5-2024

Make revocation cheaper: Hardware-based revocable attribute-based encryption

Xiaoguo LI

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

Tao XIANG

Shengmin XU

Bowen ZHAO

See next page for additional authors

Published in 2024 IEEE Symposium on Security and Privacy (SP). May 2024, pp. 3109-3127
https://doi.org/10.1109/SP54263.2024.00100

Make Revocation Cheaper: Hardware-Based Revocable Attribute-Based Encryption

Xiaoguo Li[†], Guomin Yang[†], Tao Xiang[‡], Shengmin Xu[§], Bowen Zhao[¶], HweeHwa Pang[†], Robert H. Deng[†]
[†]*Singapore Management University, {xiaoguoli, gmyang, hhpang, robertdeng}@smu.edu.sg*
[‡]*Chongqing University, txiang@cqu.edu.cn*
[§]*Fujian Normal University, smxu1989@gmail.com*
[¶]*Xidian University (Guangzhou Institute of Technology), bwinzhao@gmail.com*

***Abstract*—As an advanced one-to-many public key encryption system, attribute-based encryption (ABE) is widely believed to be a promising technology for achieving flexible and fine-grained access control of encrypted data on untrusted storage servers (e.g., public cloud servers). However, user revocation in ABE is a critical but challenging problem, and designing efficient revocable ABE has been an active research topic in the past decade. Almost all the existing revocable ABE schemes incorporate a timestamp in the encryption algorithm such that revoked users cannot decrypt ciphertexts generated in future time intervals. To prevent revoked users from decrypting past ciphertexts, the storage server needs to perform a process called ciphertext delegation (Sahai et al., CRYPTO'12) that periodically updates the timestamp for all ciphertexts. As the number of ciphertexts could be huge in a storage system, ciphertext delegation could pose a huge computation overhead to the server.**

Motivated by the popularity of commodity Trusted Execution Environment (TEE) technologies, this paper initiates the study on hardware-based revocable ABE (HR-ABE) to eliminate the (unscalable) ciphertext delegation and prevent collusion attacks between an untrusted storage server and revoked users. We formalize this new notion and present an efficient HR-ABE construction that also supports outsourced decryption for resource-constrained data users. Furthermore, HR-ABE is also designed to address the potential secret leakage problem suffered by TEE (e.g., due to side-channel attacks) so that the leakage of secrets possessed by TEE does not lead to leakage of user data. We prove HR-ABE's security formally and benchmark its performance experimentally.

***Index Terms*—Attribute-based Encryption, Trusted Execution Environment, Data Sharing, Revocation**

1. Introduction

Attribute-based encryption (ABE) [14] is a one-to-many public key encryption system that enables flexible and fine-grained access control of encrypted data on an untrusted storage server. There are two types of ABE, ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE) [35]. In CP-ABE, a key generation center (KGC) issues a private key to every data user (DU) based on their user attributes. Each data owner (DO) binds an access policy with a ciphertext such that a DU can decrypt a ciphertext only if the set of attributes associated with the DU's private key satisfy the access policy in the ciphertext. KP-ABE operates in a dual manner where the positions of the policy and the attributes are swapped. Without loss of generality, we focus on CP-ABE, but our results can also be extended to KP-ABE.

Despite many elegant ABE constructions being proposed in the literature [1], [9], [14], [29], [30], efficient user revocation remains as a critical but challenging problem. There exist two revocation mechanisms: direct [24], [34] and indirect [3] revocation. In direct revocation, a DO encrypts a file by explicitly specifying the revocation list (as a separate policy) such that revoked users cannot decrypt the ciphertext. Direct revocation does not affect the non-revoked DUs; however, it requires all DOs to maintain the latest revocation list, which grows over time and affects the ciphertext size. The indirect revocation mechanism, first proposed for identity-based encryption (IBE) [7], has been more popular and extensively studied in the past decade (e.g., [12], [23], [30]). The general idea of indirect revocation is to embed a timestamp in the ciphertext and the KGC updates the private keys of non-revoked DUs periodically such that revoked DUs will not be able to obtain valid key updates to decrypt ciphertexts generated in any future time intervals. Some attractive features of the existing indirect revocation schemes include public broadcasting and logarithmic size (with respect to the number of DUs) key update for each time interval.

Although the indirect revocation approach addresses the user revocation for future ciphertexts, the revoked DUs can still have the power to decrypt past ciphertexts they were originally allowed to. To mitigate this problem, Sahai et al. introduced the concept of *ciphertext delegation* [30], which allows the storage server to periodically update the timestamps in all ciphertexts. However, as the number of ciphertexts in a storage server is huge and all ciphertexts need to be updated at each time interval, the process could pose a huge workload to the server.

In summary, the existing indirect revocation mechanisms for ABE have the following limitations:

1) all non-revoked users periodically update their keys;

TABLE 1: Performance comparison among existing revocable attribute-based encryption schemes

Scheme	Rev. type	Cipher delegation	Revocation cost	Key update size	Key size	Ciphertext size
[3]	Direct	**Not supported**	**Not required**	**Not required**	$O(4\|A\|) \cdot \|\mathbb{G}\|$	$O(R(\log \frac{N}{R} + 1) + \|A\|) \cdot \|\mathbb{G}\|$
[30]	Indirect	$O(\|A\| \log \|\Omega\| \cdot \|D\|)$	$O((N-R) \cdot \log \frac{N}{R}) \cdot \|A\| \cdot C_{\mathcal{E}}$	$O((N-R) \cdot \log \frac{N}{R}) \cdot \|\mathbb{G}\|$	$O(\log N\|A\|) \cdot \|\mathbb{G}\|$	$O(\log T(\|A\| + 1)) \cdot \|\mathbb{G}\|$
[12]	Indirect	$O(\|\Omega\|) \cdot \|D\|$	$O((N-R) \cdot \log \frac{N}{R}) \cdot C_{\mathcal{E}}$	$O((N-R) \cdot \log \frac{N}{R}) \cdot \|\mathbb{G}\|$	$O(\log \frac{N}{R}\|A\|) \cdot \|\mathbb{G}\|$	$O(3\|A\| + 3) \cdot \|\mathbb{G}\|$
HR-ABE	TEE	**Not required**	$O(R \log N) \cdot C_{\mathcal{H}}$	**Not required**	$O(2\|A\|) \cdot \|\mathbb{G}\|$	$O(2\|A\| + 4) \cdot \|\mathbb{G}\|$

N: the number of all users; R: the number of revoked DUs; $|\Omega|$: the universe of attribute; $|A|$: the size of attribute set; T: the maximum number of time periods; $|D|$: the number of all previous ciphertexts; $|\mathbb{G}|$: the length of element in group $\mathbb{G}$; $C_{\mathcal{E}}$: the cost of exponentiation operation; $C_{\mathcal{H}}$: the cost of hash operation.

2) the storage server needs to periodically perform (unscalable) ciphertext delegation to prevent revoked users from decrypting past ciphertexts.

Our question. The Trusted Execution Environment (TEE) (e.g., Intel SGX [11], AMD TrustZone [16]) has become a popular and promising technology to provide confidentiality and integrity guarantee for secure protocol design. This motivates us to investigate the possibility of eliminating drawbacks of the above-mentioned revocation mechanisms using hardware technology. Since the host where a TEE is deployed could be malicious, we elaborate on two common knowledgeable TEE trust models.

- *Completely trusted TEE model [33].* In this model, users believe that the code inside an enclave is robust enough to prevent side channel attacks, so that the TEE guarantees both confidentiality and integrity of the internal state (code and data) of the enclave.
- *Transparent TEE model [19].* This model makes a very minimal trust assumption, namely the adversary who controls the TEE host is able to observe the entire state of an enclave except for the attestation signature key, such that the TEE only guarantees integrity of the internal state (code and data) of the enclave.

Based on the two TEE trust models, this work aims to provide answers to the following two research questions:

(1) Can we use commodity TEE technologies to build secure and practical revocable ABE systems that eliminate the drawbacks of the existing revocation mechanisms?

(2) If the answer to the previous question is yes, what level of security can the hardware-based system achieve under different TEE trust models?

1.1. Our Contributions

We initial the study and provide affirmative answers to the above questions by formalizing the notion of hardware-based revocable ABE (HR-ABE) and proposing a concrete scheme that eliminates ciphertext delegation and key update. Specifically, our main results include two folds.

- Under the completely trusted TEE model, we formulate a security model (sIND-ColA) for HR-ABE, which captures the collusion attacks. We show HR-ABE is provable security, even if collusion attacks happened between the cloud server and revoked DUs.
- Under the transparent TEE model, we also formulate a security model (sIND-CTA) for HR-ABE that captures the **Corrupted TEE Attacks** (CTAs), in which malicious server can learn TEE's internal state, including the user data, due to side-channel attacks [13], [27], [32]. We show that HR-ABE preserves semantic security against the malicious server, even if CTAs happened.

We demonstrate that HR-ABE outperforms the existing pure crypto-based revocation solutions and benchmark HR-ABE's performance. Table 1 presents a comparison between our HR-ABE and the existing revocable ABE schemes. As shown in the table, we stress that HR-ABE does not require ciphertext delegation or key update.

1.2. Technical Overview

The use of TEE motivated us to first consider solutions for revocable attribute-based encryption based on a less powerful cryptographic primitive – proxy re-encryption (PRE [6]), in which a re-encryption key is involved to transform a ciphertext encrypted under a public key into a new ciphertext under a different key. In our case, we let TEE hold the re-encryption key and then only transform ciphertexts for non-revoked DUs. Unfortunately, the above straightforward idea does not work: if we let TEE manage the access policy, then we lose the ability to let data owner (DO) directly specify the access policy during encryption; if we allow DO to specify the access policy, CSP can arbitrarily tamper with the policy set by DO who is not supposed to apply any secret key in the encryption process.

We observe that the outsourced ABE (OABE) [15] can be adapted for the purpose of user revocation. OABE was introduced to let a server (e.g., a cloud service provider, or CSP) perform most of the decryption workloads (also known as outsourced description) for end DUs. On one hand, OABE involves a key pair $(\mathrm{TK}, \mathrm{SK})$, in which the CSP employs TK (transformation key) to cancel the attribute-based access control layer in an ABE ciphertext and the DU keeps an SK (secret key) to recover the message. On the other hand, OABE is carefully designed to achieve replayable chosen-ciphertext attacks (RCCA) security [15] in the outsourced partial decryption setting, which ensures the CSP can only produce a legitimate transformed ciphertext containing the original message to the DU for decryption.

To perform user revocation, a naive solution is to let CSP maintain a DU list and its corresponding transformation keys; CSP then removes a DU and his/her transformation key from the list when the DU is revoked. While this approach does not require ciphertext delegation, **Collusion**

Attacks between CSP and revoked DUs could easily fail the revocation mechanism.

1.2.1. Hardware-based Revocation Mechanism. A natural remedy to address the collusion attack is to perform the ciphertext transformation within a TEE deployed on the untrusted server. It is desirable to retain the main merit of OABE, i.e., reducing the workload for resource-constrained DUs. However, the ciphertext transformation (i.e., partial decryption for canceling the attribute-based access control layer) is the most expensive operation in ABE decryption. Given that the TEE needs to perform the transformation for a large number of ciphertexts from multiple DUs, the solution becomes unscalable because of TEE's resource limits, e.g., the available memory size in SGX is only about 90MB.

Based on such an observation, our basic idea is to split the original ABE decryption key into three parts: a transformation key (TK), a revocation key (RK), and a decryption key (DK). We let TEE hold the RK and now the transformation process consists of two stages. (1) CSP employs TK to transform a ciphertext into a partially transformed ciphertext (PTC); (2) TEE checks that DU is not revoked (i.e., in the DU list) and utilizes RK to transform the PTC into a transformed ciphertext (TC). Finally, TC is sent to the DU for final decryption using DK. We call such a cryptographic building block ABE with two-stage outsourced decryption (ABE-2OD). In ABE-2OD, the most resource-intensive operations are offloaded from the TEE to CSP. Consequently, the TEE only needs to perform lightweight operations.

1.2.2. ABE-2OD Construction. The syntax of ABE-2OD is similar to that of OABE [15], except that it now involves two transformation algorithms, i.e., Transform1 executed by the TK holder (CSP) and Transform2 run by the RK holder (TEE). Nonetheless, we cannot easily extend OABE to construct a secure ABE-2OD, as shown below.

Our first attempt is to split the OABE DU secret key SK = z into two parts $(\mathrm{SK}_1 = \gamma, \mathrm{SK}_2 = \beta)$, where $z = \beta \cdot \gamma$, and then set RK = SK_1 and DK = SK_2. However, the malicious CSP may submit *arbitrary transformation queries* to TEE. We observe that answering arbitrary transformation queries without checking PTC's legitimacy may lead to leakage of the revocation key RK. Unlike the OABE scheme, in which DU can recover the message and check the ciphertext's legitimacy (i.e., via OABE's RCCA security), TEE in such a design holding RK = SK_1 can only produce a TC, which needs to be sent to the DU (hence observable by the attacker) for the final decryption. In other words, TEE cannot recover the message and thus has no way to check whether a given PTC is a legitimate one while being invoked by the CSP to perform Transform2.

To check a PTC's legitimacy, our second attempt is to employ the "*decrypt-then-reencypt*" paradigm in TEE, which is built on OABE and a public key encryption (PKE) scheme. Let (TK, SK) be an OABE key pair and (ek, dk) be a PKE key pair. Then the revocation key RK = (SK, ek) held by the TEE binds the attribute-based key TK with a DU's final decryption key dk. Given an OABE ciphertext, after the first stage transformation done by CSP (with TK), TEE (with RK) recovers the original message using SK, and then re-encrypts the message using ek if the DU is not revoked. Although the above "decrypt-then-reencypt" design allows TEE to perform a sanity check on a PTC, it only works if we have full trust on TEE since the TEE essentially becomes a "super DU" who can decrypt and see all the messages.

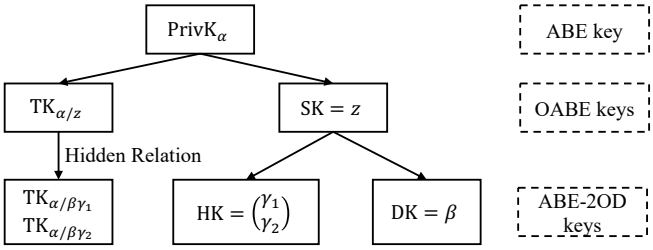


Figure 1: Keys in ABE, OABE, and ABE-2OD. α is the master key. HK is a helper key that corresponds to RK held by the TEE.

However, there could be concerns raised by data owners/users when TEE can see all the messages, as CTA (described before) could happen in reality when TEE is deployed on a malicious host. As a result, CSP may learn TEE's internal state, including the user data. The "decrypt-then-reencrypt" approach loses all security protection under CTA, which is undesirable. Therefore, our goal is to design a secure ABE-2OD that TEE only serves the duty of revocation but does not see the user data (or weaken the security of OABE). Our final design of ABE-2OD circumvents the above-mentioned flaws. Figure 1 shows the high-level idea of our design. In Green's OABE, an ABE private key PrivK$_\alpha$ containing the master key α is split into TK$_{\alpha/z}$ and z. In our ABE-2OD, for checking the legitimacy of PTC, we set DK = β, RK = (γ_1, γ_2) and then hide the information of RK into TK = (TK_1, TK_2), which is related to RK (i.e., TK$_{\alpha/(\beta\gamma_1)}$, TK$_{\alpha/(\beta\gamma_2)}$). With such a method, a valid PTC calculated by CSP includes two components C_{P1} and C_{P2} such that $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$. If the hidden relation holds, TEE cancels out the term $1/\gamma_1$ in C_{P1}, which is then forwarded to DU for decryption using β.

We give an intuitive analysis for the mechanics underlying the design that can achieve our objectives. (1) Each DU has a unique pair (γ_1, γ_2), which prevents PTCs calculated from inconsistent TKs from bypassing the legitimacy checking mechanism. (2) Only TEE holds the RK and is capable of verifying the relation. By checking the PTC's legitimacy, RK is protected from being leaked when generating the corresponding TC. (3) The design handles corrupted TEE attacks because TEE only helps DU cancel out the blinding factor γ_1, without ever accessing the sensitive message M.

1.2.3. HR-ABE Construction. Our HR-ABE utilizes the above ABE-2OD as its main building block. Collusion attacks are prevented since ABE-2OD ensures that for any DU, without its RK (i.e., the help of TEE), no adversary

can learn any information about the underlying message of a target ciphertext, even if both TK and DK are corrupted.

While ABE-2OD looks intuitive to enable user revocation in an ABE system by allowing the TEE to hold RKs, the malicious CSP could launch rollback attacks by supplying a staled DU list to the TEE. Therefore, ensuring the integrity, especially the freshness, of the DU list is crucial for securing user revocation. We resist rollback attacks through two aspects: a signature from the KGC (signed on the latest DU list and a timestamp) and trusted replay-protection services provided by TEE (e.g., Intel SGX's RPS module).

To summarize, our HR-ABE system, built on top of an ABE-2OD, a signature scheme and TEE, answers the research question affirmatively. HR-ABE achieves DU revocation efficiently (i.e., no ciphertext delegation, no key update, lightweight transformation within TEE) and securely (i.e., resisting collusion attacks and corrupted TEE attacks).

sMHT as the state. In our HR-ABE system, the DU list is utilized as the TEE state. Since the list contains all the unrevoked DUs, the cost of loading the entire DU list to TEE will become unaffordable in practice. By leveraging an signed Merkle hash tree (sMHT) as the state, we are able to reduce the size of a membership proof, i.e., checking whether a DU is in the latest DU list, to logarithmic size.

1.3. Roadmap

The remainder of this paper is organized as follows. In Section 2, we present the related preliminaries. The detailed ABE-2OD and HR-ABE (including framework, definitions, construction, and security analysis) are presented in Section 3 and Section 4, respectively. After that, we describe how to use sMHT as the state in Section D and benchmark the performance in Section 5. Finally, we conclude the paper in Section 6. Besides, well-formed appendices are presented.

2. Preliminaries

2.1. Signature Scheme

Definition 1. *(Signature Scheme [17].) A signature scheme consists of three probabilistic algorithms* Π_{Sig} = (KeyGen, Sign, Ver) *between a signer and a verifier. Specifically,*
$(vk, sk) \leftarrow$ KeyGen(1^λ)*: The probabilistic key generation algorithm takes as input a security parameter* 1^λ*. It generates a signing key* sk *and a verification key* vk*.*
$\sigma \leftarrow$ Sign(sk, M)*: The probabilistic signature calculation algorithm takes as inputs the signing key* sk *and a message* M*. It calculates a signature* σ*.*
$\{0,1\} \leftarrow$ Ver(vk, M, σ)*: The deterministic signature verification algorithm takes as inputs* vk*,* M*, and* σ*. It outputs* 1 *if the signature is generated by* Sign*; otherwise, outputs* 0*.*

The existential unforgeability against chosen message attacks (EUF-CMA, [17]) is the standard security definition for digital signature schemes. In a nutshell, a signature scheme is said to be EUF-CMA secure if it is computationally infeasible for an attacker to produce a valid signature on a message that the attacker has not seen before, even if the attacker has access to a signature oracle that can produce signatures on arbitrary messages of the attacker's choice.

2.2. RCCA-secure OABE Scheme

LSSS access policy. Most of the attribute-based encryption schemes are based on the LSSS-style access policy, which is proved equivalent to the tree-style access policy [4]. For more details, please refer to Appendix A. Briefly, we represent an access policy as $(\mathbb{M}, \rho)$ and an attribute set as A. We denote by $A \vdash (\mathbb{M}, \rho)$ that an attribute set A satisfies an access policy $(\mathbb{M}, \rho)$.

Definition 2. *(ABE with Outsourced Decryption (OABE) [15].) An OABE scheme is executed among a KGC, a proxy server, multiple DOs, and multiple DUs. It consists of five algorithms. Specifically,*
(PK, MSK) $\leftarrow$ Setup$(1^\lambda, \Omega)$*: The probabilistic setup algorithm takes as input a security parameter* 1^λ *and an attribute universe* Ω*, KGC generates a public key PK and a master key MSK.*
$C \leftarrow$ Enc$(PK, M, (\mathbb{M}, \rho))$*: The probabilistic encryption algorithm takes as input a public key PK, a message* M*, and an LSSS-style policy* $(\mathbb{M}, \rho)$*, DO outputs a ciphertext C.*
(TK, SK) $\leftarrow$ KeyGen(MSK, A)*: The probabilistic key generation algorithm takes as input a master key MSK, and an attribute set* $A \subset \Omega$*, KGC outputs a transformation key TK and a secret key SK.*
$\{TC, \perp\} \leftarrow$ Transform(TK, C)*: The deterministic transformation algorithm takes as input a transformation key TK and a ciphertext C, proxy server outputs a transformed ciphertext TC if* $A \vdash (\mathbb{M}, \rho)$ *and the error symbol* $\perp$ *otherwise.*
$\{M, \perp\} \leftarrow$ Dec(SK, TC)*: The deterministic decryption algorithm takes as input a secret key SK and a transformed ciphertext TC, DU outputs a message* M *if* $A \vdash (\mathbb{M}, \rho)$*; otherwise, it outputs an error symbol* $\perp$*.*

The replayable chosen-ciphertext attacks (RCCA) security is a *weak variant* of the classical CCA security [8]. RCCA has shown its importance in outsourced decryption because it captures the following security: given a ciphertext, the proxy server cannot change the underlying message in a meaningful way. In fact, RCCA provides confidentiality and integrity guarantees for the underlying message during transformation, which ensures the security of outsourced ciphertexts. Please refer to Section 2.2 in [15] for OABE's RCCA security model.

2.3. Trusted Execution Environment (TEE)

TEE performs confidential computing based on user-defined programs in an isolated secure enclave. In a nutshell, TEE offers several essential features.

- *Attestation:* A client can build a secure channel with TEE by remote attestation. Without loss of generality, let K_{TEE} be the symmetric key shared between the client and TEE. Furthermore, let $\Pi_{AE} = (\mathsf{End}, \mathsf{Dec})$ be an authenticated

encryption scheme, then the communication between TEE and KGC is encrypted by $\mathsf{Enc}(\mathrm{K}_{\mathrm{TEE}}, \cdot)$ and decrypted by $\mathsf{Dec}(\mathrm{K}_{\mathrm{TEE}}, \cdot)$. The client installs a program prog [1] via

$$\mathcal{G}.\mathsf{install}(\mathsf{prog}).$$

- *Isolated execution:* TEE executes the program

$$(\mathsf{outp}, \Gamma) \leftarrow \mathcal{G}.\mathsf{resume}(\text{“prog”}, \mathsf{inp})$$

in the secure enclave. The execution within TEE is impervious to tampering by either software or physical attackers. After completing the execution, TEE sends the output, denoted as outp, along with a proof Γ, to the client. The proof Γ indicates that outp is indeed produced by the TEE.
- *Seal and unseal:* Seal, accomplished via $\Pi_{AE}.\mathsf{Enc}(\mathrm{K}_{\mathrm{TEE}}, \cdot)$, denotes TEE encrypting a secret and storing it in non-volatile external storage. Unseal, achieved through $\Pi_{AE}.\mathsf{Dec}(\mathrm{K}_{\mathrm{TEE}}, \cdot)$, denotes TEE retrieving the secret from external storage.

TEE provides trusted time service against rollback attacks. Any commodity TEE that provides this service can be utilized to deploy our HR-ABE system. For instance, Intel SGX uses converged security and management engine (CSME, e.g., protected real-time-clock), which is a dedicated firmware in the Platform Control Hub (PCH) [10]. By calling sgx_get_trusted_time, a trusted timestamp can be obtained. Additionally, CSME also has a replay-protected storage (RPS). Using sgx_create_monotonic_counter, we can store a counter value in RPS and retrieve the value from RPS by calling sgx_read_monotonic_counter.

Remark. While TEE ensures confidentiality protection on both input and output, it has been shown that commodity TEEs leak internal data (e.g., messages, states, intermediate results) due to side-channel attacks [13], [27], [32].

3. ABE-2OD: ABE with Two-Stage Outsourced Decryption

3.1. Syntax of ABE-2OD

Let $\mathcal{P}$ denote the policy space and $(\mathbb{M}, \rho) \in \mathcal{P}$ represent an LSSS-style policy. Let Ω denote the attribute universe and $A \subset \Omega$ be a set of attributes. Let $\mathcal{M}$ be the message space and $M \in \mathcal{M}$ be a message to be encrypted.

Definition 3. *(ABE-2OD). An ABE-2OD scheme Π is defined by the following six algorithms over the policy space $\mathcal{P}$, the attribute universe Ω, and the message space $\mathcal{M}$.*

- $(PK, MSK) \leftarrow \mathsf{Setup}(1^\lambda, \Omega)$*: The probabilistic setup algorithm takes as input a security parameter 1^λ and an attribute universe Ω. It outputs a public key PK and a master secret key MSK.*
- $C \leftarrow \mathsf{Enc}(PK, M, (\mathbb{M}, \rho))$*: The probabilistic encryption algorithm takes as input a public key PK, a message $M \in \mathcal{M}$, and an LSSS-style policy $(\mathbb{M}, \rho) \in \mathcal{P}$. It outputs a ciphertext C.*
- *(TK, HK, DK)* $\leftarrow \mathsf{KeyGen}(MSK, A)$*: The probabilistic key generation algorithm takes as input a master secret key MSK, and an attribute set $A \subset \Omega$. It outputs a key triple (TK, HK, DK), where TK is the transformation key, HK is the helper key, and DK is the decryption key.*
- $\{PTC, \perp\} \leftarrow \mathsf{Transform1}(TK, C)$*: The first transformation algorithm takes as input a transformation key TK and a ciphertext C. It outputs a partially transformed ciphertext PTC if $A \vdash (\mathbb{M}, \rho)$ and the error symbol $\perp$ otherwise.*
- $\{TC, \perp\} \leftarrow \mathsf{Transform2}(HK, PTC)$*: The second transformation algorithm takes as input a helper key HK and a partially transformed ciphertext PTC. It outputs a transformed ciphertext TC or an error symbol $\perp$.*
- $\{M, \perp\} \leftarrow \mathsf{Dec}(DK, TC)$*: The decryption algorithm takes as input a decryption key DK and a transformed ciphertext TC. It outputs M or an error symbol $\perp$.*

Decryption correctness. ABE-2OD requires that for any λ, Ω, $\forall A \in \Omega$, $\forall M \in \mathcal{M}$, $\forall (\mathbb{M}, \rho) \in \mathcal{P}$, (PK, MSK) $\leftarrow \mathsf{Setup}(1^\lambda, \Omega)$, C $\leftarrow \mathsf{Enc}(\mathrm{PK}, M, (\mathbb{M}, \rho))$, (TK, HK, DK) $\leftarrow \mathsf{KeyGen}(\mathrm{MSK}, A)$, PTC $\leftarrow \mathsf{Transform1}(\mathrm{TK}, \mathrm{C})$, and TC $\leftarrow \mathsf{Transform2}(\mathrm{HK}, \mathrm{PTC})$, we have

- If $A \vdash (\mathbb{M}, \rho)$,

$$\Pr[\mathsf{Dec}(\mathrm{DK}, \mathrm{TC}) = M] \geqslant 1 - negl(\lambda);$$

- Otherwise $A \nvdash (\mathbb{M}, \rho)$,

$$\Pr[\mathsf{Dec}(\mathrm{DK}, \mathrm{TC}) = M] \leqslant negl(\lambda);$$

where $negl(\lambda)$ is a negligible function.

Transformation correctness. We remark that decryption correctness implies transformation correctness. The transformation correctness means that if (TK, HK, DK) are generated properly, then we have that

$$\Pr[\mathsf{Transform2}(\mathrm{HK}, \mathrm{PTC}) = \perp] \leqslant negl(\lambda);$$

3.2. Security Models of ABE-2OD

Intuition. In the OABE system presented by Green et al. [15] in Section 2.2, message confidentiality is ensured even if attackers have access to the corresponding TK. This means that attackers without the DK cannot gain any information about the message M. In ABE-2OD, the original ABE private key is split into three parts: TK, HK, and DK. If an attacker corrupts only one of the two keys (HK or DK), they will not be able to decrypt a ciphertext, even if $A \vdash (\mathbb{M}, \rho)$. More precisely, while DK provides decryption security, HK guarantees transformation security.

- *HK for transformation security.* We define a formal security model, called sIND-CDA (selective indistinguishability against corrupted DK attack), which means attackers without HK learn nothing of M, even if both TK and DK are corrupted. To model the adversary's capabilities, we permit it to query an $\mathcal{O}_{\mathsf{CorruptDK}}$ oracle (which allows the adversary to corrupt DKs) and an $\mathcal{O}_{\mathsf{Transform2}}$ oracle

1. Here, we use the $\mathcal{G}$ abstraction that allows parties to recognize a genuine hardware deployed in the malicious host. Readers are referred to [28] for details.

(which allows the adversary to query a PTC, and then responds to the adversary a transformed result).
- *DK for decryption security.* We define another formal security model, called sIND-CHA (selective indistinguishability against corrupted HK attack), which ensures that attackers without DK learn nothing of M, even if both TK and HK are corrupted. In order to model the adversary's capabilities, we permit it to query an $\mathcal{O}_{\mathsf{CorruptHK}}$ oracle (which allows the adversary to corrupt HKs) and an $\mathcal{O}_{\mathsf{Dec}}$ oracle (which allows the adversary to query a TC, and then responds to the adversary a decrypted result).

Please refer to Appendix B.1 for detailed security models.

Remark. Note that (TK, HK, DK) is said to be a *consistent key tuple* if it is generated from one invocation of algorithm KeyGen. Since PTC is generated from a ciphertext C and a transformation key TK, a real adversary may generate the PTC from a modified ciphertext C' or an inconsistent TK$'$. We stress that sIND-CDA implies the consistency of TK whereas sIND-CHA implies the non-modifiable of C. Specifically, (1) if PTC is generated from an inconsistent TK$'$, Transform2 outputs $\perp$ with an overwhelming probability; (2) if PTC is generated from a modified ciphertext C', the probability that Dec outputs $\perp$ is also close to 1.

3.3. Design of ABE-2OD

Before presenting our design, we first briefly describe how Green's OABE [15] works. As shown in Figure 1, OABE involves a key pair $(\mathrm{TK}_{\alpha/z}, \mathrm{SK} = z)$. For decrypting a ciphertext, a transformation algorithm uses $\mathrm{TK}_{\alpha/z}$ to cancel the attribute-based access control layer in the ciphertext and outputs a transformed ciphertext, which is an El Gamal ciphertext in the group $\mathbb{G}_T$ and can be viewed as a ciphertext for the user who holds z as the private key. Finally, the decryption algorithm leverages z to recover the message.

3.3.1. Naive Design. To achieve sIND-CDA, one naive solution, based on OABE, is to split the OABE's SK = z into two parts (γ, β), where $z = \beta \cdot \gamma$, and then set HK = γ and DK = β. Then Transform1 uses TK to cancel out the access policy layer and generates a PTC = (C_0, C_P). After that, the algorithm Transform2 uses γ to produce a TC = (C_0, C_P^{γ}), which then is decrypted by β.

Arbitrary transformation queries. However, the naive solution cannot guarantee sIND-CDA security because the adversary may submit arbitrary queries to $\mathcal{O}_{\mathsf{Transform2}}$. For example, adversary can query a modified PTC$'$ $= (C_0', C_P' = h)$, where h can be randomly or specifically chosen by the adversary and $\mathcal{O}_{\mathsf{Transform2}}$ always outputs $(C_P')^{\gamma} = h^{\gamma}$ to the adversary. In such a way, adversary can obtain many pairs of (h, h^{γ}), which may lead to the leakage of γ. The absence of a legitimacy checking mechanism for PTC fundamentally causes the naive design to fail in achieving sIND-CDA.

3.3.2. Concrete Construction. To avoid the leakage of HK, the algorithm Transform2 must check whether a given PTC is legitimate without performing a full decryption. Based on Green's OABE [15], we present a concrete construction of ABE-2OD in Figure 2. Our approach to check a PTC's legitimacy is by setting HK = (γ_1, γ_2), which are chosen randomly in each invocation of KeyGen. The information of (γ_1, γ_2) is bound into a pair of OABE transformation keys TK = (TK_1, TK_2). With such an approach, a PTC consists of two components (C_{P1}, C_{P2}) and the relation $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$ holds only if PTC is calculated from the TK, that is consistent with HK.

The ciphertext of our ABE-2OD is the same as Green's RCCA OABE scheme. In detail, the access control layer $\{D_i, E_i\}_{i \in [\ell]}$ hides a secret (i.e., encryption randomness) s. Such a ciphertext is decrypted via three steps.

- *The first stage transformation.* During Transform1, if $A \vdash (\mathbb{M}, \rho)$, its corresponding TK = $(\mathrm{TK}_1, \mathrm{TK}_2)$, combined with the access policy layer $\{D_i, E_i\}_{i \in [\ell]}$ and the term $C'' = g^s$, can recover C_{P1} and C_{P2} correctly, which equal $e(g,g)^{\alpha s/(\beta\gamma_1)}$ and $e(g,g)^{\alpha s/(\beta\gamma_2)}$, respectively. In the end, it produces a PTC = (C, C', C_{P1}, C_{P2}), where C and C' remain unchanged.
- *The second stage transformation.* Note that C_{P1} and C_{P2} contain γ_1 and γ_2 in their exponents. Since HK = (γ_1, γ_2), Transform2 can verify whether $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$. If the relation holds, it means that PTC is a legitimate one. After that, it is able to cancel out the γ_1 in the exponent of C_{P1} and calculates a TC that consists of $(C, C', C_{P1}^{\gamma_1} = e(g,g)^{\alpha s/\beta})$.
- *The final decryption.* Obviously, TC is an El Gamal ciphertext in the group $\mathbb{G}_T$, but also with RCCA security. Since DK = β, algorithm Dec recovers and verifies the message M.

3.4. Security Analysis

3.4.1. sIND-CDA. sIND-CDA reflects that *given (TK, DK), attackers without corresponding HK learn nothing of M.*

From the construction, we can observe that only the party who holds HK can calculate the correct TC. Specifically, $C_{P1} = e(g,g)^{\alpha s/(\beta\gamma_1)}$ can also be treated as an El Gamal ciphertext under private key $1/\gamma_1$. Hence, if no information of γ_1 is leaked, C_{P1} appears like a random element of $\mathbb{G}_T$.

On the other hand, our ABE-2OD does not leak γ_1 to the adversary. First, Transform2 verifies the legitimacy of the PTC via checking $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$. It ensures γ_1 will not be leaked when performing Transform2 under the Knowledge of Exponent (KEA) assumption (refer to Appendix B for details). Secondly, the randomness t_1 and t_2 are leveraged to blind the (γ_1, γ_2) in $(\mathrm{TK}_1, \mathrm{TK}_2)$, which ensures the adversary cannot obtain the information of (γ_1, γ_2) from TKs. Lastly, the output TC = $e(g,g)^{\alpha s/\beta}$ contains no information of γ_1. Therefore, the adversary cannot recover M from C_{P1}, even DK is leaked. Hence, our ABE-2OD achieves sIND-CDA, which is formally described in THEOREM 1. The detailed proof is deferred to Appendix B.2.

Theorem 1. *ABE-2OD presented in Figure 2 is sIND-CDA if Green's OABE scheme (i.e., Π_{OABE}) is sIND-CPA secure. Specifically, let Π be the ABE-2OD scheme and $\mathcal{A}$ be an*

ABE-2OD: OABE with two-stage transformation

1. $(\text{PK}, \text{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, \Omega)$: // Setup

Let $(p, g, \mathbb{G}, \mathbb{G}_T)$ represents a bilinear map $e : \mathbb{G} \times \mathbb{G} \mapsto \mathbb{G}_T$, where p is the order of group $\mathbb{G}$ and g be a generator of $\mathbb{G}$. The setup algorithm first chooses two random exponents $\alpha, a \in \mathbb{Z}_p$ and three hash functions $H : \{0,1\}^* \mapsto \{0,1\}^k$, $H_1 : \{0,1\}^* \mapsto \mathbb{G}$, $H_2 : \{0,1\}^* \mapsto \mathbb{Z}_p$. Then it outputs the public key PK and the master secret key MSK as follows.

$$\text{PK} = \left(g, e\left(g,g\right)^{\alpha}, g^{a}, H, H_1, H_2\right), \text{MSK} = g^{\alpha}$$

2. $\text{C} \leftarrow \mathsf{Enc}\left(\text{PK}, M \in \{0,1\}^k, (\mathbb{M}, \rho)\right)$: // Encrypt

The encryption algorithm chooses a random element $R \in \mathbb{G}_T$ and a random vectors $\boldsymbol{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$ where $s = H_2(R, M)$. For each $i \in [\ell]$, it calculates $\lambda_i = (\mathbb{M} \cdot \boldsymbol{v})_i$. In addition, the algorithm also chooses ℓ random numbers $r_1, r_2, \cdots, r_\ell \in \mathbb{Z}_p$ and outputs the ciphertext C, which consists of three layers.

$$\boxed{\text{El Gamal layer}} \quad : \quad C = R \cdot e(g,g)^{\alpha s}, C' = H(R) \oplus M, C'' = g^{s};$$

$$\boxed{\text{access policy layer}} \quad : \quad \left(D_1 = g^{r_1}, E_1 = g^{a\lambda_1} H_1(\rho(1))^{-r_1}\right), \cdots, \left(D_\ell = g^{r_\ell}, E_\ell = g^{a\lambda_\ell} H_1(\rho(\ell))^{-r_\ell}\right).$$

3. $(\text{TK}, \text{HK}, \text{DK}) \leftarrow \mathsf{KeyGen}\left(\text{MSK}, A\right)$: // KeyGen

The key generation algorithm first chooses random numbers $t_1, t_2, \beta, \gamma_1, \gamma_2, \in \mathbb{Z}_p$, and sets the decryption key $\text{DK} = \beta$, the helper key $\text{HK} = (\gamma_1, \gamma_2)$, the transformation key $\text{TK} = (\text{TK}_1, \text{TK}_2)$, where

$$\text{TK}_i = \left(K_i = (g^{\alpha})^{\frac{1}{\beta\gamma_i}} (g^{a})^{\frac{t_i}{\beta\gamma_i}}, L_i = g^{\frac{t_i}{\beta\gamma_i}}, \{K_{y,i} = H_1(y)^{\frac{t_i}{\beta\gamma_i}}\}_{y \in A}\right); i \in \{1,2\}$$

4. $\{\text{PTC}, \perp\} \leftarrow \mathsf{Transform1}\left(\text{TK}, \text{C}\right)$: // Transform1

If A does not satisfy the access structure $(\mathbb{M}, \rho)$, it outputs $\perp$. Otherwise, let $I = \{i | \rho(i) \in A\}$ and it employs the method in Appendix. A to calculate $\{\lambda_i\}_{i \in I}$ and $\{w_i\}_{i \in I}$ such that $\sum_{i \in I} w_i \lambda_i = s$. Finally, it calculates a partially transformed ciphertext $\text{PTC} = (C, C', C_{P1}, C_{P2})$, where

$$C_{P1} = \frac{e(C'', K_1)}{\prod_{i \in I} \left(e(E_i, L_1) e(D_i, K_{\rho(i),1})\right)^{w_i}}, \qquad C_{P2} = \frac{e(C'', K_2)}{\prod_{i \in I} \left(e(E_i, L_2) e(D_i, K_{\rho(i),2})\right)^{w_i}};$$

5. $\{\text{TC}, \perp\} \leftarrow \mathsf{Transform2}\left(\text{HK}, \text{PTC}\right)$: // Transform2

This algorithm is the second stage transformation, which is run within TEE. It first verifies whether $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$ holds. If not, it outputs $\perp$. Otherwise, TEE outputs a TC as follows.

$$\text{TC} = \left(T = C, T' = C', T'' = C_{P1}^{\gamma_1}\right);$$

6. $\{M, \perp\} \leftarrow \mathsf{Decrypt}\left(\text{DK}, \text{TC}\right)$: // Decrypt

The decryption algorithm calculates $R = T/(T'')^{\beta}$, $M = H(R) \oplus T'$ and $s = H_2(R, M)$. If $T = R \cdot e(g,g)^{\alpha s}$ and $(T'')^{\beta} = e(g,g)^{\alpha s}$, it outputs M; otherwise it outputs $\perp$.

Figure 2: Detailed construction of ABE-2OD.

adversary for attacking Π*, there exists a simulator* $\mathcal{B}$ *such that* $\mathcal{A}$*'s advantage in breaking ABE-2OD's sIND-CDA is bounded by Eq. (1).*

$$\mathsf{Adv}^{\text{sIND-CDA}}_{\Pi,\mathcal{A}} \leqslant \mathsf{Adv}^{\text{sIND-CPA}}_{\Pi_{OABE},\mathcal{B}} + q_T \cdot \mathsf{Adv}^{\mathsf{KEA}}_{\mathcal{A},\mathcal{E}}. \tag{1}$$

where $\mathsf{Adv}^{\text{sIND-CPA}}_{\Pi_{OABE},\mathcal{B}}$ *is the advantages of* $\mathcal{B}$ *breaking* Π_{OABE}*'s CPA security,* $\mathsf{Adv}^{\mathsf{KEA}}_{\mathcal{A},\mathcal{E}}$ *is the advantage of breaking the KEA assumption, and* q_T *is the number of queries to the* $\mathcal{G}_{\mathsf{Transform2}(\cdot,\cdot)}$ *oracle.*

3.4.2. sIND-CHA. sIND-CHA reflects that *given (TK, HK), attackers without a valid DK learn nothing of* M.

DK is considered valid if the triple (TK, HK, DK) is generated via ABE-2OD's KeyGen from A such that $A \vdash (\mathbb{M}, \rho)$, where $(\mathbb{M}, \rho)$ is the access policy associated with the target ciphertext. Note that DK in our ABE-2OD essentially plays the same role as the user decryption key in the OABE scheme. Hence, the sIND-CHA of ABE-2OD is reduced to the RCCA security of Green's OABE scheme. Specifically, from the attacker's viewpoint, TC appears as an El Gamal ciphertext. Therefore, our ABE-2OD achieves sIND-CHA, which is formally described in THEOREM 2. The detailed proof is deferred to Appendix B.3.

Theorem 2. *ABE-2OD presented in Figure 2 is sIND-CHA if Green's OABE scheme is RCCA secure. Specifically, let* Π_{OABE} *be Green's OABE scheme, there exists a simulator* $\mathcal{B}$ *such that* $\mathcal{A}$*'s advantage in breaking the sIND-CHA is*

upper-bound by Eq. (2)

$$Adv^{sIN}_{\Pi,\mathcal{A}}$$

where $Adv^{RCCA}_{\Pi_{OABE},\mathcal{B}}$ *is adv security.*

4. Hardware-base

With the functionali in mind, this section pre ABE (HR-ABE) system HR-ABE in Section 4. design goals in Section describe the concrete co discuss the features of

4.1. System Archite

As shown in Figure 3, there are four types of entities involved in the HR-ABE framework.

- *A key generation center (KGC).* The KGC is mainly responsible for system setup (e.g., generating and broadcasting public parameters/keys to others 1, initializing all system modules 2) and user management (e.g., generating cryptographic keys for newly registered users or revoking users). Each user is associated with an attribute set A and three cryptographic keys: transformation key (TK), revocation key (RK), and decryption key (DK). A user list (UL) is maintained by KGC, which is updated whenever a user is registered or revoked. In detail, UL is defined as

$$\mathrm{UL} = \{ID, A_{ID}, \mathrm{TK}_{ID}, \mathsf{Enc}(\mathrm{K}_{\mathrm{TEE}}, \mathrm{RK}_{ID})\}_{ID\in\mathcal{U}},$$

  where $\mathcal{U}$ denotes the set of legitimate DUs, A_{ID} is the attribute set associated with ID, TK_{ID} (*resp.*, RK_{ID}, DK_{ID}) denotes the corresponding transformation key (*resp.*, revocation key, decryption key) for user ID, and $\mathrm{K}_{\mathrm{KEE}}$ is a shared symmetric authenticated encryption key established through the remote attestation between KGC and the TEE module.
  KGC operates over time intervals of length T each (e.g., hours, days). At the beginning of each time interval, KGC deletes the revoked DUs and adds newly joined DUs. The new DU list is denoted as UL_i and KGC generates a signed state st_i from UL_i.

$$\mathsf{st}_i = (\sigma, \mathrm{UL}_i, i),$$

  where σ is KGC's signature on the time interval index i and UL_i. KGC stores st_i in the state module 3.
- *Data owners (DOs).* DOs encrypt messages and upload the ciphertexts to the storage module and each ciphertext is associated with an access policy 4.
- *A cloud service provider (CSP).* The CSP is abundant in computing and storage resources and consists of four modules.

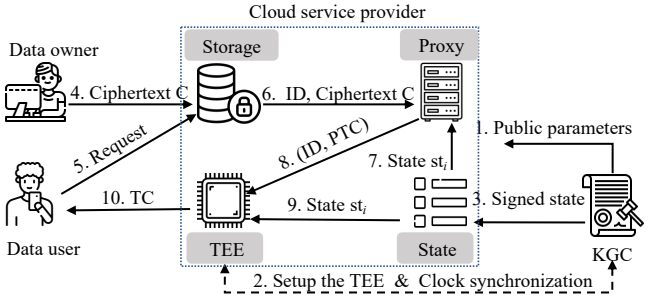


Figure 3: Framework of HR-ABE

  - Storage module is in charge of storing ciphertexts. Upon receiving a request for data access 5, storage module retrieves the requested ciphertext and then forwards both the ciphertext and the DU's identity (i.e., ID) to the proxy module 6.
  - State module maintains st_i, where i is the index of the current time interval.
  - Proxy module undertakes partial decryption operations. Upon receiving a request containing a user ID and a ciphertext, it retrieves st_i from the state module 7 and transforms the ciphertext to a partially transformed ciphertext PTC using ID's TK_{ID} (the first stage transformation, 8) if ID is in the latest DU list; otherwise, it rejects the request.
  - TEE module, holding $\mathrm{K}_{\mathrm{TEE}}$, provides trusted transformation services for unrevoked DUs. Note that $\mathrm{K}_{\mathrm{TEE}}$ can be used to recover RK_{ID}. Upon receiving (ID, PTC), TEE retrieves st_i from the state module 9 and verifies st_i's integrity and freshness. After that, TEE checks whether ID is indeed in the latest DU list. If yes, TEE calculates a transformed ciphertext TC using RK_{ID} and forwards it to DU 10; otherwise, TEE rejects the request.
- *Data users (DUs).* Each DU holds the DK_{ID} for executing the final decryption.

4.2. Threat Model

The KGC and DOs are fully trusted. CSP in HR-ABE is considered malicious and can launch active attacks, such as rollback attacks, collusion (with malicious DUs) attacks and corrupted TEE attacks. We categorize DUs into three types: (1) revoked DUs, (2) unrevoked and unauthorized DUs whose attribute sets fail to satisfy an access policy, and (3) unrevoked and authorized DUs whose attribute sets meet the access policy. The first two types of DU are considered malicious and can collude with CSP.

4.2.1. Rollback Attacks. Rollback attacks target breaking TEE state integrity or freshness. Specifically, CSP may attempt to modify the state of a time interval or provide a staled state to the TEE. Such attacks may reset the system to a previous state such that revoked DUs could decrypt

ciphertexts illegally. Indeed, existing commodity TEEs provide mechanisms to resist rollback attacks, such as Intel SGX's trusted time services [10], AMD SEV-SNP [2].

4.2.2. Collusion Attacks. CSP may collude with both revoked and unauthorized DUs and aim to break the confidentiality of HR-ABE. The collusion attacks between CSP and DUs could be in different forms. For example, given a target ciphertext, the CSP can transform it to a valid PTC and ask the TEE to perform the second stage transformation using the ID of an unrevoked but unauthorized DU, and subsequently attempts to decrypt the transformed ciphertext produced by the TEE using the decryption key of ID and that of an authorized but revoked DU. We define a formal security model, called selective indistinguishability against collusion attack (sIND-ColA), to capture the above intuitions under the completely trusted TEE model. Specifically, we introduce a corruption oracle in sIND-ColA that never reveal RKs to the adversary, which reflects the assumption that TEE provides internal state confidentiality. Appendix C.1 presents this detailed security model.

4.2.3. Corrupted TEE Attacks. In addition to the above attacks, as the host of the TEE, a malicious CSP may also perform side-channel attacks [13], [27], [32] to observe some information inside the TEE, which is leaked during the TEE computation. Without loss of generality, the attacks that could lead to TEE internal information leakage are called corrupted TEE attacks (CTA). We define a formal security model, called selective indistinguishability against corrupted TEE attacks (sIND-CTA), to capture the security of HR-ABE under the transparent TEE model. Specifically, we let the corruption oracle in sIND-CTA reveal RKs to the adversary, which reflects the threat model that TEE does not provide internal state confidentiality as described before. Appendix C.2 presents the detailed security model.

4.3. Design Goals

According to the system architecture and threat model, HR-ABE should achieve the following goals.

- *G1: Efficient revocation.* An HR-ABE is said to be efficient if the following sub-goals are achieved.
  - G1-1: No ciphertext delegations and no key updates. CSP does not need to perform ciphertext delegation to prevent revoked DUs from decrypting past ciphertexts. Non-revoked DUs do not need to update their decryption keys periodically.
  - G1-2: Outsourced decryption. Instead of computing extensive bilinear map operations as in traditional ABE, DUs just decrypt an EL Gamal-like ciphertext.
  - G1-3: Lightweight transformation within TEE. The transformation within TEE only takes few cryptographic operations that are independent of the size of the access policy.
- *G2: Secure revocation against collusion attacks.* CSP colluding with both revoked and unauthorized DUs cannot extract any useful information from target ciphertexts. HR-ABE should also resist rollback attacks to ensure that TEE only provides the second transformation services for non-revoked DUs.
- *G3: Semantic security against corrupted TEE attacks.* Revocation key leakage does not lead to the loss of message confidentiality. In other words, HR-ABE achieves the same security level as the OABE system, when TEE is corrupted.

Remark. HR-ABE is designed to resist either collusion attacks or corrupted TEE attacks, but not both. It is easy to see if an adversary gains access to the entire key tuple (TK_{ID}, RK_{ID}, DK_{ID}), which is the case when considering both attacks simultaneously, it is impossible to create an HR-ABE scheme that can preserve security.

4.4. Construction

We now present our HR-ABE scheme from an ABE-2OD scheme and a signature scheme. In a nutshell, we use ABE-2OD to handle the key generation, encryption, two stages for transformation, and decryption operations of HR-ABE, and the signature scheme combined with TEE's replay-protection service to ensure the integrity and freshness of the user list. Specifically, an HR-ABE system consists of four phases: system setup, encryption, state update, and decryption, as shown in Figure 5.

4.4.1. System Setup Phase. The system setup phase generates public/private keys and initializes the modules of the whole system. Let Π_1 be an ABE-2OD scheme and Π_{Sig} be a signature scheme. KGC generates the system secret key MSK, which includes three components: Π_1's master key MSK_1, Π_{Sig}'s singing key, and an authenticated encryption key $\mathrm{K}_{\mathrm{TEE}}$ between KGC and TEE. Two programs prog_1 and prog_2 are installed into TEE via

$$\mathcal{G}.\mathsf{install}(\mathsf{prog}_1) \quad \text{and} \quad \mathcal{G}.\mathsf{install}(\mathsf{prog}_2).$$

We present details of the two programs in Figure 4.

- prog_1 synchronizes the starting time t_0 between TEE and KGC. Note that t_0 and the duration of a time interval T are stored in the replay-protected storage (RPS), which is a core module in TEE (e.g., Intel SGX) that provides trusted replay protection services.
- prog_2 provides transformation services for non-revoked DUs, which works in a "validate-then-transform" manner. *Only* a non-revoked DU can pass the validations (i.e., state freshness and integrity validation, and DU is non-revoked) and obtain a TC.

4.4.2. Encryption Phase. DO specifies an LSSS-style policy, encrypts the message, and uploads the ciphertext to the storage module. Note that the ciphertexts in HR-ABE are identical to those in ABE-2OD. For practical deployment, HR-ABE could utilize hybrid encryption [20], i.e., HR-ABE only encrypts a symmetric key which encrypts the real data.

Enclave operations

$/ * *$ prog_1: synchronizes the time interval $* * /$
$\mathcal{G}.\mathsf{resume}(\text{“Synchronize”}, T)$

TEE leverages sgx_get_trusted_time() to extract a trusted timestamp t_0 and then stores t_0 and T into RPS via sgx_create_monotonic_counter(). Finally, TEE returns t_0 to KGC.

$/ * *$ prog_2: provides transformation services $* * /$
$\{\text{TC}, \perp\} \leftarrow \mathcal{G}.\mathsf{resume}(\text{“Transform2”}, \text{K}_{\text{TEE}}, \mathsf{st}_i, ID, \text{PTC})$

Validation :

(1) TEE calculates a trusted time interval index $i^* = \lfloor (t_{cur} - t_0)/T \rfloor$, where $t_{cur} \leftarrow$ sgx_get_trusted_time(). If $i^* \neq i$, TEE aborts.
(2) TEE parses $\mathsf{st}_i = (\sigma, \text{UL}_i, i)$ and calculates

$$b_1 \leftarrow \Pi_{Sig}.\mathsf{Ver}(vk, \langle \text{UL}_i, i \rangle, \sigma); \quad b_2 \leftarrow (ID \in \text{UL}_i).$$

If $b_1 = 0$ or $b_2 = 0$, it aborts.

Transform2 :

TEE fetches $\mathsf{Enc}(\text{K}_{\text{TEE}}, \text{RK}_{ID})$ from UL_i and decrypts it to RK_{ID} using K_{TEE}. Finally, TEE outputs

$$\text{TC} \leftarrow \Pi_1.\mathsf{Transform2}(\text{RK}_{ID}, \text{PTC}).$$

Figure 4: Enclave operations in HR-ABE

4.4.3. State Update Phase. KGC manages the DU list. Newly joining DU's keys are produced by ABE-2OD's key generation algorithm. At the beginning of a time interval (say i-th time interval), KGC signs DU list on time interval i and generates a new state $\mathsf{st}_i = (\sigma, \text{UL}_i, i)$. KGC updates the state module with (σ, UL_i, i), which makes user registration and revocation effect.

4.4.4. Decryption Phase. The decryption phase includes three sub-phases. Both CSP and TEE need to validate whether the DU is legitimate before executing the respective transformation algorithm.

- *The 1st transformation by proxy module.* The proxy retrieves st_i from the state module and validates whether the DU is legitimate. If yes, the proxy generates a PTC via ABE-2OD's $\mathsf{Transform1}$ and forwards it to TEE.
- *The 2nd transformation by TEE module.* TEE runs prog_2 and produces a TC via ABE-2OD's $\mathsf{Transform2}$.
- *The final decryption by DU.* The non-revoked DU recovers the message from TC via ABE-2OD's Dec algorithm.

4.5. Discussions

In this section, we discuss how HR-ABE thwarts attacks outlined in Section 4.2 and fulfills the design goals.

4.5.1. HR-ABE Achieves G1. Since HR-ABE does not contain any timestamp in ciphertexts or decryption keys, it doesn't require periodically updating all previous ciphertexts or DUs' decryption keys. Instead of relying solely on cryptographic design, HR-ABE shifts the management of DUs to in-enclave design for achieving efficient DU revocation (i.e., G1-1).

As for G1-2, in HR-ABE, TK_{ID} held by CSP is for validating the access control, and DK_{ID} maintained by DU is for decrypting a simple El Gamal ciphertext. Thus, HR-ABE validates the access control on CSP side and largely eliminates ABE decryption overhead for resource-constrained DUs. HR-ABE achieves G1-3 because the computation within TEE only includes three exponentiations.

4.5.2. Collusion Attack Resistance (G2). In the system setup phase, the KGC instructs TEE to store a trusted timestamp t_0 and the duration of a time interval T in Intel SGX's RPS [10]. Since RPS is replay-protected storage provided by TEE, no adversary can tamper with t_0 or T. TEE also retrieves the current time via its protected real-time clock, hence the correctness of the time interval index i can be guaranteed. On the other hand, the KGC utilizes Π_{Sig} to sign both the latest DU list and the current time interval i. The unforgeability property of Π_{Sig} ensures that the TEE always uses the latest state to verify whether a DU is revoked or not. Therefore, our HR-ABE scheme is resistant to rollback attacks.

Given that no rollback attack would happen, HR-ABE achieves collusion attack resistance based on two facts: (1) TEE only performs the second stage transformation for non-revoked DUs, and (2) the adversary, who corrupts the CSP and revoked DUs simultaneously, cannot obtain the revocation key RK. Therefore, HR-ABE is resistant to collusion attacks. Furthermore, in case CSP asks the TEE to perform a transformation using the ID of an unrevoked DU and then attempts to decrypt the transformed ciphertext using the DK of an authorized but revoked DU. This type of collusion attack is infeasible because of our PTC legitimacy checking mechanism. Formally, we have the following result. The proof is detailed in Appendix C.3.

Theorem 3. *The HR-ABE scheme in Figure 5 is sIND-ColA secure if Π_1 is sIND-CDA, Π_{Sig} is EUF-CMA, and Π_{AE} is CCA secure. Specifically, there exist simulators $\mathcal{B}$, $\mathcal{B}_1$ and $\mathcal{F}$, such that $\mathcal{A}$'s advantage in breaking HR-ABE's sIND-ColA security is upper-bounded by Eq. (3).*

$$\mathsf{Adv}^{\text{sIND-ColA}}_{\Pi^{\mathcal{G}},\mathcal{A}} \leqslant \mathsf{Adv}^{\text{EUF-CMA}}_{\Pi_{Sig},\mathcal{F}} + \mathsf{Adv}^{\text{CCA}}_{\Pi_{AE},\mathcal{B}_1} + \mathsf{Adv}^{\text{sIND-CDA}}_{\Pi_1,\mathcal{B}}, \tag{3}$$

where $\mathsf{Adv}^{\text{EUF-CMA}}_{\Pi_{Sig},\mathcal{F}}$ is the advantage of $\mathcal{F}$ breaks Π_{Sig}'s unforgeability and $\mathsf{Adv}^{\text{CCA}}_{\Pi_{AE},\mathcal{B}_1}$ is the advantage of $\mathcal{B}_1$ breaking the authenticated encryption scheme.

4.5.3. Corrupted TEE Attack Resistance (G3). Our HR-ABE presented in Figure 5 achieves corrupted TEE attack resistance through ABE-2OD's sIND-CHA security, which ensures that an adversary, whose attribute set does not satisfy the access policy associated with a specific ciphertext, learns

HR-ABE Construction

System setup phase : Let Π_1 = (Setup, Enc, KeyGen, Transform1, Transform2, Dec) be an ABE-2OD scheme and Π_{Sig} = (KeyGen, Sign, Ver) be a signature scheme. This phase consists of one algorithm, denoted as $(\text{MPK}, \text{MSK}, \mathsf{st}_0, \text{UL}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^\lambda, \Omega, T)$, that works as follows.

- KGC initializes TEE and shares a symmetric key K_{TEE} via remote attestation. Then it loads programs prog_1 and prog_2 by $\mathcal{G}.\mathsf{install}(\mathsf{prog}_i), i \in \{1, 2\}$; Then KGC invokes the prog_1 by $\mathcal{G}.\mathsf{resume}$("Synchronize", T);
- KGC sets $\text{MSK} = (\text{MSK}_1, sk, \text{K}_{\text{TEE}})$ and $\text{MPK} = (\text{PK}_1, vk)$, where

$$(\text{PK}_1, \text{MSK}_1) \leftarrow \Pi_1.\mathsf{Setup}(1^\lambda, \Omega) \quad \text{and} \quad (vk, sk) \leftarrow \Pi_{Sig}.\mathsf{KeyGen}(1^\lambda).$$

- KGC runs $\mathsf{st}_0 \leftarrow \mathsf{Update}\,(\text{MSK}, \text{UL}, 0)$ and outputs $(\text{MPK}, \text{MSK}, \mathsf{st}_0, \text{UL})$.

Encryption phase : During the encryption phase, DOs encrypt messages, denoted as $\text{C} \leftarrow \mathsf{Enc}\,(\text{MPK}, M, (\mathbb{M}, \rho))$ and upload the ciphertexts to CSP's storage module. It produces the ciphertext by straightforwardly invoking

$$\text{C} \leftarrow \Pi_1.\mathsf{Enc}\,(\text{PK}_1, M, (\mathbb{M}, \rho))\,.$$

State update phase : This phase includes three algorithms: Join (i.e., new DUs registration and their key generation), Rev (i.e., DU revocation), and Update (i.e., making registrations or revocations effectiveness). Specifically, the three algorithms work as follows.

- $(\text{UL}, (\text{TK}_{ID}, \text{RK}_{ID}, \text{DK}_{ID})) \leftarrow \mathsf{Join}\,(\text{MSK}, \text{UL}, ID, A)$: KGC generates keys for ID and updates UL through

$$(\text{TK}_{ID}, \text{HK}_{ID}, \text{DK}_{ID}) \leftarrow \Pi_1.\mathsf{KeyGen}\,(\text{MSK}_1, A)\,, \text{UL} = \text{UL} \cup \{ID, A, \text{TK}_{ID}, \mathsf{Enc}(\text{K}_{\text{TEE}}, \text{RK}_{ID})\},\ \text{RK}_{ID} = \text{HK}_{ID}.$$

- $\text{UL} \leftarrow \mathsf{Rev}\,(\text{UL}, ID)$: KGC deletes the record associated with ID from UL and outputs an updated DU list UL.
- $\mathsf{st}_i \leftarrow \mathsf{Update}\,(\text{MSK}, \text{UL}, i)$: KGC generates a new state $\text{st}_i = (\sigma, \text{UL}_i, i)$ of i-th time interval, where

$$\sigma \leftarrow \Pi_{Sig}.\mathsf{Sign}(sk, \langle \text{UL}_i, i \rangle).$$

Decryption phase : The decryption phase includes three algorithms, as in the ABE-2OD.

- $\text{PTC} \leftarrow \mathsf{Transform1}\ (\text{TK}_{ID}, \mathsf{st}_i, ID, \text{C})$: The proxy parses $\mathsf{st}_i = (\sigma, \text{UL}_i, i)$ and calculates $b_1 \leftarrow \Pi_{Sig}.\mathsf{Ver}(vk, \langle \text{UL}_i, i \rangle, \sigma)$ and $b_2 \leftarrow (ID \in \text{UL}_i)$. If $b_1 = 0$ or $b_2 = 0$, it aborts; otherwise, it fetches TK_{ID} from UL_i and outputs a PTC via

$$\text{PTC} \leftarrow \Pi_1.\mathsf{Transform1}\,(\text{TK}_{ID}, \text{C})\,.$$

- TEE produces a TC by calling the prog_2 that was installed into TEE during the system setup phase.

$$\text{TC} \leftarrow \mathcal{G}.\mathsf{resume}(\text{"Transform2"}, \text{K}_{\text{TEE}}, \mathsf{st}_i, ID, \text{PTC}).$$

- $M \leftarrow \mathsf{Dec}\,(\text{DK}_{ID}, \text{TC})$: The DU decrypts TC to M by running

$$M \leftarrow \Pi_1.\mathsf{Dec}\,(\text{DK}_{ID}, \text{TC})\,.$$

Figure 5: HR-ABE construction from ABE-2OD.

nothing of the underlying message, even if the adversary obtains both TK and RK. Formally, we have the following Theorem.

Theorem 4. *The HR-ABE scheme in Figure 5 is sIND-CTA secure if Π_1 is sIND-CHA. Specifically, there exists a simulator $\mathcal{B}$ such that $\mathcal{A}$'s advantage in breaking HR-ABE's sIND-CTA security is upper-bound by Eq. (4).*

$$\mathsf{Adv}^{\text{sIND-CTA}}_{\Pi^{\mathcal{G}}, \mathcal{A}} \leqslant \mathsf{Adv}^{\text{sIND-CHA}}_{\Pi_1, \mathcal{B}}. \tag{4}$$

The proof is straightforward and we omit this proof due to the page limit.

4.6. sMHT as The State

The HR-ABE construction in Section 4.4 loads the entire DU list into TEE for state validation because the KGC's signature is generated for the whole up-to-date DU list. Suppose we use a 32-bit number to present an identity ID and each attribute. Let N be the number of DUs, then the size of the DU list is

$$(32 + 32 \cdot |A| + (|A| + 2) \cdot |\mathbb{G}| + |\mathbb{Z}_p|) \cdot N,$$

where $|A|$ is the number of attributes in set A, $|\mathbb{G}|$ and $|\mathbb{Z}_p|$ are the length of each item in $\mathbb{G}$ and $\mathbb{Z}_p$, respectively. Obviously, with the growth of the DU number in the system,

the cost of loading the entire DU list to TEE will become unaffordable. We observe that the state module in HR-ABE is "plug-and-play", and one can replace the implementation of the state module with any authenticated data structure (ADS [18]) for verifying $ID \in \text{UL}_i$. In other words, the primitives (such as Merkle hash tree [26], Verkle tree [21]) for membership proof can be plugged into HR-ABE. For example, MHT is a binary hash tree that allows provers to generate a short membership proof.

sMHT, also called signed tree head (SHT), is a popular technology in Google's certificate transparency [22]. sMHT has the following complexity: let N be the size of UL, then the storage size of sMHT is $O(N)$, the proof size is $O(\log N)$. Besides, the time complexities of Build, GenProof, and VerProof are $O(N)$, $O(\log N)$, and $O(\log N)$, respectively. By integrating sMHT into our HR-ABE, the data loaded into TEE for membership proof include only one element in UL and the corresponding proof Γ, which has a logarithmic size. Assuming $N = 10^6$, the input size of TEE for one decryption request is about 12Kb. Even for SGX with about 90MB usable memory, it can handle tens of thousands of requests. The detailed approach for integrating sMHT into HR-ABE is deferred to Appendix D.

5. Performance Evaluation

5.1. Experimental Configuration

Baseline. To report the performance of HR-ABE, we compare HR-ABE with the baseline system in which TEE maintains the TKs and performs the entire OABE's transformation algorithm (i.e., canceling the access control layer), and DU performs the final decryption.

We implement HR-ABE and the baseline in C++ and conduct experiments on a server with 64-bit Ubuntu 20.04 LTS operating system, Intel Core i7-8700 CPU with 6 physical cores and two threads per core, totaling 12 threads, and 32GB of total memory. Type A elliptic curve (rbits: 512, qbits: 1024) in the symmetric setting (PBC library [2]) is used in our implementation to evaluate the pairing operations. We use the Intel SGX in hardware mode [3] to implement TEE. All experiments are measured 20 times, and we take their average as the experimental result. In our experiments, we utilize multiple threads (denoted N_{thread}, i.e., 1, 3, 6) to simulate non-TEE resources and one thread to simulate TEE (i.e., by setting TSCNum = 1 in the file "Envclave.configure.xml").

Access policy setup. Both the transformation and encryption algorithm in the baseline and HR-ABE depend on the complexity of the access policy. In our experiments, we use access policies in the form

$$A_1 \text{ AND } A_2 \text{ AND } \cdots \text{ AND } A_\ell$$

, where ℓ denotes the complexity of the access policy.

2. https://crypto.stanford.edu/pbc/
3. https://github.com/intel/linux-sgx.git

TABLE 2: Time cost (ms) $v.s.$ the policy size

Policy size	Baseline		HR-ABE		
	Transform in TEE	Decryption by DU	Transform1 by CSP	Transform2 in TEE	Decryption by DU
10	166.17	7.62	225.23	3.22	7.61
20	324.36	7.61	430.15	3.23	7.61
30	482.97	7.62	637.71	3.22	7.62

Remark. We note that the baseline does not consider the transformation key security problem due to arbitrary transformation queries to TEE.

Workload and performance metrics. We call a ciphertext decryption request a task and use N_{task} to denote the number of tasks. For both HR-ABE and the baseline, we measure the *average latency.* Note there are N_{thread} threads to execute the non-TEE part and one thread to execute the TEE part. When $N_{\text{task}} > N_{\text{thread}}$, some of the tasks enter a waiting state and wait for the resource to become available. The duration of wait time (denoted T_{wait}) directly affects system responsiveness. We define each task's latency as $T_{\text{wait}} + T_{\text{execute}}$, where T_{execute} is the task's execution time. Let T_i denote the i-th task's latency, then the workload's average latency is defined as $(\sum_{i=1}^{N_{\text{task}}} T_i)/N_{\text{task}}$.

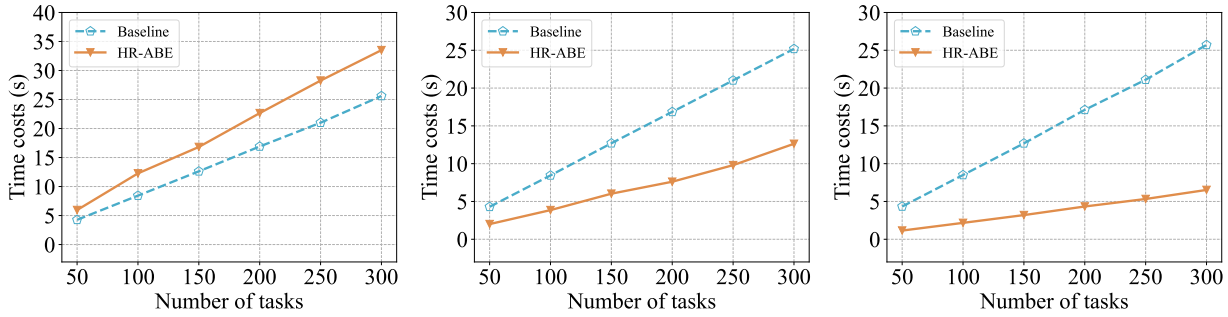
5.2. Performance Comparison with Baseline

We first benchmark the time cost of the transformation and decryption algorithms. Table 2 presents the performance when the policy size varies from 10 to 30. In the baseline, the transformation is run in TEE, and the transformation cost grows from 166 (ms) to 483 (ms). In HR-ABE, only the transform2 is run in TEE; the time cost for non-TEE (i.e., CSP) part grows from 225 to 638 (ms), and the time cost for TEE part is only about 3.2 (ms). Therefore our HR-ABE achieves a lightweight transformation in TEE. HR-ABE achieves lightweight decryption for end users, with a decryption cost of about 7.6 (ms).

Note that the results shown in Table 2 only consider one thread and one task. In the following experiments, we will showcase how the performance of HR-ABE is impacted by the complexity of the policy (ℓ), as well as the number of tasks (N_{task}) and the number of threads (N_{thread}) involved.

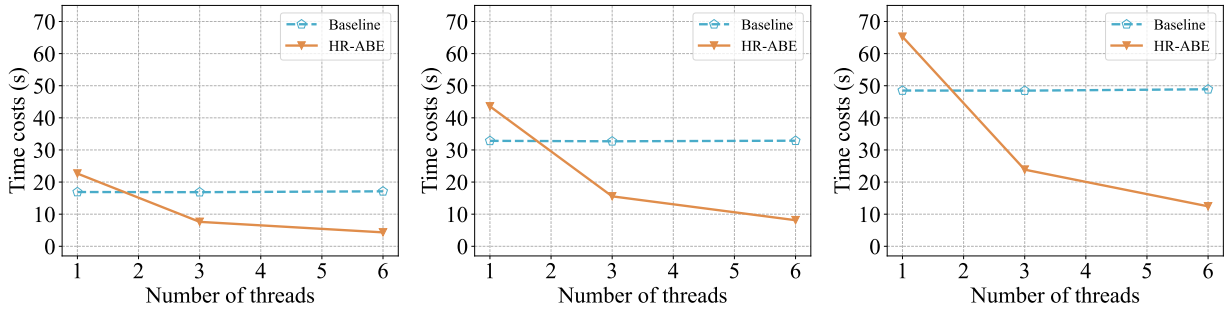
(1) The effect of N_{task}. Figure 6 presents the performance when $N_{\text{task}} = \{50, 100, 150, 200, 250, 300\}$. In this experiment, we set $\ell = 10$. The average latency is linear to the task numbers in both the baseline and HR-ABE. When $N_{\text{thread}} = 1$, HR-ABE is slightly slower than the baseline, since our HR-ABE needs to calculate C_{P1} and C_{P2}, which cancels out the access policy layer two times. When $N_{\text{task}} = 300$, the average latency in HR-ABE is about 1.99-3.76 times better than the baseline.

(2) The effect of N_{thread}. Figure 7 presents the performance when $N_{\text{thread}} = \{1, 3, 6\}$. In this experiment, we set $\ell = \{10, 20, 30\}$ and $N_{\text{task}} = 200$. On the one hand, when $N_{\text{thread}} = \{3, 6\}$, HR-ABE is 3.71 – 4.14 times better. As a result, it shows that the greater the difference in computing power between TEE and non-TEE parts, the better the performance of HR-ABE. On the other hand, as the size



(a) $N_{\text{thread}} = 1$ (b) $N_{\text{thread}} = 3$ (c) $N_{\text{thread}} = 6$

Figure 6: Comparisons between baseline and HR-ABE when number of tasks varies.



(a) Policy size: 10 (b) Policy size: 20 (c) Policy size: 30

Figure 7: Comparisons between baseline and HR-ABE for different policy sizes.

TABLE 3: Revocation cost (ms) $v.s.$ the numbers of DUs

Scheme		2^{10}	2^{12}	2^{14}	2^{16}	2^{18}
SR-ABE	KGC	1217.10	1469.87	1822.06	1952.48	2191.37
	CSP	1576.09	1898.30	2354.69	2530.42	2840.29
HR-ABE	KGC	0.67	0.70	0.73	0.75	0.76
	CSP	0.15	0.16	0.19	0.21	0.22

of policy ℓ increases from 10 to 30, the performance gap between them becomes significantly wider.

In our experiment, the performance bottleneck comes from the CSP (225ms for policy_size=10) instead of the TEE (3.2ms), since CSP needs to perform heavy computations, and we only used a PC to simulate the CSP. In a real cloud setting, the computation power of CSP is software-only and can be scaled up with increasing resources.

5.3. Revocation Comparison with Prior Work

SR-ABE [12] is the state-of-the-art server-aided revocable ABE. As shown in TABLE 3, HR-ABE is much more efficient in user revocation than SR-ABE. Note that user revocation in SR-ABE requires both KGC and CSP to compute a large number of exponential operations, while user revocation in HR-ABE only requires KGC and CSP to perform some hash computations to update the sMHT. As the number of DUs varies from 2^{10} to 2^{18}, the costs for KGC and CSP in HR-ABE are capped at 0.756 (ms) and 0.22 (ms), respectively.

6. Conclusion

While existing user revocation mechanisms in ABE present a challenge due to the unscalable key update and ciphertext delegation, this paper proposed hardware-based revocable ABE (HR-ABE) that eliminates periodic key updates and ciphertext delegation, and supports outsourced decryption so that both TEE and data users only perform lightweight computation. Another attractive feature of HR-ABE is that the TEE only enforces user revocation but cannot access the user data. We formalized the security models for HR-ABE and provided formal security proofs for the proposed scheme. Experimental benchmarks were also provided to demonstrate the performance of the proposed scheme under different settings.

Acknowledgements

This research was supported by the Singapore National Research Foundation under NCR Award Number NRF2018NCR-NSOE004-0001. It was also supported by the National Key R&D Program of China under Grant 2022YFB3103500, the Lee Kong Chian Chair Professor Fund, the AXA Research Fund, and the Lee Kong Chian Fellowship Fund.

References

[1] Shashank Agrawal and Melissa Chase. FAME: fast attribute-based message encryption. In *CCS*, 2017.

[2] AMD. SEV-SNP: Strenthing vm isolation with integrity protection and more, 2020.

[3] Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In *Conference on Cryptography and Coding*, 2009.

[4] Amos Beimel. Secure schemes for secret sharing and key distribution. *Thesis of Technion-Israel Institute of technology*, 1996.

[5] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In *CRYPTO*, 2004.

[6] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, 1998.

[7] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. In *CCS*, 2008.

[8] Ran Canetti, Hugo Krawczyk, and Jesper B Nielsen. Relaxing chosen-ciphertext security. In *CRYPTO*, 2003.

[9] Ling Cheung and Calvin Newport. Provably secure ciphertext policy abe. In *CCS*, 2007.

[10] Intel Corporation. Trusted time and monotonic counters with intel® software guard extensions platform services, 2017.

[11] Victor Costan and Srinivas Devadas. Intel SGX explained. *Manuscript*, 2016.

[12] Hui Cui, Robert H. Deng, Yingjiu Li, and Baodong Qin. Server-aided revocable attribute-based encryption. In *ESORICS*, 2016.

[13] Shufan Fei, Zheng Yan, Wenxiu Ding, and Haomeng Xie. Security vulnerabilities of SGX and countermeasures: A survey. *CSUR*, 2021.

[14] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, 2006.

[15] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security*, 2011.

[16] David Kaplan, Jeremy Powell, and Tom Woller. AMD memory encryption. *White paper*, 2016.

[17] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. 2020.

[18] Igjae Kim, J Hyun Kim, Minu Chung, Hyungon Moon, and Sam H Noh. A log-structured merge tree-aware message authentication scheme for persistent key-value stores. In *USENIX FAST*, 2022.

[19] Nishant Kumar, Mayank Rathee, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow: Secure tensorflow inference. In *S&P*, 2020.

[20] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *CRYPTO*, 2004.

[21] John Kuszmaul. Verkle trees. *Verkle Trees*, 1, 2019.

[22] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. Technical report, 2013.

[23] Jin Li, Xiaofeng Chen, Jingwei Li, Chunfu Jia, Jianfeng Ma, and Wenjing Lou. Fine-grained access control system based on outsourced attribute-based encryption. In *ESORICS*, 2013.

[24] Joseph K Liu, Tsz Hon Yuen, Peng Zhang, and Kaitai Liang. Time-based direct revocable ciphertext-policy attribute-based encryption with short revocation list. In *ACNS*, 2018.

[25] Zhen Liu, Zhenfu Cao, and Duncan S Wong. Efficient generation of linear secret sharing scheme matrices from threshold access trees. *Cryptology ePrint Archive*, 2010.

[26] Ralph Charles Merkle. *Secrecy, authentication, and public key systems.* Stanford University, 1979.

[27] Kit Murdock, David Oswald, Flavio D Garcia, Jo Van Bulck, Daniel Gruss, and Frank Piessens. Plundervolt: Software-based fault injection attacks against intel SGX. In *S&P*, 2020.

[28] Rafael Pass, Elaine Shi, and Florian Tramer. Formal abstractions for attested execution secure processors. In *EUROCRYPT*, 2017.

[29] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *CCS*, 2013.

[30] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In *CRYPTO*, 2012.

[31] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *cryptology eprint archive*, 2004.

[32] Stephan van Schaik, Marina Minkin, Andrew Kwong, Daniel Genkin, and Yuval Yarom. Cacheout: Leaking data on intel cpus via cache evictions. In *S&P*, 2021.

[33] Pengfei Wu, Jianting Ning, Jiamin Shen, Hongbing Wang, and Ee-Chien Chang. Hybrid trust multi-party computation with trusted execution environment. In *NDSS*, 2022.

[34] Shucheng Yu, Cong Wang, Kui Ren, and Wenjing Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM*, 2010.

[35] Yinghui Zhang, Robert H. Deng, Shengmin Xu, Jianfei Sun, Qi Li, and Dong Zheng. Attribute-based encryption for cloud computing access control: A survey. *CSUR*, 2020.

A. Linear Secret Sharing Scheme (LSSS)

Definition 4. *(LSSS [4].) A secret sharing scheme over a set of parties Ω is called linear if there exists an $\ell \times n$ matrix $\mathbb{M}$ and a label function $\rho : [\ell] \mapsto \Omega$ [4] such that for any random vector $\boldsymbol{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$, where s represents the secret, we have that $\mathbb{M}\boldsymbol{v}$ shares the secret s to parties, and $(\mathbb{M}\boldsymbol{v})_i$ belongs to the party $\rho(i)$. Usually, the secret scheme is denoted as a pair $(\mathbb{M}, \rho)$.*

Let $(\mathbb{M}, \rho)$ be a LSSS scheme, then for any authorized set $A \subset \Omega$, (1) there exists a set of valid shares $\boldsymbol{\lambda} = \{\lambda_i \in \mathbb{Z}_p\}_{i \in I}$, where $I = \{i : \rho(i) \in A\}$; and (2) there exists an efficient algorithm [25] to calculate a set of constants $\boldsymbol{w} = \{w_i \in \mathbb{Z}_p\}_{i \in I}$ to recover the secret from the valid shares by $\sum_{i \in I} w_i \lambda_i = s$. Here we only assume the existence of such constants. Refer to [25] for the detailed algorithm, which can be optimized in real implementations.

B. ABE-2OD Security Models and Proofs

Assumption 1. *(KEA [5]). For any adversary $\mathcal{A}$ that takes $g^{1/x}, g^{1/y}$ and returns (u, v) with $v^y = u^x$, then there exists an "extractor" $\mathcal{E}$, which given same inputs as $\mathcal{A}$ and returns t such that $(g^{1/x})^t = u, (g^{1/y})^t = v$. Without loss of generality, we let*

$$\mathsf{Adv}^{\mathsf{KEA}}_{\mathcal{A},\mathcal{E}}(1^\lambda) = \Pr[\mathcal{E}(g^{1/x}, g^{1/y}, (g^{1/x})^t, (g^{1/y})^t) \neq t].$$

Specifically, KEA holds if for every adversary $\mathcal{A}$ there exists an extractor $\mathcal{E}$ and a negligible error function ν such that

$$\mathsf{Adv}^{\mathsf{KEA}}_{\mathcal{A},\mathcal{E}}(1^\lambda) \leqslant \nu(\lambda).$$

4. For simplicity of description, we denote $[\ell] := \{1, 2, \cdots, \ell\}$.

B.1. Security Models

We first describe the selective indistinguishability security experiment $\mathbf{EXP}^{\text{sIND-ATK}}_{\Pi,\mathcal{A}}(1^\lambda)$ between a challenger and a PPT adversary $\mathcal{A}$ in Figure 8. We define two security

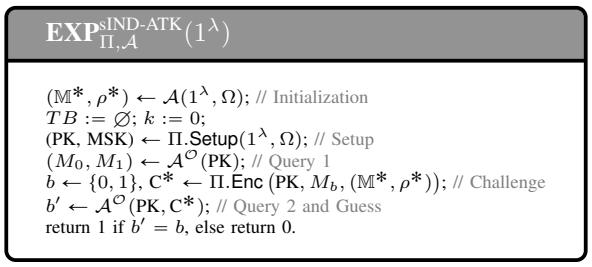


Figure 8: Experiment for ABE-2OD

models (i.e. ATK $\in$ {CDA, CHA}), which are different in the oracles allowed by $\mathcal{A}$.

$$\mathcal{O} = \begin{cases} \{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{CorruptDK}}, \mathcal{O}_{\mathsf{Transform2}}\} & \text{ATK = CDA;} \\ \{\mathcal{O}_{\mathsf{KeyGen}}, \mathcal{O}_{\mathsf{CorruptDK}}, \mathcal{O}_{\mathsf{CorruptHK}}, \mathcal{O}_{\mathsf{Dec}}\} & \text{ATK = CHA;} \end{cases}$$

sIND-CDA. To capture the adversary's ability, the sIND-CDA model follows the experiment $\mathbf{EXP}^{\text{sIND-CDA}}_{\Pi,\mathcal{A}}(1^\lambda)$ and allows $\mathcal{A}$ to query the set of oracle defined as above. Specifically, these oracles are described as follows.

- $\mathcal{O}_{\mathsf{KeyGen}(\text{MSK},\cdot)}$ is a key generation oracle that allows $\mathcal{A}$ to query an attribute set $A_k \subset \Omega$. For each query, it runs $\mathsf{KeyGen}(\text{MSK}, A_k)$ to obtain the key triple (TK_k, HK_k, DK_k) and responds $\mathcal{A}$ with TK_k. Besides, it records $TB[k] := (A_k, \text{TK}_k, \text{HK}_k, \text{DK}_k)$ and updates $k \leftarrow k+1$.
- $\mathcal{O}_{\mathsf{CorruptDK}(\cdot)}$ is a corruption oracle that allows $\mathcal{A}$ to query an index i. If $TB[i]$ is not defined, it outputs $\perp$; otherwise fetches $(A_i, \text{TK}_i, \text{HK}_i, \text{DK}_i)$ and forwards DK_i to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Transform2}(\cdot,\cdot)}$ is the second stage transformation oracle that allows $\mathcal{A}$ to query an index i and a partially transformed ciphertext PTC. If $TB[i]$ is not defined, it outputs $\perp$; otherwise, retrieves A_i and HK_i. If the query is in the second query phase and $A \vdash (\mathbb{M}^*, \rho^*)$, it forwards $\perp$ to $\mathcal{A}$; otherwise, sends the transformed result of $\mathsf{Transform2}(\text{HK}_i, \text{PTC})$ to $\mathcal{A}$.

Definition 5. *(sIND-CDA for ABE-2OD). An ABE-2OD scheme is sIND-CDA if for any PPT adversary $\mathcal{A}$, the following advantage is negligible.*

$$\mathsf{Adv}^{\text{sIND-CDA}}_{\Pi,\mathcal{A}}(1^\lambda) = \left| \Pr\left[\boldsymbol{EXP}^{\text{sIND-CDA}}_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2} \right|.$$

sIND-CHA. To capture the adversary's ability, the sIND-CHA model follows the experiment $\mathbf{EXP}^{\text{sIND-CHA}}_{\Pi,\mathcal{A}}(1^\lambda)$ and allows $\mathcal{A}$ to query the set of oracle defined as above. Specifically, these oracles are described as follows.

- $\mathcal{O}_{\mathsf{KeyGen}(\text{MSK},\cdot)}$ is a key generation oracle that is the same as in sIND-CDA.
- $\mathcal{O}_{\mathsf{CorruptHK}(\cdot)}$ is a corruption oracle that allows $\mathcal{A}$ to query an index i. If $TB[i]$ is not defined, it outputs $\perp$; otherwise fetches $(A_i, \text{TK}_i, \text{HK}_i, \text{DK}_i)$ and forwards HK_i to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{CorruptDK}(\cdot)}$ is a corruption oracle that allows $\mathcal{A}$ to query an index i. If $TB[i]$ is not defined or $A \vdash (\mathbb{M}^*, \rho^*)$, it outputs $\perp$; otherwise it fetches $(A_i, \text{TK}_i, \text{HK}_i, \text{DK}_i)$ from $TB[i]$ and forwards DK_i to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Dec}(\cdot,\cdot)}$ is a decryption oracle that allows $\mathcal{A}$ to query an index i and a transformed ciphertext TC. If $TB[i]$ is not defined, it outputs $\perp$; otherwise, retrieves A_i and DK_i and calculates $\text{Out} \leftarrow \mathsf{Dec}(\text{DK}_i, \text{TC})$. If the query is in the second query phase, it forwards $\mathcal{A}$ a special symbol **test** if $\text{Out} \in \{M_0, M_1\}$; otherwise, it sends Out to $\mathcal{A}$.

Definition 6. *(sIND-CHA for ABE-2OD). An ABE-2OD scheme is sIND-CHA if for any PPT adversary $\mathcal{A}$, the following advantage is negligible.*

$$\mathsf{Adv}^{\text{sIND-CHA}}_{\Pi,\mathcal{A}}(1^\lambda) = \left| \Pr\left[\boldsymbol{EXP}^{\text{sIND-CHA}}_{\Pi,\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2} \right|.$$

B.2. Proof of THEOREM 1

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ who has a non-negligible advantage in the game of sIND-CDA, we build a simulator $\mathcal{B}$ that calls $\mathcal{A}$ as a subroutine to break the Green's OABE scheme (i.e., Π_{OABE}). The simulation consists of six phases and the details are as follows.

Init. The simulator $\mathcal{B}$ runs a copy of $\mathcal{A}$. $\mathcal{A}$ determines an LSSS-style policy $(\mathbb{M}^*, \rho^*)$, which is then submitted by $\mathcal{B}$ for challenging.

Setup. Receiving a public parameter PK from Π_{OABE}'s challenger, $\mathcal{B}$ forwards PK to the adversary, where

$$\text{PK} = (g, e(g,g)^\alpha, g^a, H, H_1, H_2).$$

Note that MSK is unknown to $\mathcal{B}$.

Query Phase 1. $\mathcal{B}$ initializes $TB := \varnothing$, $k := 0$, and an H-list. Then $\mathcal{B}$ answers the queries from $\mathcal{A}$ as follows.

$\mathcal{O}_{\mathsf{KeyGen}(\text{MSK},\cdot)}$: Receiving an attribute set A as input, $\mathcal{B}$ forwards A to Π_{OABE}'s key generation oracle and then obtains the transformation key $\overline{\text{TK}}$ in Π_{OABE}, i.e., there exist $(K', L', \{K'_y\}_{y\in A})$ such that

$$\overline{\text{TK}} = \left(\bar{K} = (K')^{\frac{1}{\beta}}, \bar{L} = (L')^{\frac{1}{\beta}}, \{\bar{K}_y = (K'_y)^{\frac{1}{\beta}}\}_{x\in A}\right)$$

where β is also unknown to $\mathcal{B}$. Then $\mathcal{B}$ chooses random numbers $t'_1, t'_2, \gamma_1, \gamma_2 \in \mathbb{Z}_p$, and simulates as follows.

- If $A \not\vdash (\mathbb{M}^*, \rho^*)$, $\mathcal{B}$ first queries Π_{OABE}'s key generation oracle to obtain β. Then $\mathcal{B}$ extracts $(K', L', \{K'_y\}_{y\in A})$ from $\overline{\text{TK}}$. Then $\mathcal{B}$ calculates $\text{TK} = (\text{TK}_1, \text{TK}_2)$, where each TK_i ($i \in \{1,2\}$) is defined

$$K = \left(K' \cdot (g^a)^{t'_i}\right)^{\frac{1}{\gamma_i \beta}}, L = \left(L' \cdot g^{t'_i}\right)^{\frac{1}{\gamma_i \beta}},$$

$$\{K_y = \left(K'_y H_1(y)^{t'_i}\right)^{\frac{1}{\gamma_i \beta}}\}_{y\in A}.$$

Let $\bar{t}$ be the randomness hidden in $\overline{\text{TK}}$, the randomness in TK_i ($i \in \{1,2\}$) is $t_i = \bar{t} + t'_i$. Therefore it perfectly simulates the transformation key. Besides $\mathcal{B}$ sets $\text{HK} = (\gamma_1, \gamma_2)$ and $\text{DK} = \beta$.

- Otherwise, $A \vdash (\mathbb{M}^*, \rho^*)$, $\mathcal{B}$ cannot query Π_{OABE}'s key generation oracle to obtain β. Instead, $\mathcal{B}$ chooses a random β^* as the fake DK. $\mathcal{B}$ calculates TK = $(\text{TK}_1, \text{TK}_2)$, where each TK_i ($i \in \{1, 2\}$) is defined

$$K = \left(\bar{K} \cdot (g^a)^{t'_i}\right)^{\frac{1}{\gamma_i \beta^*}}, L = \left(\bar{L} \cdot g^{t'_i}\right)^{\frac{1}{\gamma_i \beta^*}},$$

$$\{K_y = \left(\bar{K}'_y H_1(y)^{t'_i}\right)^{\frac{1}{\gamma_i \beta^*}}\}_{y \in A}.$$

  Note that he randomness in TK_i ($i \in \{1, 2\}$) is $t_i = \bar{t} + \beta \cdot t'_i$. Besides $\mathcal{B}$ sets HK = (γ_1, γ_2) and DK = β^*. The pair (TK, HK, DK) is not well-formed, but the (TK, DK) is properly distributed from $\mathcal{A}$'s view.

Finally, $\mathcal{B}$ sets $TB[k] := \{A, \text{TK}, \text{HK}, \text{DK}\}$, $k \leftarrow k+1$, and forwards TK to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{CorruptDK}(\cdot)}$: Receiving an index i from $\mathcal{A}$, it outputs error symbol $\perp$ if $TB[i]$ is not defined; otherwise it fetches DK from $TB[i]$ and forwards DK to $\mathcal{A}$.

$\mathcal{G}_{\mathsf{Transform2}(\cdot,\cdot)}$: Receiving an index i and a partially transformed ciphertext PTC, $\mathcal{B}$ answers $\mathcal{A}$'s query as follows.

- If $A \not\vdash (\mathbb{M}^*, \rho^*)$, $\mathcal{B}$ fetches corresponding (TK, HK, DK) from TB. Since (TK, HK, DK) is a well-formed key tuple, $\mathcal{B}$ runs ABE-2DO's $\mathsf{Transform2}$ and sends the transformed result to $\mathcal{A}$.
- Otherwise, $A \vdash (\mathbb{M}^*, \rho^*)$, $\mathcal{B}$ does not maintain the well-formed key tuple in this case. $\mathcal{B}$ also fetches corresponding (TK, HK = (γ_1, γ_2), DK = β^*) from TB. Let PTC = (C, C', C_{P1}, C_{P2}). In sIND-CDA, the relation $C_{P1}^{\gamma_1} = C_{P2}^{\gamma_2}$ always holds. Without loss of generality, let

$$C_{P1} = \left(e(g,g)^{\frac{\alpha s}{\beta \gamma_1}}\right)^t, \quad C_{P2} = \left(e(g,g)^{\frac{\alpha s}{\beta \gamma_2}}\right)^t.$$

  $\mathcal{B}$ also generates a random cipher $\widehat{\text{C}}$ via setting the randomness $s = 1$ during encryption and uses the corresponding TK to generate a $\widehat{\text{PTC}} = (\widehat{C}, \widehat{C'}, \widehat{C_{P1}}, \widehat{C_{P2}})$ such that

$$\widehat{C_{P1}} = e(g,g)^{\frac{\alpha}{\beta \gamma_1}}, \quad \widehat{C_{P2}} = e(g,g)^{\frac{\alpha}{\beta \gamma_2}}.$$

  From KEA assumption, given $\widehat{\text{PTC}} = (C_{P1}, C_{P2}, \widehat{C_{P1}}, \widehat{C_{P2}})$, there exists an extractor that extract $s \cdot t$ from the tuple. $\mathcal{B}$ calculates $\overline{C''} = (e(g,g)^{\alpha})^{\frac{s \cdot t}{\beta^*}}$. Finally, $\mathcal{B}$ returns $(T = C, T' = C', T'' = \overline{C''})$.

Challenge. Receiving a message pair (M_0, M_1) from $\mathcal{A}$, $\mathcal{B}$ forwards M_0 and M_1 to Π_{OABE}'s challenger, which responds by a ciphertext C*. Finally it passes C* to $\mathcal{A}$. Note that the bit b of Π_{OABE}'s challenger is unknown to $\mathcal{B}$.

Query Phase 2. $\mathcal{B}$ continues to answer queries from $\mathcal{A}$ as in Query Phase 1, except that: in the $\mathcal{G}_{\mathsf{Transform2}(\cdot,\cdot)}$ query, if $A \vdash (\mathbb{M}^*, \rho^*)$, it forwards $\perp$ to $\mathcal{A}$.

Guess. Receiving a bit b' from $\mathcal{A}$, $\mathcal{B}$ eventually outputs the same bit b' as its guess.

This ends the simulation. In case $A \vdash (\mathbb{M}^*, \rho^*)$, $\mathcal{A}$ always generates a proper TC for $\mathcal{A}$. Obliviously, it perfectly simulates the game. We have

$$\mathsf{Adv}^{\text{sIND-CDA}}_{\Pi, \mathcal{A}} \leqslant \mathsf{Adv}^{\text{sIND-CPA}}_{\Pi_{\text{OABE}}, \mathcal{B}} + q_T \cdot \mathsf{Adv}^{\mathsf{KEA}}_{\mathcal{A}, \mathcal{E}},$$

where $\mathsf{Adv}^{\text{sIND-CPA}}_{\Pi_{\text{OABE}}, \mathcal{B}}$ is advantage of $\mathcal{B}$ wins Π_{OABE}'s CPA security game and q_T is the number of queries to the $\mathcal{G}_{\mathsf{Transform2}(\cdot,\cdot)}$ oracle. Therefore, the advantage of breaking the sIND-CDA is negligible, which completes the proof. □

B.3. Proof Sketch of THEOREM 2

Proof. ABE-2OD splits OABE's secret key into two parts: γ-part (γ_1, γ_2) and β-part. In case of corrupted HK attacks, we can view the β as the secret key in OABE, and the combination of TK and γ-part as the transformation key in OABE. Therefore, ABE-2OD's sIND-CHA security directly follows OABE's RCCA security. □

C. HR-ABE's Security Models and Its Proofs

C.1. The sIND-ColA model.

$\mathbf{EXP}^{\text{sIND-ColA}}_{\Pi^{\mathcal{G}}, \mathcal{A}}(1^\lambda)$

$(\mathbb{M}^*, \rho^*) \leftarrow \mathcal{A}(1^\lambda)$; //Initialize
$TB := \emptyset$;
$(\text{MPK}, \text{MSK}, \text{st}_0, \text{UL}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^\lambda, \Omega, T)$; //Setup
$(M_0, M_1, \text{st}_i) \leftarrow \mathcal{A}^{\mathcal{G}, \mathcal{O}}(\text{MPK}, \text{st}_0)$; //Phase 1
$\text{UL} = \text{UL} \backslash (T_{cor} \cap T_{sat})$; //Revoke
$b \leftarrow \{0, 1\}$;
$\text{C}^* \leftarrow \mathsf{Enc}(\text{MPK}, M_b, (\mathbb{M}^*, \rho^*))$; //Challenge
Let $\boldsymbol{st} = \{\text{st}_0, \text{st}_1, \cdots, \text{st}_i\}$;
$b' \leftarrow \mathcal{A}^{\mathcal{G}, \mathcal{O}}(\text{MPK}, \text{C}^*, \boldsymbol{st})$; //Phase 2
return 1 if $b' = b$, else return 0.

Figure 9: Experiment of the sIND-ColA for HR-ABE

Figure 9 presents an experiment between a challenger and an adversary $\mathcal{A}$ for defining selective indistinguishability against collusion attacks (sIND-ColA). The experiment $\mathbf{EXP}^{\text{sIND-ColA}}_{\Pi^{\mathcal{G}}, \mathcal{A}}$ allows the adversary $\mathcal{A}$ to query a set of oracle $\mathcal{O} = \{\mathcal{O}_{\mathsf{Join}}, \mathcal{O}_{\mathsf{Rev}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{Corrupt}}\}$ and $\mathcal{G} = \{\mathcal{G}_{\mathsf{Transform2}}\}$. Let A_{ID} denote the attribute set of DU ID. We define

$$T_{sat} := \{ID : ID \in \mathcal{U} \wedge A_{ID} \vdash (\mathbb{M}^*, \rho^*)\},$$
$$T_{cor} := \{ID : ID \in \mathcal{U} \wedge ID\text{'s DK was corrupted}\}.$$

$\mathcal{A}$ can adaptively query join oracle $\mathcal{O}_{\mathsf{Join}}$, revocation oracle $\mathcal{O}_{\mathsf{Rev}}$, and update oracle $\mathcal{O}_{\mathsf{Upd}}$ to determine the set of DUs that are in the latest UL at each time interval. The corruption oracle $\mathcal{O}_{\mathsf{Corrupt}}$ allows $\mathcal{A}$ to obtain the decryption key DK of a DU and $\mathcal{G}_{\mathsf{Transform2}}$ is the transformation oracle that inputs $\mathcal{A}$'s query (ID, PTC) and outputs the transformed ciphertext TC to $\mathcal{A}$. Specifically, these oracles are defined as follows.

- $\mathcal{O}_{\mathsf{Join}(\text{MSK}, \text{UL}, \cdot, \cdot)}$ is a join oracle that allows $\mathcal{A}$ to query an identity ID and an attribute set A. It runs $(\text{UL}, (\text{TK}, \text{RK}, \text{DK})) \leftarrow \mathsf{Join}(\text{MSK}, \text{UL}, ID, A)$, and sets $TB[ID] := (A, \text{TK}, \text{RK}, \text{DK})$ and returns TK to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Rev}(\text{UL}, \cdot)}$ is a revocation oracle that allows $\mathcal{A}$ to query an identity ID. For each query, it runs $\mathsf{Rev}(\text{UL}, ID)$ to obtain a new DU list UL.

- $\mathcal{O}_{\mathsf{Upd}(\mathrm{MSK},\mathrm{UL},\cdot,\cdot)}$ is a state update oracle that allows $\mathcal{A}$ to query a state st_{i-1}, and a time interval index i. For each query, it responds $\mathcal{A}$ with a new state st_i at i-th time interval, where $\mathsf{st}_i \leftarrow \mathsf{Update}(\mathrm{MSK}, \mathsf{st}_{i-1}, \mathrm{UL}, i)$.
- $\mathcal{O}_{\mathsf{Corrupt}(\cdot)}$ is a corruption oracle that allows $\mathcal{A}$ to query an identity ID. If $TB[ID]$ is not defined, it outputs error symbol $\perp$; otherwise it fetches $(A, \mathrm{TK}, \mathrm{RK}, \mathrm{DK}) \leftarrow TB[ID]$ and forwards DK to $\mathcal{A}$. If the query is in the second query phase and $A \vdash (\mathbb{M}^*, \rho^*)$, it outputs $\perp$.
- $\mathcal{G}_{\mathsf{Transform2}(\cdot,\cdot)}$ is the transformation oracle that allows $\mathcal{A}$ to query an ID and a PTC. It invokes $\mathsf{Transform2}$ and forwards the result to $\mathcal{A}$.

Definition 7. *(sIND-ColA for HR-ABE). An HR-ABE scheme is said to be sIND-ColA secure if for any PPT adversary $\mathcal{A}$, the following advantage is negligible.*

$$\mathsf{Adv}^{\textit{sIND-ColA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda) = \left|\Pr\left[\boldsymbol{EXP}^{\textit{sIND-ColA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}\right|.$$

C.2. The sIND-CTA model.

Besides sIND-ColA presented in Section 4.2, HR-ABE needs a new security model to reflect the following intuition.

Corrupted TEE attacks launched by any attacker have no help to obtain the actual message as long as unrevoked DUs' decryption keys still keep private.

$\mathbf{EXP}^{\mathrm{sIND\text{-}CTA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda)$

$(\mathbb{M}^*, \rho^*) \leftarrow \mathcal{A}(1^\lambda)$; //Initialize
$TB := \varnothing$;
$(\mathrm{MPK}, \mathrm{MSK}, \mathsf{st}_0, \mathrm{UL}) \leftarrow \mathsf{Setup}^{\mathcal{G}}(1^\lambda, \Omega, T)$; //Setup
$(M_0, M_1, \mathsf{st}_i) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathrm{MPK}, \mathsf{st}_0)$; //Phase 1
$b \leftarrow \{0, 1\}$;
$\mathrm{C}^* \leftarrow \mathsf{Enc}(\mathrm{MPK}, M_b, (\mathbb{M}^*, \rho^*))$; //Challenge
Let $\boldsymbol{st} = \{\mathsf{st}_0, \mathsf{st}_1, \cdots, \mathsf{st}_i\}$;
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathrm{MPK}, \mathrm{C}^*, \boldsymbol{st})$; //Phase 2
return 1 if $b' = b$, else return 0.

Figure 10: Experiment of the sIND-CTA for HR-ABE

Figure 10 presents the experiments $\mathbf{EXP}^{\mathrm{sIND\text{-}CTA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda)$ for HR-ABE for our new security model called selective indistinguishability against corrupted TEE attacks (sIND-CTA), which is also run between a challenger and an adversary. Compared to sIND-ColA, the differences include two aspects. On one hand, sIND-CTA allows $\mathcal{A}$ to corrupt revocation keys via corruption oracle whereas sIND-COlA allows $\mathcal{A}$ to obtain decryption keys; on the other hand, sIND-CTA allows $\mathcal{A}$ to query a decryption oracle while sIND-ColA allows $\mathcal{A}$ to query the transformation oracle. Specifically, sIND-CTA allows $\mathcal{A}$ to query a set of oracle $\mathcal{O} = \{\mathcal{O}_{\mathsf{Join}}, \mathcal{O}_{\mathsf{Rev}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}_{\mathsf{Corrupt}}, \mathcal{O}_{\mathsf{Dec}}\}$. While $\mathcal{O}_{\mathsf{Join}}$, $\mathcal{O}_{\mathsf{Rev}}$ and $\mathcal{O}_{\mathsf{Upd}}$ are same as in sIND-ColA, $\mathcal{O}_{\mathsf{Corrupt}}$ and $\mathcal{O}_{\mathsf{Dec}}$ in sIND-CTA are defined as follows.

- $\mathcal{O}_{\mathsf{Corrupt}(\cdot)}$ is a corruption oracle that allows $\mathcal{A}$ to query an identity ID. If $TB[ID]$ is not defined, it outputs $\perp$; otherwise it fetches $(A, \mathrm{TK}, \mathrm{RK}, \mathrm{DK}) \leftarrow TB[ID]$. If $A \nvdash (\mathbb{M}^*, \rho^*)$, it returns (RK, DK) to $\mathcal{A}$; otherwise, it returns (RK, $\perp$) to $\mathcal{A}$.
- $\mathcal{O}_{\mathsf{Dec}(\cdot,\cdot)}$ is a decryption oracle that allows $\mathcal{A}$ to query an identity ID, and a transformed ciphertext TC. If $TB[ID]$ is not defined, it outputs $\perp$; otherwise it fetches $(A, \mathrm{TK}, \mathrm{RK}, \mathrm{DK}) \leftarrow TB[ID]$, and returns the result of $\mathsf{Dec}(\mathrm{DK}, \mathrm{TC})$ to $\mathcal{A}$. If the query is in the second query phase and $\mathsf{Dec}(\mathrm{DK}, \mathrm{TC}) \in \{M_0, M_1\}$, it outputs a special symbol **test** to $\mathcal{A}$.

Definition 8. *(sIND-CTA for HR-ABE). An HR-ABE scheme achieves sIND-CTA if for any PPT adversary $\mathcal{A}$, the following advantage is negligible.*

$$\mathsf{Adv}^{\textit{sIND-CTA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda) = \left|\Pr\left[\boldsymbol{EXP}^{\textit{sIND-CTA}}_{\Pi^{\mathcal{G}},\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}\right|.$$

C.3. Proof of THEOREM 3

Proof. The proof applies the hybrid argument of games [31] and below we provide a proof sketch of THEOREM 3. We start the proof by defining three games: Game_0, Game_1, and Game_2. Let E_i denotes the event that $\mathcal{A}$ wins in Game_i. In detail, the two games are defined as follows.

- Game_0. This game is the original sIND-COlA game.
- Game_1. This game is the same as Game_0, except that if the adversary is able to provide a different state $\mathsf{st}^*_i = (\sigma^*, \mathrm{UL}^*_i, i^*)$ from that maintained by the challenger $\mathsf{st}_i = (\sigma, \mathrm{UL}_i, i)$ then the challenger outputs a random bit and aborts the game.
- Game_2. This game is the same as Game_1, except that the encryption of RKs (e.g., $\mathrm{E}(\mathrm{K_{TEE}}, \mathrm{RK})$) is replaced by random ciphertexts.

Claim 1. *If the signature scheme Π_{Sig} is EUF-CMA, the difference between the adversary's advantage in Game_0 and Game_1 is negligible.*

Proof sketch. Game_0 and Game_1 are the same unless a forgery event (i.e., the adversary produces a forged signature $\sigma^* \neq \sigma$ for a modified $\mathrm{UL}^*_i \neq \mathrm{UL}_i$, which passes the verification) occurs, given that the freshness of a valid UL_i is guaranteed by TEE's replay protection mechanism. Therefore, there exists a forger $\mathcal{F}$ such that

$$|\Pr[E_0] - \Pr[E_1]| \leqslant \mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\Pi_{Sig},\mathcal{F}}.$$

where $\mathsf{Adv}^{\mathrm{EUF\text{-}CMA}}_{\Pi_{Sig},\mathcal{F}}$ is the advantage of $\mathcal{F}$ breaking Π_{Sig}.

Claim 2. *If $\mathsf{Enc}(K_{TEE}, \cdot)$ (say Π_{AE}) is a secure authenticated encryption scheme [17], the difference between the adversary's advantage in Game_1 and Game_2 is negligible.*

Proof sketch. Suppose there exists an adversary that has a non-negligible advantage for distinguishing Game_1 and Game_2, it implies that the adversary can distinguish a real ciphertext and a random ciphertext, which contradicts the CCA security of Π_{AE}. Formally, there exists a simulator $\mathcal{B}_1$ such that

$$|\Pr[E_1] - \Pr[E_2]| = \mathsf{Adv}^{\mathrm{CCA}}_{\Pi_{AE},\mathcal{B}_1}.$$

where $\mathsf{Adv}^{\mathrm{CCA}}_{\Pi_{AE},\mathcal{B}_1}$ is the advantage of $\mathcal{B}_1$ breaking Π_{AE}.

Claim 3. *If the ABE-2OD scheme is sIND-CDA, the adversary's advantage in* Game_2 *is negligible.*

Proof. Suppose there exists a PPT adversary $\mathcal{A}$ who has a non-negligible advantage in Game_2, we build a simulator $\mathcal{B}$ that calls $\mathcal{A}$ as a subroutine to win sIND-CDA of the ABE-2OD scheme (i.e., Π_1). The simulation consists of six phases and the details are as follows.

Init. The simulator $\mathcal{B}$ runs a copy of $\mathcal{A}$. $\mathcal{A}$ determines a policy $(\mathbb{M}^*, \rho^*)$, which is submitted for challenging.

Setup. Receiving a public parameter PK_1 from Π_1's sIND-CDA challenger, $\mathcal{B}$ first generates a random symmetric key $\mathrm{K_{TEE}}$, and constructs an initialized DU list UL. Besides, $\mathcal{B}$ also runs a copy of the simulated signature scheme $(vk, sk) \leftarrow \Pi_{Sig}.\mathsf{KeyGen}(1^\lambda)$. Then $\mathcal{B}$ maintains the MSK $= (\boxdot, sk, \mathrm{K_{TEE}})$ privately, where '$\boxdot$' denotes unknown master secret key in Π_1. $\mathcal{B}$ also maintains variables $\mathrm{UL} = \varnothing$, and st_0, which is produced in a similar way as in oracle $\mathcal{O}_{\mathsf{Upd(MSK,UL,\cdot,\cdot)}}$. Finally, $\mathcal{B}$ forwards $(\mathrm{MPK}, \mathsf{st}_0)$ to the adversary, where $\mathrm{MPK} = (\mathrm{PK}_1, vk)$.

Query Phase 1. $\mathcal{B}$ initializes $TB = \varnothing$, $T_{cor} = \varnothing$, and $T_{sat} = \varnothing$. Then $\mathcal{B}$ answers the queries for $\mathcal{A}$ as follows.

$\mathcal{O}_{\mathsf{Join(MSK,UL,\cdot,\cdot)}}$: Receiving an identity ID and an attribute set A as input, $\mathcal{B}$ first queries Π_1's challenger with (ID, A) to obtain the transformation key TK_1. $\mathcal{B}$ corrupts the DK_1 via querying Π_1's $\mathcal{O}_{\mathsf{CorruptDK}}$ and records $TB[ID] := (A, \mathrm{TK}, \boxdot, \mathrm{DK})$. Besides, $\mathcal{B}$ forwards TK to $\mathcal{A}$ and sets $\mathrm{UL} \leftarrow \mathrm{UL} \cup \{ID, \mathrm{TK}, \mathsf{Enc}(\mathrm{K_{TEE}}, \mathbf{0})\}$, where $\mathsf{Enc}(\mathrm{K_{TEE}}, \mathbf{0})$ denotes a random ciphertext. Furthermore, $\mathcal{B}$ records $T_{sat} = T_{sat} \cup \{ID\}$ if $A \vdash (\mathbb{M}^*, \rho^*)$. Note that RK is unknown to $\mathcal{B}$ because $\mathcal{B}$ cannot corrupt the helper keys in Π_1's sIND-CDA game.

$\mathcal{O}_{\mathsf{Rev(UL,\cdot)}}$: Receiving an identity ID, $\mathcal{B}$ deletes the record associated to ID and sends the updated UL to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{Upd(MSK,UL,\cdot,\cdot)}}$: Receiving a state st_{i-1} and a time interval index i, $\mathcal{B}$ sets $\mathrm{UL}_i = \mathrm{UL}$ and signs a signature σ on UL_i and the time interval index i. Finally it forwards $\mathsf{st}_i = (\sigma, \mathrm{UL}_i, i)$ to $\mathcal{A}$.

$\mathcal{O}_{\mathsf{Corrupt(\cdot)}}$: Receiving an identity ID from $\mathcal{A}$, it outputs error symbol $\perp$ if $TB[ID]$ is not defined; otherwise it fetches DK from $TB[ID]$, records $T_{cor} = T_{cor} \cup \{ID\}$, and forwards DK to $\mathcal{A}$.

$\mathcal{G}_{\mathsf{Transform2(\cdot,\cdot)}}$: Receiving an identity ID and a partially transformed ciphertext PTC, $\mathcal{B}$ checks whether $ID \in$ UL. If not, $\mathcal{B}$ outputs $\perp$ to $\mathcal{A}$; otherwise, it submits PTC to Π_1's $\mathcal{O}_{\mathsf{Transform2}}$ oracle and forwards response Out to $\mathcal{A}$. Note that if $\mathsf{Out} = \perp$, it implies the PTC is a non-legitimate one.

Challenge. Receiving a message pair (M_0, M_1) from $\mathcal{A}$, $\mathcal{B}$ deletes all $ID \in T_{cor} \cap T_{sat}$ from UL and forwards the message pair to its challenger and obtains a ciphertext C^* under $(\mathbb{M}^*, \rho^*)$. Finally it passes C^* to $\mathcal{A}$.

Query Phase 2. $\mathcal{B}$ continues to answer queries from $\mathcal{A}$ by following the simulation in Query Phase 1, except that when answering $\mathcal{O}_{\mathsf{Corrupt(\cdot)}}$ queries, $\mathcal{B}$ returns $\perp$ for any queried ID with attribute set $A \vdash (\mathbb{M}^*, \rho^*)$.

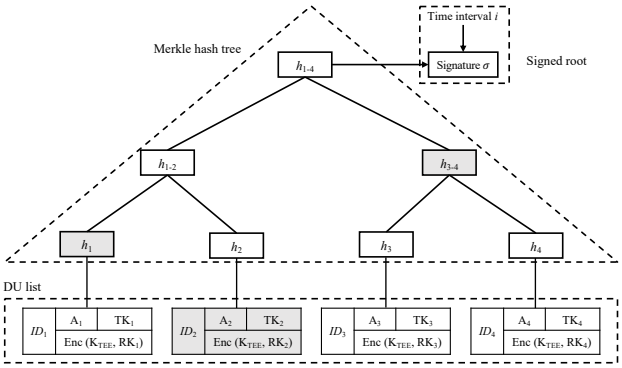


Figure 11: Merkle hash tree as a state

Guess. Receiving a bit b' from $\mathcal{A}$, $\mathcal{B}$ eventually outputs the same bit b' as its guess.

It completes the simulation perfectly. Therefore we have

$$\Pr[E_2] = \mathsf{Adv}^{\text{sIND-CDA}}_{\Pi_1,\mathcal{B}} + \frac{1}{2}$$

where $\mathsf{Adv}^{\text{sIND-CDA}}_{\Pi_1,\mathcal{B}}$ is the adversary's advantage in winning the sIND-CDA game of an ABE-2OD scheme.

Taking all claims together, we have that

$$\begin{aligned}\Pr[E_0] &\leqslant |\Pr[E_0] - \Pr[E_1]| + |\Pr[E_1] - \Pr[E_2]| + \Pr[E_2] \\ &= \mathsf{Adv}^{\text{EUF-CMA}}_{\Pi_{Sig},\mathcal{F}} + \mathsf{Adv}^{\text{CCA}}_{\Pi_{AE},\mathcal{B}_1} + \mathsf{Adv}^{\text{sIND-CDA}}_{\Pi_1,\mathcal{B}} + \frac{1}{2}.\end{aligned}$$

Therefore, the advantage of breaking sIND-COIA security is negligible. It completes the proof of THEOREM 3. □

D. sMHT as The State

sMHT, also called signed tree head (SHT), is a popular technology in Google's certificate transparency [22]. Figure 11 describes a toy example to exhibit how sMHT works. Let $H : \{0,1\}^* \mapsto \{0,1\}^\lambda$ be a cryptographic hash function. Given any DU list UL, MHT is built in a bottom-up manner, where the leaf nodes are hash values of original elements in UL, e.g., $h_1 = H(ID_1, \mathrm{A}_1, \mathrm{TK}_1, \mathsf{Enc}(\mathrm{K_{TEE}}, \mathrm{RK}_1))$, and the internal nodes are hash values of their children, e.g., $h_{1\text{-}2} = H(h_1, h_2)$. Instead of signing the whole DU list, now only the root node (i.e., $h_{1\text{-}4}$) is signed by the KGC to ensure integrity of the DU list. Meanwhile, a time interval index i is also embedded into the signature to ensure the freshness of the state, i.e., $\sigma \leftarrow \Pi_{Sig}.\mathsf{Sign}(sk, \langle h_{1\text{-}4}, i\rangle)$ for the given example. With the sMHT, the membership proof includes the siblings of the nodes located at the path from the leaf to the root node (e.g., the nodes highlighted by the gray box in Figure 11). In such a way, TEE only takes the logarithmic-sized proof from the state module as input and then completes the integrity checking in a bottom-up manner by verifying the hash values in the path and the signature of the root node. For example, TEE calculates the hash values $h_2 = H(ID_2, \mathrm{A}_2, \mathrm{TK}_2, \mathsf{Enc}(\mathrm{K_{TEE}}, \mathrm{RK}_2))$, $h_{1\text{-}2} = H(h_1, h_2)$, $h_{1\text{-}4} = H(h_{1\text{-}2}, h_{3\text{-}4})$ and verifies the signature by $\{0,1\} \leftarrow \Pi_{Sig}.\mathsf{Ver}(vk, \langle h_{1\text{-}4}, i\rangle, \sigma)$.

E. Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

E.1. Summary

The paper presents HR-ABE, a technique for revocation in attribute-based encryption that leverages the trusted execution environment of a machine to ensure that a revoked user is not able to access data that pre-dates their revocation without having to preform many costly re-encryptions. Instead, revoked users are flagged and known as revoked to the TEE. Data owner autonomy over data policies is maintained by splitting the ABE decryption key into a triple, including a revocation key, given to the TEE, a translation key, and a decryption key. The triple is used in a 2-stage outsourced ABE decryption mode to fulfil the revocation check function. The paper then shows the security and performance of this approach, and argues that it is a step forward in performance for ABE revocation.

E.2. Scientific Contributions

- Addresses a long-known Issue
- Provides a valuable step forward in an established field

E.3. Reasons for Acceptance

1) This paper addresses the problem of revocation in ABE in a novel way – through use of a TEE to eliminate ciphertext delegation and key rollover/update.
2) HR-ABE reduces the cost of revocation in cloud-based systems, and overall represents a more scalable approach to the adoption of ABE for confidentiality within such a system.