,1}, K_1) \cdot e(C_{i,2} \cdot u^{C_{i,5}}, K_{j,2} \cdot u^{K_{j,4}}) \cdot e(C_{i,3}, K_{j,3}))^{w_i}$, resulting in $e(g,g)^{\alpha s}$.

6 Performance Analysis

We provide estimates on the performance of the proposed schemes in Figures 1 and 2. These numbers are extrapolated from operation times on a 256-bit Bareto-Naehrig curve using version 0.3.1 of the RELIC library [3]. Times are measured in milliseconds (averaged over 10,000 iterations) and were computed on an Intel Core i7 processor with 16GB RAM [2]. We ignore small numbers of operations which will be negligible by comparison, such as arithmetic in $\mathbb{Z}_p$.

A natural question to ask is: how much pre-processing can I do for an ABE encryption (similarly, key generation) before I know the message I want to encrypt or the access structure that I want to encrypt under? It may come as a surprise that the results are so drastic. Indeed, our estimates show that the answer to this question is: you can do *almost all* of the encryption work, before you know any of the specifics of what/to whom you are encrypting.

Indeed, our *worst*-case for encryption was key-policy ABE in pooling mode, and even then over 99% of the work could be done offline. Similarly, the worst-case for key generation was ciphertext-policy ABE in pooling mode, and even

Encryption Algorithm	Bilinear Operations	Est. Time $P = 10$	Est. Time $P = 100$
KP-ABE from [19, App. C]	$1\mathbb{E}_T + (3P+2)\mathbb{E}_1 + 2P\mathbb{M}_1$	.133	1.134
KP-Offline Sec. 3	$1\mathbb{E}_T + (3P+2)\mathbb{E}_1 + 2P\mathbb{M}_1$	.133	1.134
KP-Online Sec. 3	0	$< .001$	$< .001$
KP-Pool-Offline Sec. 3.2	$1\mathbb{E}_T + (3P+2)\mathbb{E}_1 + P\mathbb{M}_1$	.133	1.132
KP-Pool-Online Sec. 3.2	$P\mathbb{M}_1$	$< .001$	.001
CP-ABE from [19]	$1\mathbb{E}_T + (5P+1)\mathbb{E}_1 + 2P\mathbb{M}_1$	.203	1.870
CP-Offline Sec. 4	$1\mathbb{E}_T + (5P+1)\mathbb{E}_1 + 2P\mathbb{M}_1$	.203	1.870
CP-Online Sec. 4	0	$< .001$	$< .001$
CP-Pool-Offline Sec. 4.1	$1\mathbb{E}_T + (5P+1)\mathbb{E}_1 + 2P\mathbb{M}_1$	.203	1.870
CP-Pool-Online Sec. 4.1	0	$< .001$	.001

Fig. 1. Performance estimates for regular and online/offline encryption algorithms. We mapped these algorithms into the asymmetric bilinear setting, placing the ciphertexts in $\mathbb{G}_1$ and keys in $\mathbb{G}_2$. Let $\mathbb{E}_i$ (resp., $\mathbb{M}_i$) denote an exponentiation (reps., multiplication) in the group $\mathbb{G}_i$. The bilinear operations are the dominate cost, so we ignore minor factors such as arithmetic in $\mathbb{Z}_p$. The variable P represents the size of the attribute list (in KP-ABE) or the complexity of the access policy (in CP-ABE). The times are in seconds. It is helpful to compare the cost of the original scheme (with a citation) to the cost of the online phase of the given algorithms. In three of the four schemes presented, all bilinear group operations for encryption can be shifted to the offline phase.

then over 99% of the work could be done offline. It is also worth noting that the total computation required between the offline and online phases is nearly identical to the work required by the original scheme. Thus, the total work remains the same, but the vast majority of it can be shifted in time to a moment when the device is least busy or has access to a power source.

We remark that the operation counts given here for the schemes in [19] differ slightly from the summary given in that work. The counts from [19] were obtained from the Charm [1] benchmarking utility, which may have performed various optimizations, whereas ours are a strict count of operations from the algorithms as presented in the paper [19]. We do not expect these differences to have any significant impact on the estimates in Figures 1 and 2.

7 Conclusions

We are exploring methods to make attribute-based encryption (ABE) more efficient for deployment. To this end, we investigated how devices might quickly encrypt ABE messages or generate user keys, even for complex policies.

We developed new "connect and correct" techniques for ABE that split the computation for encryption and key generation into two phases: a preparation phase that does the vast majority of the work to encrypt a message or create a secret key *before* it knows the message or the attribute list/access control policy that will be used (or even the size of the list or policy). A second phase can then rapidly assemble an ABE ciphertext or key when the specifics become known.

Key Generation Algorithm	Bilinear Operations	Est. Time $P = 10$	Est. Time $P = 100$
KP-ABE from [19, App. C]	$5P\mathbb{E}_2 + 2P\mathbb{M}_2$	.370	3.703
KP-Pool-Offline Sec. 5.1	$5P\mathbb{E}_2 + 2P\mathbb{M}_2$	.370	3.703
KP-Pool-Online Sec. 5.1	0	$< .001$	$< .001$
CP-ABE from [19]	$(3P + 4)\mathbb{E}_2 + (2P + 1)\mathbb{M}_2$	.252	2.253
CP-Pool-Offline Sec. 5.2	$(3P + 4)\mathbb{E}_2 + (P + 1)\mathbb{M}_2$	.251	2.251
CP-Pool-Online Sec. 5.2	$P\mathbb{M}_2$	$< .001$	.003

Fig. 2. Performance estimates for regular and online/offline key generation algorithms. We mapped these algorithms into the asymmetric bilinear setting, placing the ciphertexts in $\mathbb{G}_1$ and keys in $\mathbb{G}_2$. Let $\mathbb{E}_i$ (resp., $\mathbb{M}_i$) denote an exponentiation (reps., multiplication) in the group $\mathbb{G}_i$. The bilinear operations are the dominate cost, so we ignore minor factors such as arithmetic in $\mathbb{Z}_p$. The variable P represents the size of the attribute list (in CP-ABE) or the complexity of the access policy (in KP-ABE). The times are in seconds. It is helpful to compare the cost of the original scheme (with a citation) to the cost of the online phase. In both schemes, our estimates show that over 99% of the work to generate a key can be shifted to the offline phase.

This concept is sometimes called "online/offline" encryption. We provided efficient constructions for both key-policy and ciphertext-policy ABE systems.

We provided performance estimates that showed over 99% of the computational work could be moved to offline phase in many scenarios. We expect that this technology could reduce battery consumption on mobile devices and help reduce the bottleneck on a master authority server tasked with generating user keys. Overall, it helps reduce the cost of bringing ABE into practice.

Acknowledgments

The authors thank Joseph Ayo Akinyele and Matthew Green for advice on performance numbers and other helpful comments. Susan Hohenberger was supported in part by NSF CNS-1154035 and CNS-1228443; the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory under contract FA8750-11-2-0211, DARPA N11AP20006, the Office of Naval Research under contract N00014-11-1-0470, and a Microsoft Faculty Fellowship. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

1. Joseph A. Akinyele, Christina Garman, Ian Miers, Matthew W. Pagano, Michael Rushanan, Matthew Green, and Aviel D. Rubin. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 3(2):111–128, 2013.
2. Joseph Ayo Akinyele and Matthew Green. Personal communication., 2013.

3. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography. `http://code.google.com/p/relic-toolkit/`.
4. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
5. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.
6. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.
7. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.
8. Sherman S. M. Chow, Joseph K. Liu, and Jianying Zhou. Identity-based online/offline key encapsulation and encryption. In *ASIACCS*, pages 52–60, 2011.
9. Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. *J. Cryptology*, 9(1):35–67, 1996.
10. 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.
11. Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of ABE ciphertexts. In *USENIX Security Symposium*, 2011.
12. Fuchun Guo, Yi Mu, and Zhide Chen. Identity-based online/offline encryption. In *Financial Cryptography*, pages 247–261, 2008.
13. Susan Hohenberger and Brent Waters. Online/offline attribute-based encryption, 2014. The full version is available from the IACR ePrint Archive, Report 2014/021.
14. Allison B. Lewko and Brent Waters. Unbounded HIBE and attribute-based encryption. In *EUROCRYPT*, pages 547–567, 2011.
15. Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198, 2012.
16. Joseph K. Liu and Jianying Zhou. An efficient identity-based online/offline encryption scheme. In *ACNS*, pages 156–167, 2009.
17. Zhongren Liu, Li Xu, Zhide Chen, Yi Mu, and Fuchun Guo. Hierarchical identity-based online/offline encryption. In *ICYCS*, pages 2115–2119, 2008.
18. 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.
19. Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 463–474, 2013.
20. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
21. S. Sharmila Deva Selvi, S. Sree Vivek, and C. Pandu Rangan. Identity based online/offline encryption and signcryption schemes revisited. In *InfoSecHiComNet*, pages 111–127, 2011.
22. Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In *CRYPTO*, pages 355–367, 2001.
23. Patrick Traynor, Kevin R. B. Butler, William Enck, and Patrick McDaniel. Realizing massive-scale conditional access systems through attribute-based cryptosystems. In *NDSS*, 2008.
24. Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, pages 53–70, 2011.