Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.073423

ARTICLE

LUAR: Lightweight and Universal Attribute Revocation Mechanism with SGX Assistance towards Applicable ABE Systems

Fei Tang[1,*], Ping Wang[1], Jiang Yu[1], Huihui Zhu[1], Mengxue Qin[1] and Ling Yang[2]

[1]The School of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China

[2]Chongqing Unicom Industrial Internet Co., Ltd., Chongqing, 401121, China

*Corresponding Author: Fei Tang. Email: tangfei@cqupt.edu.cn

Received: 17 September 2025; Accepted: 05 November 2025; Published: 12 January 2026

ABSTRACT: Attribute-Based Encryption (ABE) has emerged as a fundamental access control mechanism in data sharing, enabling data owners to define flexible access policies. A critical aspect of ABE is key revocation, which plays a pivotal role in maintaining security. However, existing key revocation mechanisms face two major challenges: (1) High overhead due to ciphertext and key updates, primarily stemming from the reliance on revocation lists during attribute revocation, which increases computation and communication costs. (2) Limited universality, as many attribute revocation mechanisms are tailored to specific ABE constructions, restricting their broader applicability. To address these challenges, we propose LUAR (Lightweight and Universal Attribute Revocation), a novel revocation mechanism that leverages Intel Software Guard Extensions (SGX) while minimizing its inherent limitations. Given SGX's constrained memory (≈90 MB in a personal computer) and susceptibility to side-channel attacks, we carefully manage its usage to reduce reliance while mitigating potential collusion risks between cloud service providers and users. To evaluate LUAR's lightweight and universality, we integrate it with the classic BSW07 scheme, which can be seamlessly replaced with other ABE constructions. Experimental results demonstrate that LUAR enables secure attribute revocation with low computation and communication overhead. The processing time within the SGX environment remains stable at approximately 55 ms, regardless of the complexity of access policies, ensuring no additional storage or computational burden on SGX. Compared to the Hardware-based Revocable Attribute-Based Encryption (HR-ABE) scheme (IEEE S&P 2024), LUAR incurs a slightly higher computational cost within SGX; however, the overall time from initiating a data request to obtaining plaintext is shorter. As access policies grow more complex, LUAR's advantages become increasingly evident, showcasing its superior efficiency and broader applicability.

KEYWORDS: Attribute-based encryption; attribute revocation; lightweight; universality

1 Introduction

In the digital era, the rapid advancement of cloud computing has brought data security and privacy protection to the forefront of both academic and industrial discussions. As enterprises and organizations increasingly migrate their data storage and processing to cloud environments, ensuring secure and efficient access control while maintaining data confidentiality has become a pressing challenge [1–3].

Attribute-Based Encryption (ABE), introduced by Sahai and Waters [4], is widely recognized for its fine-grained access control capabilities. Unlike Identity-Based Encryption (IBE) [5], ABE binds ciphertexts and secret keys to sets of attributes rather than unique identities. Decryption is permitted only if the attributes

embedded in a user's key satisfy the predefined access policy. This design makes ABE particularly well-suited for applications including secure cloud computing [6–8], healthcare data sharing [9,10], Internet of Things [11], and social networks [12–14].

Despite its advantages, ABE suffers from a long-standing challenge in attribute revocation—the ability to prevent access for users whose attributes have been revoked. Existing revocation strategies typically fall into two categories: direct [15–17] and indirect [18–20]. In both cases, the revocation mechanism either imposes high computation/storage overhead or sacrifices forward security by allowing revoked users to access previously obtained ciphertexts.

To mitigate these problems, researchers have explored ciphertext update mechanisms [21] as well as hardware-based solutions [15,22] using Trusted Execution Environments (TEEs), particularly Intel SGX [23]. However, traditional TEE-based ABE revocation schemes often rely on storing revocation lists or transformation keys inside the enclave, quickly exceeding SGX's 90MB memory limit and creating a central point of failure.

To address these challenges, we propose LUAR (Lightweight and Universal Attribute Revocation), a novel revocation framework that integrates Ciphertext-Policy ABE (CP-ABE) with Intel SGX in a lightweight and generalizable manner. LUAR features a timestamp-based key expiration mechanism, lightweight Merkle-proof-based verification, and a ciphertext blinding strategy that protects sensitive information from enclave exposure. These innovations reduce reliance on SGX memory, eliminate the need for key/ciphertext updates, and enable seamless integration with a variety of ABE schemes.

Our key contributions are as follows.

- **Lightweight.** LUAR introduces an efficient attribute revocation mechanism that avoids re-encrypting all cloud-stored files upon attribute changes or key revocation. Even under Cloud Service Provider (CSP)-user collusion, LUAR ensures secure and low-cost revocation without a heavy computational burden. It minimizes reliance on complex encryption, revocation list maintenance, and conversion key management within the TEE, alleviating the 90 MB SGX memory constraint in a personal computer.
- **Universal.** LUAR is designed for flexibility and scalability. While implemented with the classic BSW07 scheme [24], it supports seamless integration with other ABE variants. It maintains low computation and communication overhead, making it suitable for large-scale or resource-constrained settings. LUAR also supports horizontal scalability in multi-user cloud environments, offering a practical solution for fine-grained access control.
- **LUAR implementation.** Experiments show stable SGX processing time (≈55 ms) regardless of policy complexity. Although LUAR has slightly higher SGX-side computation than HR-ABE, it achieves significantly lower overall latency, especially as policy complexity grows. The source code is available at https://github.com/Aw-creat/SGX_ABE (accessed on 10 Oct 2025).

2 Related Work

Direct and Indirect Revocation. ABE systems is often categorized into direct and indirect methods. In direct revocation, such as in Wang et al. [16] and FDR-CP-ABE [17], a revocation list is embedded during encryption, allowing data owners to control access. However, this increases ciphertext size and imposes frequent updates. In indirect revocation, seen in schemes like RABE [19] and RS-CPABE-ASP [20], the trusted authority periodically updates keys for legitimate users. These approaches often require both key and ciphertext renewal, leading to high communication overhead. In summary, while these schemes can prevent unauthorized access, they introduce significant operational complexity and communication costs, particularly in dynamic or large-scale environments.

Blockchain-Based Revocation. Blockchain-assisted ABE revocation has emerged as an approach to provide decentralized and tamper-resistant revocation tracking. For example, some schemes utilize smart contracts to maintain and query revocation status [25–29]. This design enhances transparency and auditability without relying on a single trusted entity. However, practical deployment faces limitations such as high gas costs, transaction latency, and synchronization issues between blockchain and off-chain access control. Moreover, integrating blockchain with real-time ABE operations remains challenging due to the mismatch in performance requirements. In summary, blockchain-based revocation offers strong decentralization and transparency, but its overhead and complexity limit its applicability to time-sensitive or large-scale systems.

TEE-Based Revocation. TEE-assisted revocation leverages secure hardware like Intel SGX to enforce attribute revocation with integrity guarantees. For instance, HR-ABE [15] uses SGX to perform epoch-based checks with Merkle proofs, ensuring that revoked users cannot bypass access control even under adversarial settings. Similarly, Fan et al. [22] generate ephemeral keys inside SGX per access attempt to isolate secrets. However, both rely heavily on SGX for storage and computation, which is constrained by limited enclave memory (e.g., 90 MB EPC in SGX) and is vulnerable to side-channel attacks [30,31]. Furthermore, the need to store revocation state or perform complex computations within SGX leads to scalability bottlenecks. In summary, while TEE-based revocation enhances security through hardware-enforced isolation, many designs suffer from over-reliance on enclave resources and reduced efficiency in high-load or multi-user scenarios.

3 Preliminaries

3.1 CP-ABE

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is a type of ABE technique that enables DOs to encrypt their data in such a way that only users possessing specific attributes can decrypt it. Instead of defining particular recipients, DOs specify an access structure that dictates the required attribute combinations for decryption. For example, an access structure such as "$(A\ AND\ B)\ OR\ C$" implies that users possessing both attributes A and B, or possessing attribute C, can decrypt the data. This approach enables fine-grained access control without knowing users' identities.

The fundamental concept of CP-ABE is embedding the access policy directly into the encryption process. A user can decrypt a ciphertext only if their attribute set satisfies the predefined access policy. The CP-ABE scheme consists of four main steps: Setup, Key Generation, Encryption, and Decryption. The detailed algorithm is presented below. (1) $(PK, MSK) \leftarrow \mathsf{CP\text{-}ABE.Setup}(1^{\lambda})$. The algorithm takes the security parameter λ as input and generates the public key PK and master secret key MSK. (2) $U_K \leftarrow \mathsf{CP\text{-}ABE.KeyGen}(\mathsf{MSK}, \mathsf{Attrs})$. The algorithm takes the master secret key MSK and user's attribute set $Attrs$ as input and generates the user's key U_K. (3) $C \leftarrow \mathsf{CP\text{-}ABE.Encrypt}(\mathsf{PK}, \mathsf{M}, \mathsf{Policy})$. The algorithm inputs the public key PK, message M, and an access policy $Policy$, and outputs the ciphertext C. (4) $M \leftarrow \mathsf{CP\text{-}ABE.Decrypt}(\mathsf{C}, \mathsf{U_K})$. The algorithm inputs the ciphertext C and user's key U_K, and outputs the plaintext M provided the user's attributes satisfy the specified access policy.

3.2 Intel SGX

Intel Software Guard Extensions (SGX) [23] is a hardware-level security technology developed by Intel to enhance data privacy and protection in modern computer systems. SGX enables applications to execute code and process sensitive data within a secure, hardware-isolated region known as an enclave. These enclaves provide a robust defense against interference from malicious software, compromised operating systems, and other external threats.

A key aspect of SGX is the interaction between enclave and non-enclave code, facilitated by two essential mechanisms: Enclave Call (Ecall) and Out Call (Ocall). Ecall allows untrusted external code to invoke functions inside the enclave, while Ocall enables enclave code to call functions outside its secure boundary. These mechanisms are fundamental to ensuring seamless communication between trusted and untrusted components while maintaining the security guarantees of SGX. Besides, SGX supports remote attestation that allows users to verify the authenticity and integrity of an enclave in a remote environment. This process establishes a secure communication channel between interacting parties. Through remote attestation, SGX guarantees that only verified and uncompromised environments are authorized to participate in sensitive operations.

3.3 Optimized SGX Utilization

In existing SGX-based schemes, there is excessive reliance on SGX as a black-box execution environment, leading to critical challenges: (1) SGX acts as a superuser, requiring unconditional trust in its security; (2) it must store revocation lists or numerous transformation keys, increasing storage burden; and (3) it handles both decryption and re-encryption internally, causing high computational overhead and potential plaintext leakage via side-channel attacks. To address these issues, we propose an optimized approach that reduces SGX dependency while preserving security and efficiency. The key improvements are as follows.

Minimized trust dependency. SGX no longer performs decryption or re-encryption operations. Instead, it ensures the proper execution of internal programs while preventing plaintext exposure. Even if the CSP gains access to intermediate data during SGX processing, it remains unable to derive the original plaintext, thereby reducing the trust burden placed on SGX.

Optimized storage management. SGX no longer maintains user revocation lists or a large number of transformation keys. Instead, it retains only a single transformation key for removing blind factors in ciphertext. This significantly reduces memory consumption and mitigates system performance bottlenecks.

Lightweight computation. SGX is solely responsible for verifying the legitimacy of user requests, ensuring that keys are valid and not revoked. Once verification is complete, SGX uses the transformation key to remove blind factors from ciphertext without engaging in complex encryption or decryption, significantly improving execution efficiency.

4 LUAR Architecture

4.1 LUAR Definition

As illustrated in Fig. 1, the system architecture comprises four key entities: Software Guard Extensions (SGX), Cloud Service Provider (CSP), Data Owner (DO), and Data Requestor (DR). (1) **SGX:** provides a protected execution environment for generating the system's master key, public key, and user keys. It plays a crucial role in verifying user legitimacy and transforming ciphertext to remove blind factors while ensuring data privacy. Importantly, SGX does not directly expose plaintext data, thus maintaining security even in untrusted environments. (2) **CSP:** is responsible for storing encrypted files uploaded by users. Beyond storage, the CSP ensures secure file retention and provides encrypted data to SGX when authorized access is granted. (3) **DO:** is responsible for encrypting and uploading data to the CSP. By defining access policies during encryption, the DO ensures that only authorized users can access the data. Additionally, blind factors can be embedded in the ciphertext to enhance security, obfuscating data to prevent unauthorized decryption. (4) **DR:** is the entity requesting data from the cloud. To access data, the DR submits a request to the CSP, retrieves the encrypted file, and collaborates with SGX to decrypt it using its local user key. Note that trusted

authorities (e.g., government or enterprise identity providers) are responsible for issuing user attributes, but are considered external to the LUAR design and are not involved in the revocation or decryption workflow.

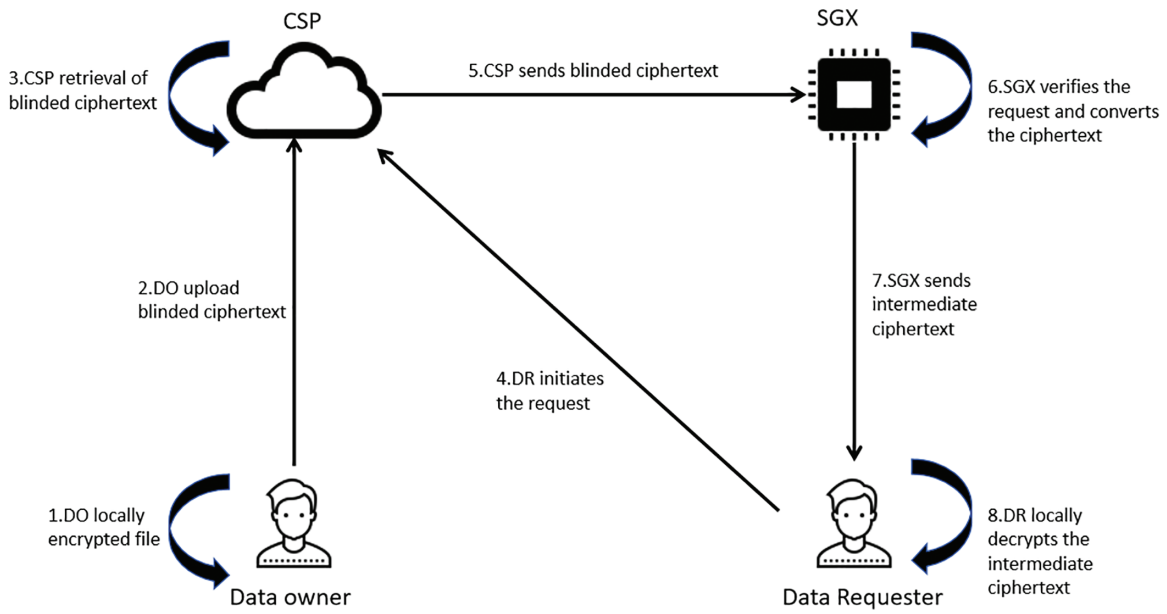

Figure 1: System architecture

4.2 LUAR Overview

The proposed mechanism operates through the following five key phases.

System Initialization. SGX generates the system master key and public key. The master key is securely stored within the SGX as a critical component for subsequent user key generation, while the public key is made available on the CSP for use in data encryption and access control.

User Key Generation. Trusted institutions, such as government agencies, banks, and official organizations, provide valid user attributes to the SGX environment. Upon receiving a request, SGX generates the corresponding user key, embedding an expiration date to facilitate lightweight revocation and request verification. Unlike traditional CP-ABE schemes, this method enhances security by ensuring that expired or revoked keys cannot be used for decryption. Once the key is generated, it is securely transmitted to the user via a remote attestation-based secure channel.

Encryption. Data encryption is carried out in two stages. In Stage 1, the DO generates a symmetric key to encrypt the file and then encrypts the symmetric key based on a selected access policy. In Stage 2, the DO incorporates a blind factor into the ciphertext to generate a blinded ciphertext, which is then uploaded to the CSP for secure storage.

SGX Verification and Transformation. When the DR initiates a data access request, the CSP retrieves the corresponding blinded ciphertext and forwards it, along with the request details, to the SGX. SGX first checks whether the key's expiration date is earlier than its internal clock time; if expired, the request is rejected. Next, SGX verifies the authenticity of the expiration date using the time-binding factor in the DR's key and checks whether the key has been revoked. If the request is deemed valid, SGX removes the blind factor from the ciphertext to obtain an intermediate ciphertext, which is then returned to DR. This process preserves data confidentiality and integrity while enforcing strict access control.

User Local Decryption. The DR utilizes their private key to decrypt the intermediate ciphertext and obtain the symmetric key. Finally, the symmetric key is used to decrypt the encrypted file, restoring the plaintext data.

5 LUAR Details

We provide a comprehensive implementation phase of LUAR and the flowchart is as shown in Fig. 2. The relevant symbols and descriptions are listed in Table 1.

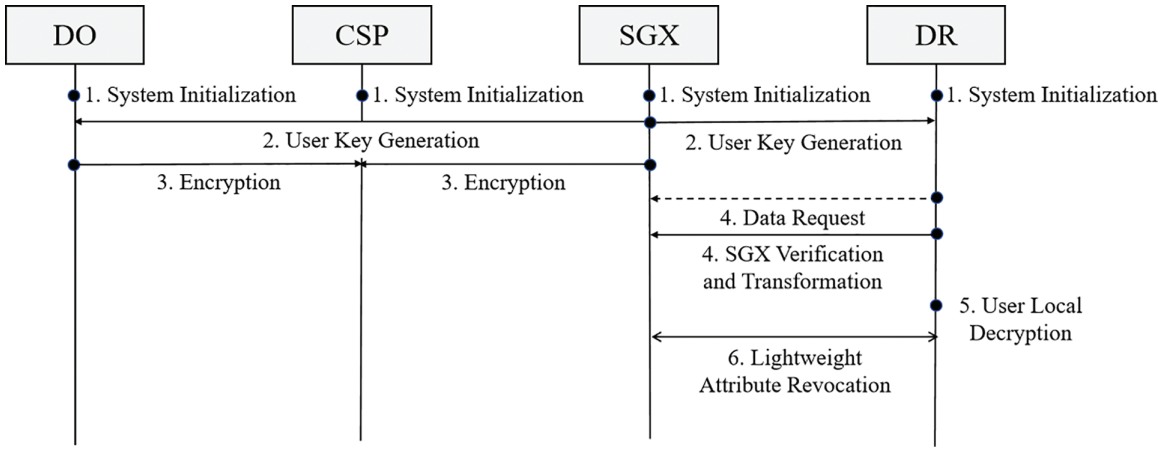

Figure 2: The flowchart of the LUAR framework

Table 1: Main symbols

Symbols	Descriptions
λ	Security parameter that determines the cryptographic strength of the system.
$\mathcal{U}$	The set of all possible attributes.
γ	A randomly selected parameter used to bind the user's key with its expiration time.
MSK	Master system key, consisting of $\alpha, \beta, \gamma, g_2^{\alpha}$.
PK	Public system key, including $g_1, h_1 = g_2^{\beta}, h_2 = g_1^{\gamma}, e(g_1, g_2)^{\alpha}$.
P	Proof of eligibility, indicating whether the user's attributes have been revoked.
$\mathcal{A}$	The set of attributes associated with a user.
τ_{expire}	Expiration time of the user's private key.
SK	User's private key, required for decrypting ciphertext after SGX transformation.
$\mathcal{T}$	Access policy tree defining user decryption permissions.
M	LSSS matrix is used to represent the access policy.
M_i	The i-th row of the LSSS matrix M.
$\tau_{current}$	Current time of the SGX internal clock, used for key expiration validation.
AES_KeyGen	AES symmetric key generation function.

(Continued)

Table 1 (continued)

Symbols	Descriptions
AES_Encrypt	AES encryption function that uses a symmetric key to encrypt files.
AES_Decrypt	AES decryption function that uses a symmetric key to decrypt files.

5.1 System Initialization

The initialization phase lays the cryptographic foundation for LUAR and ensures the secure deployment of critical components. During this phase, the code responsible for verifying user requests and removing the blinding factor is securely embedded into the SGX enclave. Upon launch, SGX generates a unique identifier to establish its trusted identity. Users can verify the integrity and authenticity of SGX via Intel's remote attestation mechanism. The complete initialization phase consists of four steps. The details are described below and presented in Algorithm 1.

Algorithm 1: System initialization

Input: Security Parameter λ

Output: System Master Key *MSK*, System Public Key *PK*

Step 1 Bilinear Groups Generation

Initialize groups $G_1 \neq G_2$, a prime order p, generators $g_1 \in G_1$, $g_2 \in G_2$

Initialize bilinear pairing $e : G_1 \times G_2 \rightarrow G_T$

Run $(G_1, G_2, G_T, p, g_1, g_2, e) = \mathsf{GroupGen}(\lambda)$

Step 2 LUAR CP-ABE Initialization

Run $(PK, MSK) \leftarrow \mathsf{LUAR\_Setup}(1^\lambda)$

Step 3 System Keys Generation

Select random numbers $\alpha, \beta, \gamma \overset{R}{\leftarrow} \mathbb{Z}_p^*$

Generate LUAR elements in *MSK* are β, γ

Generate LUAR elements in *PK* are $h_1 = g_2^\beta,\ h_2 = g_1^\gamma$

Step 4 Hash Function Definition

Define $H : \{0,1\}^* \rightarrow G_2$

Define $H(x) = g_2^{a_x}$, $a_x \in \mathbb{Z}_p$

5.2 User Key Generation

User key generation is performed within the SGX enclave. Unlike traditional CP-ABE schemes, LUAR integrates an expiration timestamp into each user's key, enabling efficient and lightweight attribute revocation while maintaining system security. The detailed process of user key generation is outlined in Algorithm 2.

Algorithm 2: User key generation

Input: Attribute Set $\mathcal{A}$, Expiration Time τ_{expire}, Master Key *MSK*

Output: User Key *SK*

Step 1 Expiration Time Key Component

Select a random number $t \overset{R}{\leftarrow} \mathbb{Z}_p^*$

Compute $K_1 = g_1^t \in G_1$

(Continued)

Algorithm 2 (continued)

3: Compute $K_\tau = H(ExpireTime \parallel \tau_{expire} \parallel nonce)^t = g_2^{a_\tau t} \in G_2$, where *nonce* is a unique identifier
Step 2 LUAR CP-ABE User Key Component
4: Run $SK = \mathsf{LUAR\_KeyGen}(MSK, \mathcal{A})$
5: Output *SK* includes Expiration Time Key Component

5.3 Encryption

Unlike traditional CP-ABE encryption schemes, LUAR incorporates a ciphertext blinding mechanism to mitigate potential collusion between the CSP and malicious users. To achieve this, the encryption process is divided into three distinct steps: Advanced Encryption Standard (AES) Encryption, CP-ABE Encryption, and ciphertext Blinding. This design ensures that even if a malicious user obtains the ciphertext from the CSP, they cannot decrypt it using an expired key. The detailed steps of the data encryption process are described in Algorithm 3.

Algorithm 3: Encryption

Input: Plaintext *F*, Access Policy Tree $\mathcal{T}$ (LSSS Matrix *M*), System Public Key *PK*
Output: Blinded Ciphertext *CT*
Step 1 AES Encryption
1: Run $S = \mathsf{AES\_KeyGen}()$
2: Run $CT(F) = \mathsf{AES\_Encrypt}(\mathsf{S}, \mathsf{F})$
Step 2 CP-ABE Encryption
3: Run $CT = \mathsf{LUAR\_Encrypt}(\mathsf{PK}, \mathcal{T})$
4: Select a random number $s \overset{R}{\leftarrow} \mathbb{Z}_p^*$
5: Compute $\mathbf{v} = (s, v_2, \ldots, v_n) \in \mathbb{Z}_p^n$
6: Compute $C_0 = S \cdot e(g_1, g_2)^{\alpha s}$
Step 3 Ciphertext Blinding
7: Get $h_1 = g_2^{\beta}$ from the public key *PK*
8: Compute $r \overset{R}{\leftarrow} \mathbb{Z}_p^*$, $C_0' = C_0 \cdot e(g_1, g_2)^{\beta rs}$, $\tilde{C}_1 = g_2^{rs} \in G_1$

5.4 SGX Verification and Transformation

In our scheme, SGX is primarily utilized for permission verification and blind factor removal. The design carefully accounts for SGX's memory constraints and potential side-channel threats. Unlike traditional black-box models, SGX functions in a quasi-white-box manner, meaning that even if the CSP can monitor the execution inside SGX and manipulate the inputs and outputs, it still cannot compromise the plaintext. The detailed process of the SGX verification and transformation phase is presented in Algorithm 4.

5.5 User Local Decryption

After receiving the intermediate ciphertext returned by SGX, the DR uses their private key *SK* to decrypt it. If the user's attributes meet the access policy requirements, they will successfully decrypt the plaintext; otherwise, the decryption will fail. The algorithm flow for the user's local decryption phase is as described in Algorithm 5.

5.6 Lightweight Attribute Revocation

This mechanism includes two types of revocation: passive revocation and active revocation, each requiring a distinct handling strategy. Passive revocation (User key expiration). Key expiration is the most common form of key invalidation in practical applications. In this passive revocation scenario, since the DO embeds a blinding factor into the ciphertext during encryption, even if the DR colludes with the CSP, they cannot recover the plaintext. Once the DR's key expires, it will fail the timestamp verification phase performed within SGX. This effectively addresses the issue of key expiration without requiring additional intervention. In other words, even if the CSP can access intermediate data during SGX execution, LUAR prevents both CSP and DR from using expired keys to decrypt data.

Algorithm 4: SGX verification and transformation

Input: Blinded Ciphertext CT, Time Component of Private Key SK, SGX Time τ_{current}, Qualification Proof P
Output: Intermediate Ciphertext CT_{inter}

Step 1 Timestamp Verification
1: **if** $\tau_{current} \leq \tau_{expire}$ **then**
2: compute $e(K_\tau, h_2) = e(g_2^{a_\tau t}, g_1^{\gamma}) = e(g_2, g_1)^{a_\tau t\gamma}$
3: compute $e\left(H(\tau_{expire} \parallel nonce), K_1^{\gamma}\right) = e(g_2^{a_\tau}, g_1^{\gamma t}) = e(g_2, g_1)^{a_\tau t\gamma}$
4: **end if**
Step 2 User Revocation Verification
5: Constructs a Merkle tree based on P (Refer to Section 5.6 for details)
Step 3 Transformation Factor Generation
6: Compute $T = e(g_1^{\beta}, \tilde{C}_1) = e(g_1^{\beta}, g_2^{rs}) = e(g_1, g_2)^{\beta rs}$
Step 4 Blind Factor Removal
7: Compute $C_0'' = \frac{C_0'}{T} = \frac{S \cdot e(g_1,g_2)^{\alpha s} \cdot e(g_1,g_2)^{\beta rs}}{e(g_1,g_2)^{\beta rs}} = S \cdot e(g_1, g_2)^{\alpha s}$
8: Compute de-blinded intermediate ciphertext CT_{inter}

Algorithm 5: User local decryption

Input: Intermediate Ciphertext CT_{inter}, Private Key SK
Output: Plaintext F

Step 1 LUAR CP-ABE Decryption
1: compute $CT(F) = \mathsf{LUAR\_Decrypt}(\mathsf{CT_{inter}}, \mathsf{SK})$
Step 2 AES Decryption
2: Compute $F = \mathsf{AES\_Decrypt}(\mathsf{S}, \mathsf{CT(F)})$

Active revocation (User attribute revocation). Active revocation occurs when a user's key is still valid, but the associated attributes have been changed or revoked. In such cases, the authorized institution must update the user's attributes and re-execute the key generation algorithm within SGX. A new user key is issued, and the DR proceeds with the standard decryption process when requesting data. To prevent revoked users from decrypting data with outdated keys, our scheme employs a Merkle tree to support active revocation without transmitting the entire DR list. Instead, only a logarithmic-size ($O(\log n)$) path hash is needed to verify membership, significantly reducing SGX's storage and computation overhead. When attributes are updated, the authorized institution reconstructs the Merkle tree based on the latest attribute set, as illustrated in Fig. 3.

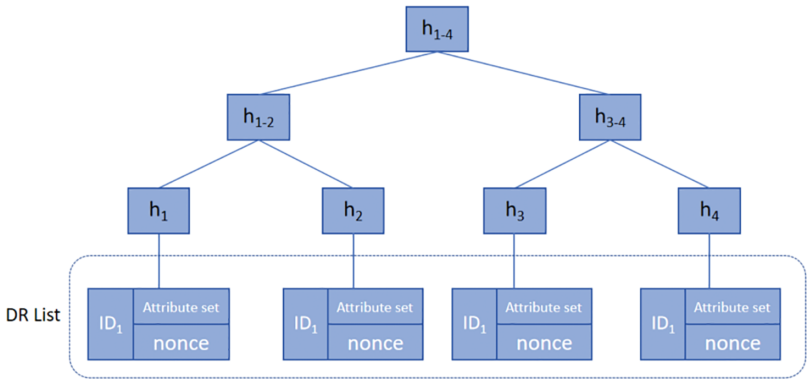

Figure 3: Merkle tree-based revocation list construction

After constructing the Merkle tree, the root hash h_{1-4} is generated and sent to SGX via remote attestation. If a revoked DR requests data, SGX receives a logarithmic-size path proof for the DR's ID, recomputes hashes from leaf to root, and checks the result against the signed root. A revoked DR fails this check. For example, for user ID1, parse the qualification proof P to obtain the path $h_1, h_2, h3-4$. SGX computes $h'1-2 = H(h_1, h_2)$ and then $h'1-4 = H(h'1-2, h3-4)$, finally verifying integrity via $0,1 \leftarrow \mathsf{Verify}(h'1-4, h1-4)$. A result of 0 means failure (revoked); 1 means success and decryption proceeds. Thus, even with an outdated key, a revoked DR learns nothing from the blinded ciphertext, preserving LUAR's security.

6 LUAR+BSW07

LUAR is designed to be a universal enhancement layer that integrates seamlessly with standard CP-ABE frameworks. In this section, we briefly describe how LUAR is embedded into the widely adopted BSW07 scheme [24], and focus on the added logic of timestamp binding, ciphertext blinding, and SGX-assisted transformation. LUAR consists of four core algorithms: LUAR_Setup, LUAR_KeyGen, LUAR_Encrypt, and LUAR_Decrypt. Below, we present LUAR's extensions and how they interface with the standard BSW07 flow.

6.1 Setup

The primary function of LUAR_Setup is to generate the master secret key MSK and the public key PK based on the BSW07 scheme. The complete construction of MSK and PK is as follows. (1) Select random values $\alpha, \beta, \gamma \xleftarrow{R} \mathbb{Z}_p^*$ as the foundational parameters for key generation. (2) The resulting outputs are system master secret key $MSK = (\alpha, \beta, \gamma, g_2^\alpha)$ and system public key $PK = \left(g_1,\ h_1 = g_2^\beta,\ h_2 = g_1^\gamma,\ e(g_1, g_2)^\alpha\right)$.

6.2 Key Generation with Timestamp Binding

The primary function of LUAR_KeyGen s to generate the user's private key SK. Unlike CP-ABE.KeyGen, LUAR appends a timestamp commitment $(K_\tau, \tau_{expire}, nonce)$ to the user's secret key. The detailed construction process is as follows. (1) A random value $t \xleftarrow{R} \mathbb{Z}_p^*$ is selected to generate the user's private key. (2) The core components are $K_0 = g_2^\alpha \cdot h_1^t = g_2^{\alpha+\beta t} \in G_2$, and $K_1 = g_1^t \in G_1$. (3) For each attribute $x \in \mathcal{A}$, the attribute components are given by $K_x = H(x)^t = g_2^{a_x t} \in G_2$. (4) The timestamp component is calculated as $K_\tau = H(ExpireTime \parallel \tau_{expire} \parallel nonce)^t = g_2^{a_\tau t} \in G_2$, where $nonce$ is generated by the SGX enclave and is to be unique per issuance. (5) The private key SK is constructed as $SK = \left(K_0,\ K_1,\ \{K_x\},\ K_\tau,\ \tau_{expire},\ nonce\right)$.

6.3 Encryption

The primary function of LUAR_Encrypt is to encrypt files. LUAR can be directly applied without modifying the underlying design of CP-ABE.Encrypt. The encryption process begins by selecting a random number $s \xleftarrow{R} \mathbb{Z}_p^*, v = (s, v_2, \ldots, v_n) \in \mathbb{Z}_p^n$, which are used to construct the ciphertext. The detailed construction is as follows. (1) The core ciphertext components are then computed as $C_0 = S \cdot e(g_1, g_2)^{\alpha s}$ and $C_1 = g_1^s \in G_1$. (2) For each row i of the policy matrix, the policy components are generated as $C_{i,1} = g_1^{M_i \cdot v}$ and $C_{i,2} = H(\rho(i))^{M_i \cdot v} \cdot h_1^{-s} = g_2^{a_{\rho(i)} M_i \cdot v - \beta s} \in G_2$. (3) The complete ciphertext is given by $CT = \left(C_0,\ C_1,\ \{C_{i,1}, C_{i,2}\} \right)$.

6.4 Decryption

The primary function of LUAR_Decrypt is to decrypt files. LUAR can be applied directly without modifying the underlying design of CP-ABE.Decrypt. The decryption process is as follows. (1) For each attribute node that satisfies the policy, the decryption of the leaf i is performed using the formula $DecryptNode(i) = \frac{e(C_{i,1}, K_x)}{e(C_{i,2}, K_1)} = \frac{e(g_1^{M_i \cdot v}, g_2^{a_x t})}{e(g_2^{a_{\rho(i)} M_i \cdot v - \beta s}, g_1^t)}$. The numerator expands to $e(g_1, g_2)^{a_x t M_i \cdot v}$, and the denominator expands to $e(g_2, g_1)^{t(a_{\rho(i)} M_i \cdot v - \beta s)} = e(g_1, g_2)^{-t(a_{\rho(i)} M_i \cdot v - \beta s)}$. Combining the results, if $\rho(i) = x \in \mathcal{A}$, we get $DecryptNode(i) = e(g_1, g_2)^{a_x t M_i \cdot v + t(a_x M_i \cdot v - \beta s)}$. (2) The decryption parameters are combined via Lagrange interpolation. The minimum covering set I is chosen with coefficients ω_i satisfying $\sum_{i \in I} \omega_i M_i = (1, 0, \ldots, 0)$. The product term is calculated as $\prod_{i \in I} DecryptNode(i)^{\omega_i} = \prod_{i \in I} \left(e(g_1, g_2)^{\beta s t} \right)^{\omega_i} = e(g_1, g_2)^{\beta s t \sum \omega_i} = e(g_1, g_2)^{\beta s t}$. (3) The symmetric key is recovered by computing $D = \frac{e(C_1, K_0)}{\prod_{i \in I} DecryptNode(i)^{\omega_i}} = \frac{e(g_1^s, g_2^{\alpha + \beta t})}{e(g_1, g_2)^{\beta s t}} = e(g_1, g_2)^{s(\alpha + \beta t)} \cdot e(g_1, g_2)^{-\beta s t} = e(g_1, g_2)^{\alpha s}$. The decryption output is $S = \frac{C_0}{D} = \frac{S \cdot e(g_1, g_2)^{\alpha s}}{e(g_1, g_2)^{\alpha s}} = S$.

Generality and Integration Flexibility. One of LUAR's core strengths lies in its generality—the ability to integrate seamlessly with a wide range of CP-ABE schemes without modifying their fundamental operations. LUAR achieves this by treating itself as a modular revocation overlay layer that interfaces with standard ABE procedures. It appends lightweight timestamp-related components to user keys and delegates revocation checks to a trusted SGX enclave, without interfering with the encryption or decryption algorithms of the underlying scheme. This design is particularly compatible with bilinear pairing-based ABE schemes that rely on linear secret-sharing structures, such as BSW07. However, integration with non-standard ABE schemes (e.g., lattice-based or those using non-linear access policies) may require slight adaptation. Overall, LUAR's abstraction of revocation logic enables a plug-and-play mechanism that enhances ABE frameworks with minimal implementation overhead and broad applicability, universality of ABE systems without compromising their original functionality or security assumptions.

7 Security Analysis

7.1 Security Model Definition

The primary objective of this scheme is to provide a secure cloud data sharing solution that addresses high attribute revocation costs and SGX overload through the use of attribute-based cryptography and SGX encoding. We make the following threat assumptions about the system entities: The CSP may engage in malicious activities, attempting to extract sensitive information from cloud data, including potentially accessing intermediate execution states of SGX. Additionally, there may be collusion between the CSP and users, where the CSP sends blinded ciphertexts directly to users with revoked attributes, allowing them to decrypt files using outdated keys. And the DO is assumed to be fully trusted.

The security of LUAR relies on standard cryptographic assumptions and the security properties of trusted execution environments. We formalize the security proof through security game definitions and polynomial-time reductions. Specifically, we define the Indistinguishability under Chosen-Plaintext Attack (IND-CPA) security game between the challenger C and the adversary A.

System Setup: The challenger C runs the initialization algorithm to generate the master public key PK and master secret key (MSK), and sends the public key PK to the adversary A.

Query Phase 1: During this phase, A can adaptively query the following oracles. (1) Key Generation Oracle $O_{\text{KeyGen}}(A, \tau)$: Given an attribute set A and expiration time τ, this oracle returns the corresponding private key SK. (2) Decryption Oracle $O_{\text{Decrypt}}(CT)$: It decrypts arbitrary ciphertexts, excluding blind factors.

Challenge Phase: In this phase, A submits two equal-length messages M_0 and M_1 along with an access policy $\mathcal{T}^*$, such that none of the key attribute sets queried by A satisfy $\mathcal{T}^*$. The challenger C randomly selects $b \overset{R}{\leftarrow} \{0,1\}$ and generates a challenge ciphertext CT^* based on M_b.

Query Phase 2: A retains access to the oracles but may not query keys satisfying $\mathcal{T}^*$.

Guess: A outputs a guess b'. The advantage of the adversary is $\text{Adv}_A = \left|\Pr[b' = b] - \frac{1}{2}\right|$.

7.2 Security Proof

Theorem 1 (IND-CPA Security). Assuming the Decisional Bilinear Diffie-Hellman (DBDH) problem is hard in groups (G_1, G_2, G_T), and that the SGX execution environment satisfies remote attestation and memory safety, LUAR achieves IND-CPA security under the random oracle model.

Game 0: Standard IND-CPA game.

Game 1: Modify challenge ciphertext by replacing $e(g_1, g_2)^{\alpha s}$ with random value $\tilde{Z} \overset{R}{\leftarrow} G_T$. If A can distinguish Game 0 and Game 1, we construct algorithm B to solve DBDH problem. Given DBDH tuple $(g_1, g_1^a, g_1^b, g_1^c, Z)$, algorithm B sets as $h_1 = g_1^b$, $h_2 = g_1^\gamma$, $e(g_1, g_2)^\alpha = e(g_1^a, g_1^b)$. The challenge ciphertext now contains $CT_0'' = M_b \cdot Z$. If $Z = e(g_1, g_2)^{abc}$, it simulates Game 0; otherwise, it simulates Game 1. A's distinguishing advantage directly translates to the DBDH-solving advantage.

Game 2: Introduce a blind factor with $\tilde{C}_1 = g_2^{rs}$. If A can decrypt using expired keys, they must compute $e(g_1, g_2)^{\beta rs}$, which is equivalent to solving the discrete logarithm problem with negligible advantage.

Game 3: SGX verifies timestamp through a hash binding equation such that $e(K_\tau, h_2) = e(H(\tau), K_1^\gamma)$. The collision resistance of H ensures the non-forgeability of the timestamp. The probability of a successful forgery is bounded by Adv_A^{CR}. Final advantage bound: $\text{Adv}_A^{\text{IND-CPA}} \leq \text{Adv}_B^{\text{DBDH}} + \text{Adv}_A^{\text{DL}} + \text{Adv}_A^{\text{CR}} + \text{negl}(\lambda)$.

Theorem 2 (Collusion Resistance). For any polynomial-time adversary A controlling at most $N-1$ malicious users, the advantage of their collusion attack satisfies: $\text{Adv}_A^{\text{Collusion}} \leq \frac{p}{N} + \text{negl}(\lambda)$, where p is the group order. Each private key contains an independent random $t_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$. The non-combinability of user-specific parameters t_i in the key components ensures that: $K_0^{(i)} = g_2^{\alpha+\beta t_i}$, $K_x^{(i)} = H(x)^{t_i}$. For any user set S, the combined keys $\prod_{i\in S}(K_0^{(i)})^{c_i}$ must satisfy the equation: $\sum_{i\in S} c_i(\alpha + \beta t_i) \equiv \alpha + \beta t^* \pmod{p}$. This requires solving the system: $\sum c_i = 1$ and $\sum c_i t_i = t^*$. The success probability for unknown c_i is bounded by $\frac{1}{p}$.

7.3 More Discusstion

Side-Channel Risk Mitigation. SGX enclaves provide strong memory isolation yet remain vulnerable to side channels (e.g., cache leakage, speculative execution) [30,31]. LUAR reduces the attack surface by keeping sensitive plaintext and heavy cryptography outside SGX. The enclave is limited to timestamp verification,

Merkle-path checking, and blind-factor removal over intermediate data. Even if execution patterns are observed, blinded ciphertext prevents disclosure of plaintext or keys. Standard hardening (e.g., access-pattern randomization, noise) can further strengthen LUAR, though full treatment is beyond this paper.

Trusted Time Assumption. LUAR's passive revocation compares enclave time $T_{current}$ with key's expiration T_{expire}, requiring a reliable monotonic source. This is provided by SGX's trusted-time application programming interface via the platform service enclave, resistant to host tampering. Where unavailable, a remote trusted time server can supply signed timestamps verified by the enclave. Either approach keeps time checks trustworthy under an adversarial operating system.

8 Experiment

In this section, we evaluate the performance and benefits of the proposed scheme. We implemented our solution and two comparison schemes on a physical machine running 64-bit Ubuntu 20.04 LTS, equipped with an Intel Core i7-12700H (12 cores, 20 threads) and 16 GB memory. Experiments were conducted in SGX simulation mode to eliminate network I/O impact, and operation times were recorded. Only one SGX group was deployed in this setup. In real-world scenarios, multiple SGX groups can be horizontally scaled to handle different attribute combinations, operating independently without interference.

8.1 Fixed Access Policy and Fixed Data Size

This experiment focuses on evaluating the runtime overhead of the proposed scheme. We conducted 30 independent trials and averaged the results. The access policy used for testing is defined as: $\mathcal{T} = (A1)\ AND\ (B2\ OR\ B3)\ AND\ (C2\ OR\ C3)$. The plaintext data size was fixed at 1000 bytes. The average time taken for each stage is summarized in Table 2.

Table 2: Average runtime overhead of our scheme

Phase	Time (ms)
System initialization	325.3
User key generation	103.6
Encryption	135.6
SGX verification and transformation	53.2
User local decryption	126.8

To clarify performance characteristics, we analyzed time-critical operations in each stage, excluding network I/O. During initialization, most time (≈247.5 ms) was spent on bilinear group generation, while constructing the master secret key (MSK) and public key (PK) took only 35.7 ms. In user key generation, most time went to building attribute components, about 54.5 ms. For the encryption phase, the primary overhead came from constructing the CP-ABE ciphertext components and the ciphertext blinding process, which took 24.3 and 16.5 ms, respectively. During the SGX verification and transformation stage, timestamp verification accounted for the bulk of the time, approximately 23.7 ms. In the final stage of user-side decryption, attribute-based decryption was the main time-consuming operation, taking roughly 95.3 ms.

It is worth noting that the total runtime within the SGX was only 53.2 ms. While our scheme incurs higher computational costs compared to symmetric encryption methods, it provides significantly stronger security guarantees. Importantly, overall runtime remains within an acceptable range, demonstrating that our security model can effectively meet user demands without compromising operational efficiency.

8.2 Fixed Access Policy and Variable Data Size

In this experiment, we continue to use the fixed access policy: $\mathcal{T} = (A1)$ *AND* $(B2\ OR\ B3)$ *AND* $(C2\ OR\ C3)$. Plaintext messages of varying lengths were selected for testing. Each test was repeated 30 times, and the average values were recorded. The average runtime for each phase is presented in Table 3.

Table 3: Average runtime overhead of our scheme

Phase time (ms)	1000 bytes	2000 bytes	5000 bytes	10000 bytes
System initialization	325.3	326.9	330.2	326.8
User key generation	103.6	108.5	107.6	105.8
Encryption	135.6	139.5	146.8	157.2
SGX verification and transformation	53.2	56.8	58.7	55.9
User local decryption	126.8	130.9	137.4	148.6

To visualize the performance trends, the data were plotted as a bar chart, as shown in Fig. 4. From the chart, it is evident that the SGX verification and transformation stage, the potential system bottleneck, does not exhibit a strong correlation with ciphertext size. Although larger ciphertext sizes slightly impact the efficiency of local encryption and decryption, the increase in time consumption is marginal. Overall, the scheme's performance remains unaffected by the size of the encrypted data. In conclusion, the experimental results indicate that the runtime overhead of our scheme scales well with increasing data sizes. The ciphertext size has a minimal impact on overall performance, and no performance bottleneck was observed in the SGX-related operations.

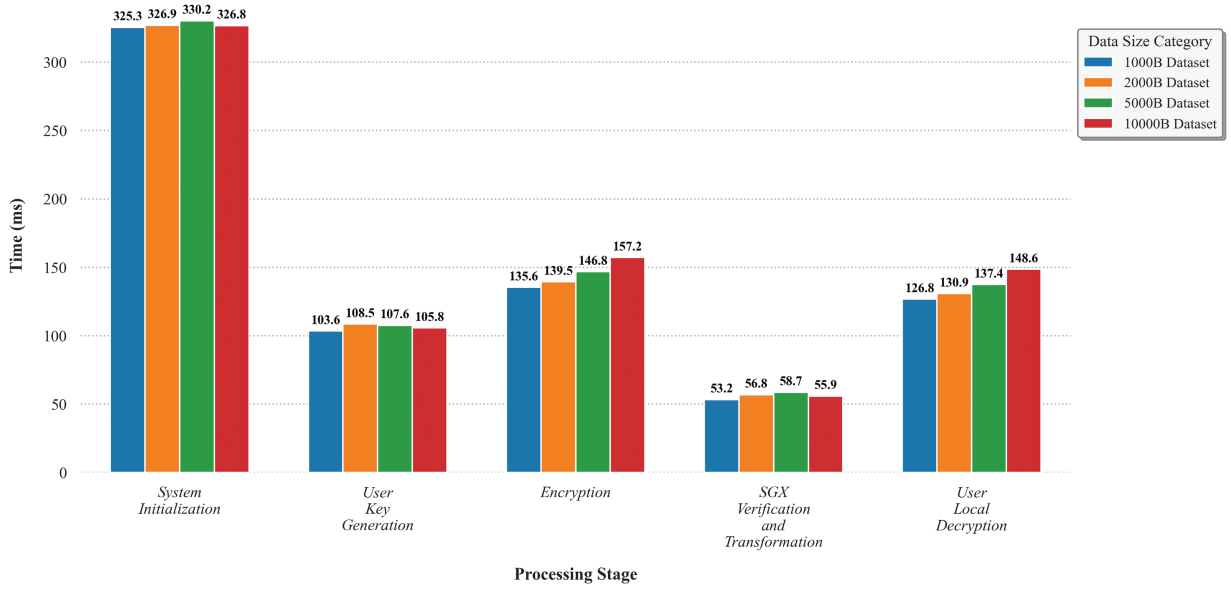

Figure 4: The average runtime overhead of our scheme

8.3 Variable Access Policy and Fixed Data Size

We evaluate variation in response time, from the moment the DR initiates a request to the time the plaintext is obtained, as the complexity of the access policy increases. The access policy is of the form

$\mathcal{T}$ = (*A*1 *AND A*2 *AND A*3 *AND B*1 *AND B*2 . . .). The complexity is increased by appending additional attributes using logical *AND* operations. We compare our scheme with two representative approaches: the HR-ABE scheme and the scheme proposed by Fan et al. [22], designed for secure data sharing and low-cost attribute revocation.

Comparison with the HR-ABE [15] scheme. The HR-ABE scheme implements attribute revocation using a Merkle tree-based revocation list. Similarly, our scheme also utilizes the Merkle tree; however, it introduces a significant optimization: in the common case of key expiration, no additional operations are required to prevent DRs from decrypting data using expired keys. Only in cases of active revocation does our scheme follow the same process as HR-ABE. Additionally, HR-ABE uses a key-splitting mechanism, which requires the CSP to process the access policy twice for a single ciphertext, once per conversion key. This results in higher processing overhead compared to our scheme.

From a broader perspective, HR-ABE adopts a malicious server model, embedding two time-related parameters in each ciphertext to ensure provable security even if the cloud colludes with revoked users. While this enhances security guarantees, it comes at the cost of increased cryptographic complexity and ciphertext size. In contrast, our scheme assumes a semi-honest server model—reasonable in regulated or auditable real-world deployments, allowing LUAR to achieve a simpler and more efficient design.

During decryption, HR-ABE delegates most of the policy-layer processing to the CSP, while SGX merely removes the blinding factor from the ciphertext. In contrast, our scheme uses SGX to verify timestamps and revocation status and to generate conversion factors for removing the blinding factor. As illustrated in Fig. 5, the SGX time in HR-ABE is negligible, whereas in our scheme, it remains consistently around 55 ms, independent of policy size. Although our scheme incurs slightly higher SGX overhead, the total time from DR request to plaintext is shorter, and this advantage becomes more pronounced as policy complexity increases.

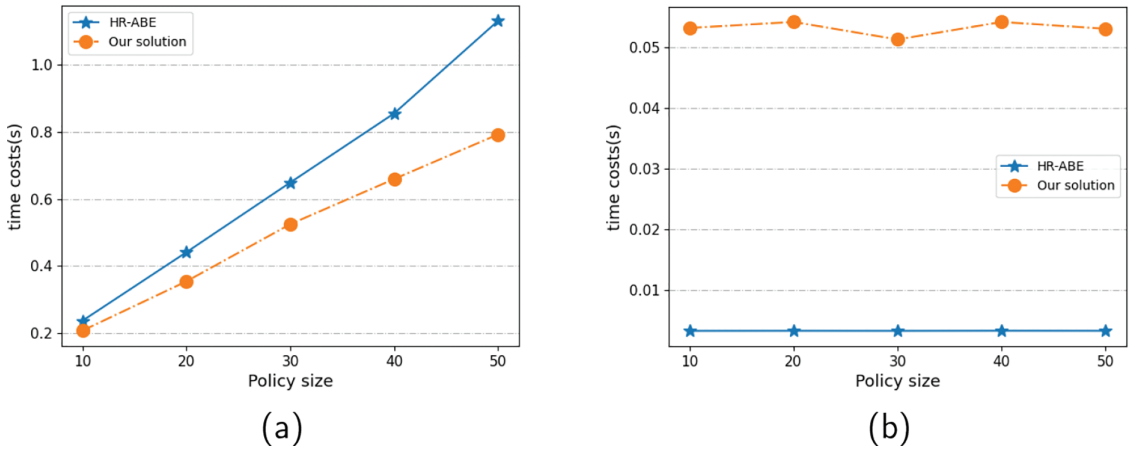

Figure 5: Compared to HR-ABE, (**a**) the total processing time and (**b**) the SGX processing time, vary with the policy length

In summary, LUAR adopts a semi-honest server model instead of HR-ABE's malicious one, enabling a more lightweight and efficient design. Furthermore, unlike HR-ABE, LUAR considers generalizability as a key design goal, supporting seamless integration with various ABE schemes, thus offering broader applicability beyond specific cryptographic constructions.

Comparison with the Fan et al. [22] scheme. Fan et al.'s scheme utilizes SGX to generate a one-time CP-ABE key per user request and performs the entire decryption operation within SGX. This design maintains a revocation list inside SGX and generates ephemeral keys on demand, an approach that rapidly

consumes enclave memory and results in latency bottlenecks under frequent access scenarios. In contrast, our approach avoids storing revocation state or transformation keys within the enclave. Instead, it uses lightweight Merkle proofs combined with timestamp-based expiration to validate revocation, significantly reducing SGX memory usage and computation overhead.

Our approach offloads most of the cryptographic burden from SGX, limiting its role to lightweight tasks such as verifying the legitimacy of requests and removing ciphertext blinding. As shown in Fig. 6a, the time difference between our scheme and Fan et al.'s scheme increases with the growth of the access policy. Fan et al.'s approach requires SGX to both generate keys and decrypt ciphertexts, which leads to a growing SGX processing burden. As depicted in Fig. 6b, their SGX processing time increases with policy size, whereas our scheme maintains a stable SGX processing time of around 55 ms. These results demonstrate that while Fan et al.'s scheme suffers from scalability issues within SGX, our approach achieves stable and low SGX overhead even as the access policy becomes more complex. Given SGX's constrained computational resources, our design is more practical and scalable for cloud-based systems supporting large user populations.

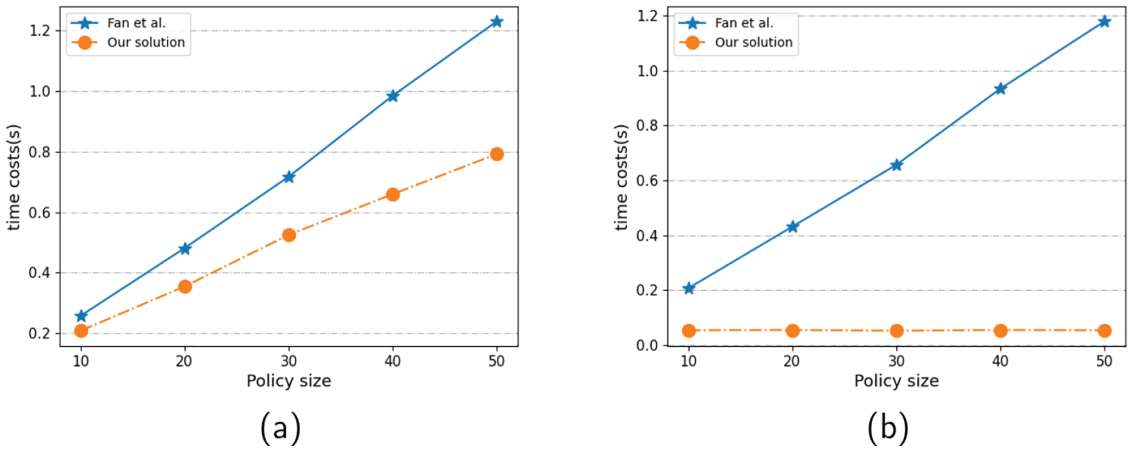

Figure 6: Compared to Fan et al.'s scheme, (**a**) the total processing time and (**b**) the SGX processing time, vary with the policy length

To conclude, by leveraging a lightweight verification mechanism and relaxing the collusion-resistance assumption, LUAR minimizes SGX workload while preserving security under the semi-honest model. Moreover, in contrast to Fan et al.'s scheme, which is tightly coupled with a specific ABE configuration, LUAR emphasizes universality, enabling broader adaptability across different CP-ABE frameworks.

8.4 Overhead Analysis

In addition to runtime performance, LUAR is specifically designed to minimize both communication overhead and SGX enclave memory consumption—two critical factors for practical deployment in TEE-assisted attribute-based encryption systems.

Communication Overhead. LUAR introduces a lightweight Verification and Transformation phase within the SGX enclave, which verifies the data requester's legitimacy and removes the timestamp-based blinding factor from the ciphertext. After this transformation, the resulting ciphertext becomes equivalent to a standard CP-ABE ciphertext and can be directly decrypted by the data requester using their private key. The only additional communication introduced by LUAR is between the cloud storage server and the enclave—both managed by the CSP. As a result, from the perspectives of the data owner and data requester, LUAR does not incur any extra communication overhead compared to traditional ABE-based schemes. For example, the

transformed ciphertext length remains at 331 bytes, identical to a conventional BSW07 ciphertext. The Merkle proof passed to SGX during revocation checking typically ranges between 600 and 800 bytes, depending on the tree depth, which is negligible relative to the ciphertext or data file size.

SGX Memory Consumption. LUAR is designed with resource efficiency in mind. Each Verification and Transformation operation inside the enclave requires only 3 pairing computations, 1 map-to-point operation, and 2 exponentiations. Although HR-ABE performs fewer enclave operations, specifically 3 exponentiations, it relies on two time-based blinding components in each ciphertext, shifting additional computational workload to the CSP side. Specifically, HR-ABE demands $(2 + 4|l|)$ pairing operations and $2|l|$ exponentiations on the CSP side, where $|l|$ is the number of attributes associated with the requester. In contrast, LUAR uses a single timestamp and a more streamlined blinding model, resulting in better total runtime efficiency. As shown in our experimental results, LUAR maintains a stable SGX processing time of approximately 55 ms, while HR-ABE's enclave time is negligible (≈2 ms) but its overall latency is higher due to heavier CSP-side computations. In terms of enclave storage, LUAR further reduces overhead by storing only a Merkle tree root (≈32 bytes) and a single master secret key inside SGX. In comparison, HR-ABE stores both a Merkle root and two master keys, increasing its memory footprint. This difference is especially important under the strict 90 MB limitation of Intel SGX's Enclave Page Cache, and becomes a scalability bottleneck when serving many users concurrently.

8.5 Deployment and Scalability Considerations

Multi-enclave scale-out. LUAR's enclave logic is stateless and lightweight (timestamp check, Merkle-path verification, blind-factor removal), so requests can be sharded across multiple SGX enclaves without inter-enclave dependency (already noted as feasible in our prototype discussion).

Bounded per-request enclave time. The SGX verification/transformation stage averages ≈53.2 ms and remains stable across settings, supporting high concurrency under horizontal scaling.

Network considerations. Our evaluation isolates compute by running in simulation mode (excluding network I/O). In deployment, end-to-end latency will add communication overhead, while the enclave's constant-time transformation remains the fixed core cost.

Revocation at scale. LUAR adopts a Merkle-tree revocation structure; each request carries an $O(logn)$ proof, keeping bandwidth and verification costs modest as the user base grows.

Acknowledgement: The authors would like to thank the School of Computer Science and Technology and the School of Cyber Security and Information Law at Chongqing University of Posts and Telecommunications for administrative and technical support, as well as the computing resources provided during this study.

Funding Statement: The authors acknowledge support from the National Key Research and Development Program of China (Grant No. 2021YFF0704102), the Chongqing Education Commission Key Project of Science and Technology Research (Grant No. KJZD-K202400610), and the Chongqing Natural Science Foundation General Project (Grant No. CSTB2025NSCQ-GPX1263).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Fei Tang and Ling Yang; methodology, Fei Tang, Ping Wang, and Jiang Yu; software, Jiang Yu, Huihui Zhu, and Mengxue Qin; validation, Ping Wang, Jiang Yu, and Huihui Zhu; formal analysis, Fei Tang, Ping Wang, and Jiang Yu; investigation, Jiang Yu and Huihui Zhu; resources, Mengxue Qin and Ling Yang; data curation, Ping Wang and Jiang Yu; writing—original draft preparation, Ping Wang and Jiang Yu; writing—review and editing, Fei Tang, Huihui Zhu, and Ling Yang; visualization, Jiang Yu and Ping Wang; supervision, Fei Tang and Ling Yang; project administration, Fei Tang; funding acquisition, Fei Tang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data and code that support the findings of this study are available from the corresponding author (Fei Tang) upon reasonable request. No personally identifiable information is included in the shared materials.

Ethics Approval: This study does not involve human participants or animals and therefore did not require ethics approval. No personally identifiable information was processed.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Chen Y, Zheng Q, Yan Z, Liu D. QShield: protecting outsourced cloud data queries with multi-user access control based on SGX. IEEE Trans Parallel Distributed Syst. 2021;32(2):485–99. doi:10.1109/tpds.2020.3024880.
2. Ghayvat H, Pandya S, Bhattacharya P, Zuhair M, Rashid M, Hakak S, et al. CP-BDHCA: blockchain-based confidentiality-privacy preserving big data scheme for healthcare clouds and applications. IEEE J Biomed Health Inform. 2022;26(5):1937–48. doi:10.1109/jbhi.2021.3097237.
3. Shen J, Zhou T, He D, Zhang Y, Sun X, Xiang Y. Block design-based key agreement for group data sharing in cloud computing. IEEE Trans Dependable Secur Comput. 2019;16(6):996–1010. doi:10.1109/tdsc.2017.2725953.
4. Sahai A, Waters B. Fuzzy identity-based encryption. In: Advances in Cryptology—EUROCRYPT 2005. Vol. 3494. Cham, Switzerland: Springer; 2005. p. 457–73 doi:10.1007/11426639_27.
5. Boneh D, Franklin MK. Identity-based encryption from the weil pairing. In: Advances in Cryptology—CRYPTO 2001. Vol. 2139. Cham, Switzerland: Springer; 2001. p. 213–29.
6. Hwang YW, Lee IY. A study on CP-ABE based data sharing system that provides signature-based verifiable outsourcing. In: 2021 International Conference on Advanced Enterprise Information System (AEIS). Piscataway, NJ, USA: IEEE; 2021. p. 1–5.
7. Wang J, Huang C, Xiong NN, Wang J. Blocked linear secret sharing scheme for scalable attribute based encryption in manageable cloud storage system. Inf Sci. 2018;424(9):1–26. doi:10.1016/j.ins.2017.09.032.
8. Xie M, Ruan Y, Hong H, Shao J. A CP-ABE scheme based on multi-authority in hybrid clouds for mobile devices. Future Gener Comput Syst. 2021;121(5):114–22. doi:10.1016/j.future.2021.03.021.
9. Quan G, Yao Z, Chen L, Fang Y, Zhu W, Si X, et al. A trusted medical data sharing framework for edge computing leveraging blockchain and outsourced computation. Heliyon. 2023;9(12):e22542. doi:10.1016/j.heliyon.2023.e22542.
10. Li F, Liu K, Zhang L, Huang S, Wu Q. EHRChain: a blockchain-based EHR system using attribute-based and homomorphic cryptosystem. IEEE Trans Serv Comput. 2022;15(5):2755–65. doi:10.1109/tsc.2021.3078119.
11. Li J, Zhang Y, Ning J, Huang X, Poh GS, Wang D. Attribute based encryption with privacy protection and accountability for CloudIoT. IEEE Trans Cloud Comput. 2022;10(2):762–73. doi:10.1109/tcc.2020.2975184.
12. Baden R, Bender A, Spring N, Bhattacharjee B, Starin D. Persona: an online social network with user-defined privacy. ACM SIGCOMM Comput Communicat Rev. 2009;39(4):135–46.
13. Dong W, Dave V, Qiu L, Zhang Y. Secure friend discovery in mobile social networks. In: 2011 Proceedings IEEE INFOCOM; 2011 Apr 10–15; Shanghai, China. Piscataway, NJ, USA: IEEE; 2011. p. 1647–55.
14. Jahid S, Mittal P, Borisov N. EASiER: encryption-based access control in social networks with efficient revocation. In: ASIACCS '11: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security; 2011 Mar 22–24; Hong Kong, China. New York, NY, USA: ACM; 2011. p. 411–5.
15. Li X, Yang G, Xiang T, Xu S, Zhao B, Pang H, et al. Make revocation cheaper: hardware-based revocable attribute-based encryption. In: 2024 IEEE Symposium on Security and Privacy (SP). Piscataway, NJ, USA: IEEE; 2024. p. 3109–27.
16. Wang H, Zheng Z, Wu L, Li P. New directly revocable attribute-based encryption scheme and its application in cloud storage environment. Clust Comput. 2017;20(3):2385–92. doi:10.1007/s10586-016-0701-7.
17. Zhang Y, Chen X, Li J, Li H, Li F. Attribute-based data sharing with flexible and direct revocation in cloud computing. KSII Trans Internet Inf Syst. 2014;8(11):4028–49.

18. Rasori M, Perazzo P, Dini G, Yu S. Indirect revocable KP-ABE with revocation undoing resistance. IEEE Trans Serv Comput. 2022;15(5):2854–68. doi:10.1109/tsc.2021.3071859.
19. Xu S, Yang G, Mu Y. Revocable attribute-based encryption with decryption key exposure resistance and ciphertext delegation. Inf Sci. 2019;479(8):116–34. doi:10.1016/j.ins.2018.11.031.
20. Huang X, Xiong H, Chen J, Yang M. Efficient revocable storage attribute-based encryption with arithmetic span programs in cloud-assisted internet of things. IEEE Trans Cloud Comput. 2023;11(2):1273–85. doi:10.1109/tcc.2021.3131686.
21. Sahai A, Seyalioglu H, Waters B. Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Advances in Cryptology—CRYPTO 2012. Vol. 7417. Cham, Switzerland: Springer; 2012. p. 199–217. doi:10.1007/978-3-642-32009-5_13.
22. Fan Y, Liu S, Tan G, Qiao F. Fine-grained access control based on Trusted Execution Environment. Future Gener Comput Syst. 2020;109(60):551–61. doi:10.1016/j.future.2018.05.062.
23. Costan V, Devadas S. Intel SGX explained. Cryptology ePrint Archive; 2016. Paper 2016/086.
24. Bethencourt J, Sahai A, Waters B. Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (SP '07); 2007 May 20–23; Berkeley, CA, USA. Piscataway, NJ, USA: IEEE; 2007. p. 321–34.
25. Yu J, Liu S, Xu M, Guo H, Zhong F, Cheng W. An efficient revocable and searchable MA-ABE scheme with blockchain assistance for C-IoT. IEEE Internet Things J. 2023;10(3):2754–66. doi:10.1109/jiot.2022.3213829.
26. Li J, Li D, Zhang X. A secure blockchain-assisted access control scheme for smart healthcare system in fog computing. IEEE Internet Things J. 2023;10(18):15980–9. doi:10.1109/jiot.2023.3268278.
27. Zhai Y, Yang H, Yao J, Wang T, Zhou Y, Zhu F, et al. CR^2-ABE: a blockchain-assisted coercion-resistant and revocable attribute-based encryption for IoMT. IEEE Internet Things J. 2025;12(9):13075–96. doi:10.1109/jiot.2024.3523959.
28. Liu C, Wang D, Li D, Guo S, Li W, Qiu X. Trusted access control mechanism for data with blockchain-assisted attribute encryption. High-Conf Comput. 2025;5(2):100265. doi:10.1016/j.hcc.2024.100265.
29. Yang X, Chang Z, Jiang M. Blockchain-assisted revocable hierarchical attribute-based encryption electronic medical record sharing scheme. J Big Data Comput. 2024;2(2):79. doi:10.62517/jbdc.202401211.
30. van Schaik S, Milburn A, Österlund S, Frigo P, Maisuradze G, Razavi K, et al. RIDL: rogue in-flight data load. In: 2019 IEEE Symposium on Security and Privacy. Piscataway, NJ, USA: IEEE; 2019. p. 88–105.
31. Kim T, Peinado M, Mainar-Ruiz G. STEALTHMEM: system-level protection against cache-based side channel attacks in the cloud. In: 21st USENIX Security Symposium (USENIX Security 12). Berkeley, CA, USA: USENIX Association; 2012. p. 189–204.