Public Verifiable Server-Aided Revocable Attribute-Based Encryption

Luqi Huang[1,✉] Fuchun Guo[1] Willy Susilo[1] and Yumei Li[1]

Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
lh852@uowmail.edu.au, fuchun@uow.edu.au, wsusilo@uow.edu.au, leamergo@gmail.com

Abstract. Revocable Attribute-Based Encryption (RABE) provides a user revocation mechanism to keep the system dynamic and protect data privacy when a user's credentials can be expired or revealed. In verifiable server-aided RABE, where most computations are delegated to an untrusted server, users with private keys and outsourced results can decrypt the ciphertext and then verify message correctness, provided they meet the decryption conditions. This verification process, commonly known as private verification, follows a "decrypt-then-verify" paradigm and is inherently restricted to legitimate users. However, such a model lacks transparency and accountability, as it depends solely on user-side validation. This limited verifiability may undermine trust in the server's behavior, exposing the system to disputes and potential misuse, such as malicious users falsely claiming incorrect computations, thereby leading to operational and financial risks. To address these challenges, we propose an improved server-aided RABE system that enables public verification, allowing any party to verify the untrusted server's computations rather than entirely relying on users. Furthermore, we introduce a "verify-then-decrypt" mechanism, ensuring the correctness of outsourced results before decryption. This approach prevents users from performing redundant or incorrect decryption, thereby improving efficiency and system reliability.

Keywords: Attribute-based encryption · Public verification · Outsourcing computation · User revocation.

1 Introduction

ABE plays a key role in enabling fine-grained and scalable access control systems, offering flexibility of managing data access in cloud computing. To maintain privacy and system integrity, RABE enforces periodic key updates to revoke users' private keys when they lose their keys or leave the cloud system. Furthermore, server-aided RABE extends the server's role beyond data storage to include key update processing and outsourced decryption. The common assumption that the fully trustworthy server is often unrealistic in practical scenarios, making verification of the server's actions essential to maintain system security and stability.

Traditional ABE systems with verifiable outsourcing in [9], [11], [18], [19] employ a "decrypt-then-verify" mechanism, where authorized users with private keys first decrypt ciphertexts using their private keys and outsourced results to retrieve the original message and then verify its correctness to ensure the accuracy of the server's computation. However, this private verification approach inherently restricts validation to authorized users with private keys, excluding third parties from confirming whether the server behaved correctly.

This single, vulnerable verification system creates opportunities for malicious user behavior. For example, a user with a valid private key could falsely claim that the server produced incorrect results to seek additional compensation, even when the outsourced computations were accurate. Since no other entities, including the server itself, can prove the correctness of outsourced results, such malicious defamation undermines the credibility of the server—particularly in cloud computing, where trust and reliability are critical.

To address the above concerns, [7], [12], [20] delegate the verification process of outsourced results to more trusted third-party auditors, such as private clouds or certification authorities. While this reduces dependence on end-users, the verification process still relies on private keys held by these entities. As a result, fully transparent and credibility verification cannot be achieved, and the risk of false claims against server computations persists. Therefore, enabling fully public verification of outsourced results on untrusted servers remains a significant challenge.

Generic Publicly Verifiable Computation (PVC) [13] provide a promising alternative, enabling lightweight clients to outsource computations to untrusted servers and verify the results independently. In 2015, James et al. [1] proposed a hybrid PVC scheme to verify outsourced computations in the ABE system. Their approach compares $H(m)$, the hash value of the outsourced result with the public verification key, where H is a hash function and m is the message, to confirm the correctness of the server's computations.

However, the hybrid PVC scheme presented in [1] has a critical limitation: the outsourced result $H(m)$ computed from the untrusted server, remains constant across verifications aiming at different authorized users. This means that once the server computes $H(m)$, it can repeatedly reuse this static hash value for all subsequent verifications. A robust verification mechanism should ensure that each verification attempt is unique and independently validates the server's current computations. Therefore, a more dynamic approach is needed—one that ensures outsourced results vary with each verification and across different authorized users.

In summary, while private verification in ABE systems lacks transparency and may result in redundant decryption on incorrect outputs, third-party auditing does not fully resolve credibility concerns. Furthermore, existing PVC integrations yield static, non-user-specific outsourced results. Achieving fully public, dynamic, and efficient verification of outsourced computations in server-aided RABE remains a challenging and open problem.

1.1 Our Contribution

In this paper, we present a Public Verifiable Server-Aided Revocable Attribute-Based Encryption (PV-SR-ABE) scheme that enables any party to verify the correctness of outsourced computations from an untrusted server. This public verification capability significantly enhances system transparency and mitigates the risks of fraud and server misconduct. Importantly, the verification process incurs only minimal computational overhead, making it suitable for practical deployment.

The PV-SR-ABE adopts a "verify-then-decrypt" paradigm, which ensures that the correctness of outsourced results is validated before decryption. This approach prevents users from redundantly decrypting incorrect or manipulated outputs, thereby improving efficiency and reliability. Additionally, our scheme guarantees that each verification generates user-specific results rather than static outsourced outputs, enhancing the system's robustness and security.

We formally prove the security of our scheme against chosen selective plaintext attacks (IND-sCPA) of user revocation under the one-user setting security model described in [15]. Moreover, we demonstrate selective public verifiability of the outsourced computations based on the Verifiable Delegable Computation (VDC) security model in [1], ensuring that our scheme meets robust security standards for verification processes.

1.2 Related Work

In 2004, Sahai and Waters proposed ABE in [17]. Since then, numerous variants have been developed to enhance flexibility in fine-grained access control. Typically, ABE schemes are categorized into ciphertext-policy ABE(CP-ABE) and key-policy ABE (KP-ABE). In CP-ABE, a user's attribute private key is linked to an attribute set, while a ciphertext specifies an access policy defined over the attribute universe of the system. Conversely, in KP-ABE, an access policy is embedded in a user's attribute private key, and a ciphertext is generated in relation to an attribute set. The first KP-ABE and CP-ABE schemes were introduced by Goyal et al. and Bethencourt et al. in [8] and [2], respectively.

However, a fundamental drawback of traditional ABE schemes is their high computational cost, which typically increases with the complexity of the access policy. To address this issue, Green et al. [9] introduced ABE with outsourced decryption, introducing a modular architecture that decomposes the decryption process into two stages: a computationally intensive preprocessing phase delegated to an untrusted proxy and a lightweight final decryption performed by the end user. After the proxy performs the heavy computation, the user obtains a simplified ciphertext that can be easily decrypted.

To preserve data privacy and ensure system integrity, revocable ABE schemes [10,6] have been developed, which enable the revocation of a user's private key when it is compromised or the user leaves the system. In 2016, Cui et al. [6] proposed a revocable CP-ABE scheme in which an untrusted server assists non-revoked users by transforming ciphertexts. Later, Qin et al. [15] identified a vulnerability related to decryption key exposure in [6], highlighting the risk of unauthorized access when revocation status is not properly enforced. Then Cheng et al. [5] proposed the first server-aided revocable ABE scheme proven secure in the multi-user setting. This advancement strengthened the practical applicability of revocable ABE in large-scale systems.

In the above server-aided revocable ABE schemes, whether the transformation process is correctly proceeded by the untrusted server that cannot be verified. To address this, Yang et al. [19] proposed a generic construction of verifiable SR-ABE that enables users with private keys to check whether the server behaves as expected. Additionally, Xu et al. [18] presented an adaptively secure scheme with verifiable outsourced decryption but not for user revocation.

2 Preliminaries

2.1 Bilinear Maps

Our design is based on some facts about groups with efficiently computable bilinear maps.

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

1. **Bilinearity:** For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. **Non-degeneracy:** $e(g, g) \neq 1$.
3. **Computability:** There is an efficient algorithm to compute $e(u, v)$ for $\forall u, v \in \mathbb{G}$.

2.2 Algorithm Definition

Based on the algorithm definition in the [15], we add the Verify algorithm for public verifiability of outsourced results. Our PV-SR-ABE scheme involves five entities: a private key generator (PKG), data owners, data users, public verifiers and an untrusted server. The PV-SR-ABE scheme that is associated with the identity space $\mathcal{I}$, the time period space $\mathcal{T}$ and the message space $\mathcal{M}$, consists of nine algorithms are defined as follows:

- **Setup**$(\lambda, \mathcal{U}, \mathcal{T}) \rightarrow (mpk, msk)$. The setup algorithm is executed by the PKG and takes as input the security parameter λ, the universal attribute set $\mathcal{U}$ and the time period space $\mathcal{T}$. It outputs the master public key, master secret key (mpk, msk).
- **UserKG**$(mpk, msk, ID, S) \rightarrow (sk_{ID}, vk_{ID}, psk_{ID,S}, st)$. The user key generation algorithm is executed by the PKG and takes as input the master pair keys (mpk, msk), identity ID and the attribute set S. It returns the private key sk_{ID}, public verification key vk_{ID}, long-term transformation key $psk_{ID,S}$, and the updated state st.

- **TKeyUp**$(mpk, msk, R, t) \rightarrow (tuk_t, st)$. The transformation key update algorithm is executed by the PKG and takes as input the master pair keys (mpk, msk), the revocation list R and time period t. It returns the update message key tuk_t and the updated state st.
- **TranKG**$(mpk, psk_{ID,S}, tuk_t) \rightarrow tk^S_{ID,t} / \perp$. The transformation key generation algorithm is executed by the server and takes as input the master public key mpk, the long-term transformation key $psk_{ID,S}$ and key update message tuk_t. It returns the short-term transformation key $tk^S_{ID,t}$ if $ID \notin R$.
- **Encrypt**$(mpk, (\mathbb{M}, \rho), t^{'}, M) \rightarrow (Hdr, c)$. The encryption algorithm is executed by a data owner and takes as input the master public key mpk, the access policy $(\mathbb{M}, \rho)$, the time period $t^{'}$ and a message M. It returns the header Hdr and the ciphertext c.
- **Transform**$(mpk, c, tk^S_{ID,t}, ID) \rightarrow \pi$. The transformation algorithm is executed by the server and takes as input master public key mpk, the ciphertext c, short-term transformation key $tk^S_{ID,t}$ and identity ID. It outputs the transformed ciphertext π if S satisfies the access policy $(\mathbb{M}, \rho)$, i.e., $S \in \mathbb{A}$ and $t^{'} = t$.
- **Verify**$(mpk, \pi, c, vk_{ID}, ID) \rightarrow 0/1$. The verify algorithm is executed by a verifier and takes as input the master public key mpk, the transformed ciphertext π, the ciphertext c, the public verification key vk_{ID}, and identity ID. It outputs the verification result 1 if the transformation was indeed done correctly.
- **Decrypt**$(mpk, sk_{ID}, \pi, Hdr, ID) \rightarrow M/ \perp$. The decryption algorithm is executed by a data user and takes as input master public key mpk, the private key sk_{ID}, the transformed ciphertext π, the header Hdr and identity ID. It outputs the message M if π is well-formed.
- **Revoke**$(ID, t, R, st) \rightarrow R$. The revoke algorithm is executed by the PKG and takes as inputs the identity ID, time period t, revocation list R and the state st. If the user's identity ID is revoked at time period t, this algorithm adds (ID, t) to R and outputs the updated revocation list R.

Correctness: We require the correctness of PV-SA-ABE are as follows:

- **TranKG Correctness:** For any $(mpk, msk) \leftarrow$ Setup$(\lambda, \mathcal{U}, \mathcal{T})$, and $(sk_{ID}$, vk_{ID}, $psk_{ID,S}, st) \leftarrow$ UserKG(mpk, msk, ID, S), $(tuk_t, st) \leftarrow$ TKeyUp$(mpk$, $msk, R, t)$, if $ID \notin R$, the TranKG algorithm always outputs the correct short-term transformation key $tk^S_{ID,t}$, otherwise, the $tk^S_{ID,t}$ can not be derived from $psk_{ID,S}$ and tuk_t.
- **Verify Correctness:** For any $(mpk, msk) \leftarrow$ Setup$(\lambda, \mathcal{U}, \mathcal{T})$, $(sk_{ID}, vk_{ID}, psk_{ID,S}$, $st) \leftarrow$ UserKG(mpk, msk, ID, S), $(Hdr, c) \leftarrow$ Encrypt$(mpk, (\mathbb{M}, \rho), t^{'}, M)$ and $\pi \leftarrow$ Transform$(mpk, c, tk^S_{ID,t}, ID)$, if the transformation by the server was indeed done correctly, the Verify algorithm always outputs the verification result 1, otherwise outputs 0.
- **Decrypt Correctness:** For any $(mpk, msk) \leftarrow$ Setup$(\lambda, \mathcal{U}, \mathcal{T})$, $(sk_{ID}$, vk_{ID}, $psk_{ID,S}, st) \leftarrow$ UserKG(mpk, msk, ID, S) and $(Hdr, c) \leftarrow$ Encrypt$(mpk, (\mathbb{M}, \rho)$, $t^{'}, M)$, $\pi \leftarrow$ Transform$(mpk, c, tk^S_{ID,t}, ID)$, if π is well-formed, the Decrypt algorithm always outputs the correctness message M, otherwise, the message cannot be decrypted using sk_{ID}.

2.3 Security Model

We adopt the security notion of *indistinguishability against chosen selective plaintext attacks* (IND-sCPA) for PV-SR-ABE from the security model in [15] for user revocation. The following game is played between an adversary $\mathcal{A}_1$ and the challenger $\mathcal{C}$.

- **Initialization.** $\mathcal{C}$ chooses an attribute universal set $\mathcal{U}$ and the time set $\mathcal{T}$. $\mathcal{A}_1$ declares access policy $\mathbb{A}^*$ and a time period t^*, which it will try to attack, and sends them to $\mathcal{C}$.
- **Setup.** $\mathcal{C}$ runs setup algorithm and gives the public parameter to $\mathcal{A}_1$. $\mathcal{C}$ keeps the master secret key msk, an initially empty revocation R and a state st.
- **Phase 1.** $\mathcal{A}_1$ adaptively issues the following queries to $\mathcal{C}$.
- • Create(ID, S): $\mathcal{C}$ runs user key generation algorithm on (ID, S) to obtain the pair $(sk_{ID}, vk_{ID}, psk_{ID,S})$ and stores in table T the entry $(ID, S, sk_{ID}, vk_{ID}, psk_{ID,S})$. It returns to $\mathcal{A}_1$ the long-term transformation key $psk_{ID,S}$ and the verification key vk_{ID}.
- • Corrupt(ID): If there exists an empty indexed by ID in table T, then $\mathcal{C}$ obtains the entry $(ID, S, sk_{ID}, vk_{ID}, psk_{ID,S})$ and sets $D = D \cup \{(ID, S)\}$. It returns to $\mathcal{A}_1$ the private key sk_{ID}. If no such entry exists, then it returns $\perp$.
- • TKeyUp(t): If $\mathcal{A}_1$ issues a key update query on a time period t, $\mathcal{C}$ runs the transformation key update algorithm TKeyUp(mpk, msk, t, R) and gives the update message key tuk_t to $\mathcal{A}_1$.
- • Revocation(ID, t): When $\mathcal{A}_1$ issues a revocation query on an identity ID and a time period t, $\mathcal{C}$ runs Revoke(ID, t, R, st) and outputs an updated revocation list R.
- **Challenge.** $\mathcal{A}_1$ submits two equal length messages M_0, M_1, an access policy $\mathbb{A}^*$ and a time period t^* satisfying the following constraints (Suppose that the attribute set associated with ID^* is S^*).
- • If there exists a tuple $(ID^*, S^*) \in D$ and $S^* \in \mathbb{A}^*$, then $\mathcal{A}_1$ must query the revocation oracle on (ID^*, t) at or before time period t^*.
- • If there exists a tuple $(ID^*, S^*, sk_{ID^*}, vk_{ID^*}, psk_{ID^*,S^*}) \in T$, $S^* \in \mathbb{A}^*$ and ID^* is not revoked at or before time period t^*, then $\mathcal{A}_1$ cannot query the private key on ID^*.

  $\mathcal{C}$ picks a random bit $b \in \{0, 1\}$ and sends the challenge header and challenge ciphertext (Hdr^*, c^*) to $\mathcal{A}_1$.
- **Phase 2.** $\mathcal{A}_1$ continues issuing queries to $\mathcal{C}$ as in Phase 1, with the same restrictions defined in the challenge phase.
- **Guess.** $\mathcal{A}_1$ makes a guess $b^{'}$ for b and wins the game if $b^{'} = b$. The advantage of $\mathcal{A}_1$ in this game is defined as $|\Pr[b^{'} = b] - 1/2|$.

Definition 1. The PV-SR-ABE scheme is IND-sCPA secure if all polynomial-time adversaries have at most a negligible advantage in the game defined above.

Here we discuss the security notion *selective public verifiability* based on the security definition when the mode setting as Verifiable Delegable Computation (VDC) in [1]. The following game is played between an adversary $\mathcal{A}_2$ and the challenger $\mathcal{C}$.

- **Initialization.** $\mathcal{C}$ chooses an attribute universal set $\mathcal{U}$ and the time set $\mathcal{T}$. $\mathcal{A}_2$ declares access policy $\mathbb{A}^*$ and a time period t^*, which it will try to attack, and sends them to $\mathcal{C}$.
- **Setup.** $\mathcal{C}$ runs the setup algorithm and obtains public parameters to $\mathcal{A}_2$. $\mathcal{C}$ keeps the master secret key to responds to queries from $\mathcal{A}_2$.
- **Challenge.** $\mathcal{C}$ sends the challenge header and ciphertext (Hdr^*, c^*) to $\mathcal{A}_2$.
- **Query.** $\mathcal{A}_2$ adaptively issues the following queries to $\mathcal{C}$.

- Create(ID, S): $\mathcal{C}$ runs user key generation algorithm on (ID, S) to obtain pair $(sk_{ID}, vk_{ID}, psk_{ID,S})$. It stores in table T the entry $(ID, S, sk_{ID}, vk_{ID}, psk_{ID,S})$ and then returns to $\mathcal{A}_2$.
- TKeyUp(t): If $\mathcal{A}_2$ issues a key update query on a time period t, $\mathcal{C}$ runs the transformation key update algorithm TKeyUp(mpk, msk, t, R) and gives the update message key tuk_t to $\mathcal{A}_2$.
- Transform(ID, S, t^*, c^*): $\mathcal{A}_2$ issues a transform query on the challenge cipheretxt c^*, identity ID with attribute set S, (ID, S) and challenge time period t^*. $\mathcal{C}$ runs the transform algorithm Transform$(mpk, c^*, tk^S_{ID,t^*}, ID)$ and gives the transformed ciphertext π to $\mathcal{A}_2$.
- Revocation(ID, t): When $\mathcal{A}_2$ issues a revocation query on an identity ID and a time period t, $\mathcal{C}$ runs Revoke(ID, t, R, st) and outputs an updated revocation list R.

- **Forgery.** $\mathcal{A}_2$ returns a forged transformed ciphertext π^* on c^* and (ID^*, S^*, t^*) (Suppose that the attribute set associated with ID^* is S^*) and wins game if

- π^* is valid and has not been queried in the transform query.
- psk_{ID^*,S^*} has not been queried in the user key query.

Definition 2. The PV-SR-ABE scheme is selective public verifiability if all polynomial-time adversaries have at most a negligible advantage in the selective public verifiability game defined above.

2.4 Binary Tree

We recall the definition of binary-tree data structure in [3]. In the $KUNodes$ algorithm, we use the following notations: BT denotes a binary-tree. $root$ denotes the root node of BT. x denotes a node in the binary tree and θ emphasizes that the node x is a leaf node. The set $Path(BT, \theta)$ stands for the collection of nodes on the path from the leaf θ to the root. If x is a non-leaf node, then x_l, x_r denote the left and right child of x, respectively. The $KUNodes$ algorithm in [3] takes as input a binary tree BT, a revocation list R and a time t, and outputs the minimal set Y of nodes, such that the corresponding key update information can only be used by the non-revoked users to generate a valid short-term transformation key. Specifically, the $KUNodes$ first marks all ancestors of users that were revoked by t as revoked nodes, then outputs all the non-revoked children of revoked nodes.

The description of the $KUNodes$ algorithm is as follows:

```
$KUNodes(BT, R, t)$ :
        $X, Y \leftarrow \emptyset$;
        $\forall(\theta_i, t_i) \in R, if\ t_i \leq t,$ then add $Path(\theta_i)$ to $X$;
        $\forall x \in X, if\ x_l \notin X,$ then add $x_l$ to $Y,$ if $x_r \notin X$ then add $x_r$ to $Y$;
        If $Y = \emptyset,$ then add root to $Y$; Return $Y$.
```

3 Constructions

3.1 Discussion and Overview

Discussion. To ensure message security, private verification using "decrypt-then-verify" restricts access to only authorized users. This is because the original plaintext—associated with the verification tag—can only be recovered by users capable of decrypting the ciphertext. Furthermore, existing data integrity verification protocols in cloud computing, such as Proof of Retrievability (PoR) and Proof of Data Possession (PDP), cannot be directly applied to server-aided RABE schemes, as the proofs generated by servers support only public verification and are not designed to facilitate the decryption process.

In contrast to the above, hybrid PVC selects a random message (instead of the original message) as the outsourced result, allowing anyone to verify its correctness. However, a constant outsourced result renders different verification meaningless. Thus, achieving public verification also requires outsourced results to be dynamically changeable for different users during decryption. To address this challenge, we propose a verify-then-decrypt mechanism that produces user-specific, publicly verifiable results that also enable authorized decryption.

Overview. Our approach separates message decryption from verification, treating them as distinct processes, which ensures that the correctness of outsourced results is validated before decryption. Therefore, the "verify-then-decrypt" process prevents users from redundantly decrypting incorrect outsourced results and introduces minimal computational overhead during public verification.

Specifically, our scheme embeds two secrets, α, β, which are used for message decryption and outsourced results verification, respectively. The encryption is structured on two parallel messages: the original message M, embedded in the header Hdr and a random message M_R, embedded in the ciphertext c. Correspondingly, each user with identity ID is issued two keys: a private key sk_{ID} for decryption and a public verification key vk_{ID} for verification. These two processes correspond to different secrets and are independent of each other.

When the user's attribute set S satisfies the access policy $(\mathbb{M}, \rho)$, such that $\rho(S) = 1$, and the update time period for updates t matches the time period t' in c, the server provides a transformed ciphertext, i.e., outsourced result π. This result π is publicly available and associated with a random message M_R, without revealing any information about the original message M. Additionally, it embeds a user-specific random value r ensuring dynamic changes. The transformed ciphertext π, supports both the following process: (1) Public Verification involving β: using the public verification key vk_{ID} provided by the PKG to obtain M_R and comparing the hashed value of M_R with the checknum in ciphertext c, any entity can verify the server's computations with π. Since the randome M_R can only be derived under the correct π and M_R is not a secret value. (2) Message Decryption relating α: after successful public verification, the authorized user possessing the private key sk_{ID} can decrypt the original message M using the validated outsourced result π and the header Hdr.

3.2 Scheme

Building upon the CP-ABE construction presented in [16] and the binary tree in [14] that is used for user revocations, we present our PV-SR-ABE scheme as follows:

– **Setup**(λ,$\mathcal{U}$, $\mathcal{T}$) The setup algorithm takes as input the security parameter λ, the universal attribute set $\mathcal{U}$ and the time set $\mathcal{T}$. It then generates a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ where $\mathbb{G}$, $\mathbb{G}_T$ are two cyclic groups of prime order $p = p(\lambda)$. Choose a random generator g of group $\mathbb{G}$. Set $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$.It selects randoms $\alpha, \beta \in \mathbb{Z}_p$, a hash function $H : \mathbb{G}_T \to \mathbb{G}$ and random elements $g_0, k, k_0, u, h, u_0, h_0, u_1, h_1, w, v \in \mathbb{G}$ and sets $e(g_0, k_0) = e(u_0, g)$. This algorithm initializes the empty revocation list R, the secret state $st = BT$ and keeps the master secret key msk=(α, β, st). It also outputs the master public key as

$$mpk = (\mathcal{G}, g_0, k, k_0, u, h, u_0, h_0, u_1, h_1, w, v, e(g, g)^{\alpha}, e(u_0, g)^{\beta}, H)$$

– **UserKG**(mpk, msk, ID, S) The user key generation algorithm takes as input the master public key and master secret key (mpk, msk). It also inputs the user's identity ID and attribute set S. The algorithm randomly chooses $r \in \mathbb{Z}_p$ and generates the user's private key sk_{ID} and the public verification key vk_{ID} as follows:

$$sk_{ID} = (g^{\alpha}(u_0^{ID} h_0)^r, g^r), vk_{ID} = g_0^{\beta+rID}$$

For the user's identity ID, it chooses an undefined leaf node θ from the binary tree BT. Next for each node $x \in Path(\theta)$, the algorithm fetches g_x from the node x. If x has not been defined, it randomly chooses $g_x \in \mathbb{G}$, and stores g_x in the node x. It also picks random exponents $r_x, r_{x,1}, \ldots, r_{x,|S|} \in \mathbb{Z}_p$. The algorithm generates long-term transformation key $psk_{ID,S}$ as follows and updates the state st.

$$psk_{ID,S} = (\{((w^{ID}k)^{r_x}/g_x)u_0^r, g^{r_x}, \{(u^{S_\tau}h)^{r_{x,\tau}} v^{-r_x}, g^{r_{x,\tau}}\}_{\tau \in S}\}_{x \in Path(\theta)})$$

– **TKeyUp**(mpk, msk, R, t) The transformation key update algorithm inputs the master public key and master secret key (mpk, msk). It also inputs the revocation list R, the update time period t. For each $x \in KUNodes(BT, R, t)$, where $KUNodes$ is the algorithm in Section 2.4 that outputs the corresponding key update information only from non-revoked users, the algorithm fetches g_x from x (g_x should always be predefined in the above UserKG algorithm), randomly chooses $s_x \in \mathbb{Z}_p$. The algorithm updates the state st and generates the update message key tuk_t as follows:

$$tuk_t = (\{g_x(u_1^t h_1)^{s_x}, g^{s_x}\}_{x \in KUNodes(BT,R,t)})$$

- **TranKG**$(mpk, psk_{ID,S}, tuk_t)$ The transformation key generation algorithm inputs the master public key mpk, long-term transformation key $psk_{ID,S}$ and update message key tuk_t. Suppose that θ is the leaf node storing the identity ID. Let $I = \{x : x \in Path(\theta)\}$ and $J = \{x : x \in KUNodes(BT, R, t)\}$. If $I \cap J = \emptyset$, it returns $\perp$. Otherwise, there must be exactly one node $x \in I \cap J$. The short-term transformation key $tk^S_{ID,t}$ can be computed as follows:

$$tk^S_{ID,t} = ((w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}, g^{r_x}, g^{s_x}, \{(u^{S_\tau} h)^{r_{x,\tau}} v^{-r_x}, g^{r_{x,\tau}}\}_{\tau \in S})$$

- **Encrypt**$(mpk, (\mathbb{M}, \rho), t', M)$ The encryption algorithm takes as input a message M, the master public key mpk, the access policy $(\mathbb{M}, \rho)$ with $\mathbb{M} \in \mathbb{Z}_p^{l \times n}$, $\rho : [l] \to \mathbb{Z}_p$ and the time period t'. It sets $\mathbb{A} = (\mathbb{M}, \rho)$ and selects randoms $s, s_\tau \in \mathbb{Z}_p$, a random message $M_R \in \mathcal{M}$ with same length of M. This algorithm also selects $\overrightarrow{y} = (s, y_2, \ldots, y_n) \in \mathbb{Z}_p^{n+1}$ and the shares is $\overrightarrow{\lambda} = \mathbb{M} \cdot \overrightarrow{y}$. It generates the header as $Hdr = (e(g,g)^{\alpha s} M, g^s, h_0^s)$, and ciphertext c as follows:

$$c = (e(u_0, g)^{\beta s} M_R, H(M_R)^s, g^s, k^s, k_0^s, (u_1^{t'} h_1)^s, \{w^{\lambda_\tau} v^{s_\tau}, (u^{\rho(\tau)} h)^{-s_\tau}, g^{s_\tau}\}_{\tau \in l})$$

- **Transform**$(mpk, c, tk^S_{ID,t}, ID)$ The transformation algorithm takes as input the ciphertext c, master public key mpk, short-term transformation key $tk^S_{ID,t}$ and the identity ID. This algorithm computes the constants $\{\omega_i \in \mathbb{Z}_p\}$ such that $\sum_{\rho(i) \in S} \omega_i \mathbb{M}_i = (1, 0, \ldots, 0)$. If S satisfies the access policy $(\mathbb{M}, \rho)$, i.e. $\mathbb{A}(S) = 1$, the server computes

$$\prod_{\rho(i) \in S} (e((u^{A_i} h)^{r_{x,i}} v^{-r_x}, g^{s_i}) \cdot e((u^{\rho(i)} h)^{-s_i}, g^{r_{x,i}}) \cdot e(w^{\lambda_i} v^{s_i}, g^{r_x}))^{\omega_i} = e(w^s, g^{r_x})$$

$$\text{If } t = t', \pi = \frac{e((w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}, g^s)}{e(w^s, g^{r_x})^{ID} e(k^s, g^{r_x}) \cdot e((u_1^{t'} h_1)^s, g^{s_x})} = e(u_0^r, g^s)$$

- **Verify**$(mpk, c, \pi, vk_{ID}, ID)$ The verify algorithm takes as input the master public key mpk, the transformed ciphertext π, the public verification key vk_{ID} and the identity ID.

  If $e(g^s, H(\frac{e(u_0,g)^{\beta s} M_R \cdot e(u_0^r, g^s)^{ID}}{e(g_0^{\beta + rID}, k_0^s)})) = e(H(M_R)^s, g)$, return 1, else return 0.

- **Decrypt**$(mpk, sk_{ID}, \pi, Hdr, ID)$ The decryption algorithm takes as input the master public key mpk, the private key sk_{ID}, and the transformed ciphertext π, the header Hdr and the identity ID.

$$\frac{e(g^\alpha (u_0^{ID} h_0)^r, g^s)}{e(u_0^s, g^r)^{ID} \cdot e(h_0^s, g^r)} = e(g,g)^{\alpha s}$$

  It outputs the message $M = e(g,g)^{\alpha s} \cdot M / e(g,g)^{\alpha s}$.

- **Revoke**(ID, t, R, st) The revoke algorithm takes as inputs the identity ID, time period t, revocation list R and the state st. If the user's identity ID is revoked at time period t, this algorithm adds (ID, t) to R and outputs the updated revocation list R.

TranKG Correctness: If $ID \notin R$, then $I \cap J = \emptyset$ where $I = \{x : x \in Path(\theta)\}$ and $J = \{x : x \in KUNodes(BT, R, t)\}$. Therefore, there must be exactly one node $x \in I \cap J$ from the definition of the $KUNodes$ algorithm. The first item in the short-term transformation key $tk^S_{ID,t}$ can be derived from $pk_{ID,S}$ and tuk_t as

$$((w^{ID}k)^{r_x}/g_x)u_0^r \cdot g_x(u_1^t h_1)^{s_x} = (w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}$$

Verify Correctness: During Transform algorithm, if S satisfies the access policy $\mathbb{A}$ in the ciphertext, i.e. $\mathbb{A}(S) = 1$, we have that $\sum_{\rho(i)\in S} \omega_i \lambda_i = s$, and compute

$$\frac{e((w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}, g^s)}{\prod_{\rho(i)\in S}(e((u^{A_i}h)^{r_{1,i}} v^{-r_x}, g^{s_i}) \cdot e((u^{\rho(i)}h)^{-s_i}, g^{r_{1,i}}) \cdot e(w^{\lambda_i} v^{s_i}, g^{r_x}))^{ID\omega_i} \cdot e(k^s, g^{r_x})}$$
$$= \frac{e((w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}, g^s)}{\prod_{\rho(i)\in S}(e(v^{-r_x}, g^{s_i}) \cdot e(w^{\lambda_i} v^{s_i}, g^{r_x}))^{ID\omega_i} \cdot e(k^s, g^{r_x})}$$
$$= \frac{e((w^{ID}k)^{r_x} u_0^r (u_1^t h_1)^{s_x}, g^s)}{e(w^s, g^{r_x})^{ID} \cdot e(k^s, g^{r_x})} = e(u_0^r (u_1^t h_1)^{s_x}, g^s)$$

If $t = t^{'}$, we can compute $\pi = \frac{e(u_0^r(u_1^t h_1)^{s_x}, g^s)}{e((u_1^{t^{'}} h_1)^s, g^{s_x})} = e(u_0^r, g^s)$.
Then the verify algorithm computes $M_R = \frac{e(u_0,g)^{\beta s} M_R \cdot e(u_0^r, g^s)^{ID}}{e(g_0^{\beta + rID}, k_0^s)}$ based on $e(g_0, k_0) = e(u_0, g)$ and then test $e(g^s, H(M_R)) \overset{?}{=} e(H(M_R)^s, g)$.
Decrypt Correctness: If S satisfies the access policy $\mathbb{A}$ in the ciphertext and $t = t^{'}$, and the tranformation by the unstruted server is public verified as correct, we can obtain the transformed ciphertext π. Then we compute

$$\frac{e(g^\alpha (u_0^{ID} h_0)^r, g^s)}{e(u_0^s, g^r)^{ID} \cdot e(h_0^s, g^r)} = e(g, g)^{\alpha s}$$

It outputs the message $M = e(g, g)^{\alpha s} \cdot M / e(g, g)^{\alpha s}$.

4 Security Analysis

4.1 Complexity Assumption

Decisional $(q - 1)$ **Assumption [16].** For our construction, we will use a $q - 1$ type assumption in [16]. This assumption can be proved secure in the following game between $\mathcal{C}$ and $\mathcal{A}$:

Initially $\mathcal{C}$ calls the group generation algorithm with input the security parameter, picks a random group element $g \in \mathbb{G}$, and $q + 2$ random exponents

$a, s, \overrightarrow{b} = \{b_1, \ldots, b_q\} \in \mathbb{Z}_p$. Then it sends to $\mathcal{A}$ the group description $(p, \mathbb{G}, \mathbb{G}_p, e)$ and all of the following terms:

$$\begin{aligned}
&g, g^s \\
&g^{a^i}, g^{b_j}, g^{sb_j}, g^{a^i b_j}, g^{\frac{a^i}{(b_j)^2}}, \forall (i,j) \in [q,q] \\
&g^{\frac{a^i b_j}{(b_{j'})^2}}, \forall (i,j,j') \in [2q,q,q], j \neq j^{'} \\
&g^{\frac{a^i}{b_j}}, \forall (i,j) \in [2q,q], i \neq q+1 \\
&g^{\frac{sa^i b_j}{b_{j'}}}, g^{\frac{sa^i b_j}{(b_{j'})^2}} \forall (i,j,j^{'}) \in [q,q,q], j \neq j^{'}.
\end{aligned}$$

$\mathcal{C}$ flips a random coin $b \in \{0,1\}$ and if $b = 0$, it gives to $\mathcal{A}$ the term $T = e(g,g)^{sa^{q+1}}$. Otherwise, it gives a random term $T = Z \in \mathbb{G}_T$. Finally the $\mathcal{A}$ outputs a guess $b^{'} \in \{0,1\}$.

q-Bilinear Diffle-Helleman Exponent Assumption [4]. We define the q-Bilinear Diffle-Helleman Exponent Assumption(q-BDHE) as follows. Choose a group $\mathbb{G}$ of prime order p according to the security parameter. Let $a \in \mathbb{Z}_p$, $f \in \mathbb{G}$ be chosen and g be a generator of $\mathbb{G}$. If an adversary is given

$$g, g^a, g^{a^2}, \ldots, g^{a^q}, g^{a^{q+2}}, g^{a^{q+3}}, \ldots, g^{a^{2q}}, f$$

it must compute $e(g,f)^{a^{q+1}}$.

4.2 Security Proof

In this section, we provide proof of security for the notions of IND-sCPA in a one-user setting based on [15] and selective verifiability in [1].

Theorem 1. (Selectively IND-CPA Security). *If the $q-1$ assumption in [16] holds then no polynomial adversary can selectively break the PV-SR-ABE scheme with a challenge access policy $(\mathbb{M}^*, \rho^*)$ and a challenge time period t^*.*

We first present the proof of the traditional security notion of IND-sCPA. In the one-user setting, only one user (called "target user") has the capacity to access the challenge ciphertext and the adversary can corrupt either his long-term transformation keys or his private keys. So, the adversary falls into the following two distinct classes.
In Type 1, the target user is revoked at or before the challenge time period t^* and thus the adversary is allowed to corrupt the target user's private key.
In Type 2, the target user is not revoked, so the adversary is not allowed to corrupt the target user's private key.
Proof of Type 1 Adversary. Suppose there exists an adversary $\mathcal{A}_1$ who breaks the proposed scheme with ϵ. We built a simulator $\mathcal{B}$ that has advantage $\frac{\epsilon}{2}$ in

solving the $q-1$ assumption in [16]. Given as input an assumption instance, $\mathcal{B}$ plays the role of the challenger $\mathcal{C}$ in the game and interacts with the adversary $\mathcal{A}_1$ as follows.

– **Initialization.** $\mathcal{B}$ first chooses the attribute universal set $\mathcal{U}$, the time set $\mathcal{T}$ and the maximum number of users n_{max}. $\mathcal{A}_1$ outputs a challenge access policy $(\mathbb{M}^*, \rho^*)$ and a challenge time period t^* that it intends to attack. We have that $\mathbb{M}^*$ is an $l \times n$ matrix, where $l \ll q$.

– **Setup.** $\mathcal{B}$ picks randoms $\omega_R, \tilde{\alpha}, \tilde{\beta} \in \mathbb{Z}_p$ and sets $\alpha = a^{q+1} + \tilde{\alpha}, \beta = a + \tilde{\beta}$. Then $\mathcal{B}$ picks the random numbers $\widetilde{k}, \widetilde{h}_0, \widetilde{u}_1, \widetilde{h}_1, \widetilde{u}, \widetilde{v}, \widetilde{h} \in \mathbb{Z}_p$ and sets $e(u_0, g) = e(g_0, k_0)$, H be the randome oracle controled by $\mathcal{B}$. Using the assumption, $\mathcal{B}$ gives to $\mathcal{A}_1$ the following public parameters:

$$g = g, g_0 = g^{\frac{a^q}{b_q}}, k_0 = g^{b_q}, w = g^a, v = g^{\widetilde{v}} \cdot \prod_{(j,k)\in[l,n]} (g^{\frac{a^{k+2}}{b_j}})^{M^*_{j,k}},$$

$$k = g^{\frac{b_q a^q}{b_1^2} + \widetilde{k}}, u = g^{\widetilde{u}} \cdot \prod_{(j,k)\in[l,n]} (g^{\frac{a^{k+1}}{b_j^2}})^{M^*_{j,k}}, u_0 = g^{a^q}, u_1 = g^{a+\widetilde{u}_1},$$

$$h = g^{\widetilde{h}} \cdot \prod_{(j,k)\in[l,n]} (g^{\frac{a^{k+1}}{b_j^2}})^{-\rho(j)^* M^*_{j,k}}, h_0 = g^{\frac{b_1 a}{b_q^2} + \widetilde{h}_0}, h_1 = g^{-t^* a + \widetilde{h}_1},$$

$$e(g,g)^\alpha = e(g^a, g^{a^q}) \cdot e(g,g)^{\tilde{\alpha}}, e(u_0, g)^\beta = e(g^a, g^{a^q}) \cdot e(g^{a^q}, g)^{\tilde{\beta}}.$$

Finally, $\mathcal{B}$ gives $\mathcal{A}_1$ mpk and initializes the empty revocation list R and the secret state $st = BT$.

– **Phase 1 and Phase 2 Queries:** $\mathcal{B}$ randomly chooses an unassigned leaf node θ^* for storing the target user ID^* and answers the $\mathcal{A}_1$'s queries as follows.

• **Create**(ID, S)**:** When $\mathcal{A}_1$ issues a query on (ID, S), $\mathcal{B}$ creates the table T's element $(ID, S, sk_{ID}, vk_{ID}, pk_{ID,S})$ and returns $pk_{ID,S}, vk_{ID}$ to $\mathcal{A}_1$ through the following way.

∗ If $S \in \mathbb{A}^*$, $\mathcal{B}$ sets $ID = ID^*$ and stores ID^* in the pre-assigned leaf node θ^*. Then $\mathcal{B}$ randomly picks $\widetilde{r} \in \mathbb{Z}_p$ and implicitly sets $r = \widetilde{r} + \omega_1 a$, $\omega_1 = \frac{-1}{ID}$. $\mathcal{B}$ generates private key sk_{ID} as follows:

$$\begin{aligned} g^\alpha (u_0^{ID} h_0)^r =& g^{a^{q+1}+\widetilde{\alpha}} \cdot ((g^{a^q})^{ID} \cdot g^{\frac{b_1 a}{b_q^2} + \widetilde{h}_0})^{\widetilde{r}+\omega_1 a} \\ =& g^{\widetilde{\alpha}} \cdot (u_0^{ID} h_0)^{\widetilde{r}} \cdot h_0^{\omega_1 a}, \end{aligned}$$

and $g^r = g^{\widetilde{r}+\omega_1 a}$, $vk_{ID} = g_0^{\beta + rID} = (g^{\frac{a^q}{b_q}})^{\widetilde{\beta}+\widetilde{r}ID}$. For each node $x \in Path(\theta^*)$, $\mathcal{B}$ randomly chooses an exponent $\widetilde{g}_x \in \mathbb{Z}_p$ and implicitly sets $g_x = g^{-\frac{a^{q+1}}{ID} + \widetilde{g}_x}$. It also randomly selects $\widetilde{r}_x, \omega_1^{'} \in \mathbb{Z}_p$ and generates $r_x = \widetilde{r}_x + \omega_1^{'} a^{q-1}$. It computes as follows

$$\begin{aligned} ((w^{ID} k)^{r_x} / g_x) u_0^r =& (g^{aID} g^{\frac{b_q a^q}{b_1^2} + \widetilde{k}_0})^{\widetilde{r}_x + \omega_1^{'} a^{q-1}} \cdot g^{\frac{a^{q+1}}{ID} - \widetilde{g}_x} \cdot (g^{a^q})^{\widetilde{r}+\omega_1 a} \\ =& (w^{ID} k)^{r_x} \cdot g^{-\widetilde{g}_x} \cdot (g^{a^q})^{\widetilde{r}} \end{aligned}$$

For each attribute $S_\tau \in S$, $\mathcal{B}$ randomly selects $\widetilde{r}_{x,\tau}, \omega_i^{'} \in \mathbb{Z}_p$ and sets $r_{x,\tau} = \widetilde{r}_{x,\tau} + \sum_{(i,i')\in[n,l],\rho^*(i')\in S} \omega_i^{'} b_{i'} a^{q+1-i}$. Thus $\mathcal{B}$ generates the $(u^{S_\tau}h)^{r_{x,\tau}} v^{-r_x} = (g^{\widetilde{u}S_\tau+\widetilde{h}})^{r_{x,\tau}} \cdot \prod_{(j,k)\in[l,n]}(g^{\frac{(S_\tau-\rho^*(j))M^*_{j,k}a^{k+2}}{b_j^2}})^{r_{x,\tau}} \cdot v^{-r_x}$. Note that $S \in \mathbb{A}^*$, the specific authorized set defined as $\mathcal{S}_1$, for all $S_\tau \in \mathcal{S}_1$, the $S_\tau - \rho^*(i^{'})$ are zero and thus the attributes sets part $(u^{S_\tau}h)^{r_{x,\tau}} v^{-r_x}$ of long-term transformation key can be simulated by $\mathcal{B}$.

$*$ If $S \notin \mathbb{A}^*$, $ID \neq ID^*$, $\mathcal{B}$ randomly chooses an unassigned leaf node θ and stores ID in it. sk_{ID}, vk_{ID} are similar with that when $S \in \mathbb{A}^*$.

1. If $x \in Path(\theta) \cap Path(\theta^*)$, $\mathcal{B}$ selects randoms $\widetilde{r}_x^{'}, \theta_0, \ldots, \theta_n \in \mathbb{Z}_p$ and sets $r_x = \widetilde{r}_x^{'} + \theta_0 a^{q-1} + \sum_{i\in[n]} \theta_i a^{q-1-i}$. Then $\mathcal{B}$ builds the long term transformation key $psk_{ID,S}$ similarly with it when $S \in \mathbb{A}^*$. For each attribute $S_\tau \in S$, $\mathcal{B}$ randomly selects $\widetilde{r}_{x,\tau}^{'} \in \mathbb{Z}_p$ and sets $r_{x,\tau} = \widetilde{r}_{x,\tau}^{'} + \frac{\sum_{(i,i')\in[n,l],\rho^*(i')\notin S} \theta_i b_{i'} a^{q-1-i}}{S_\tau-\rho^*(i')}$. Note that $S \notin \mathbb{A}^*$, the specific unauthorized set defined as $\mathcal{S}_2$, for all $S_\tau \in \mathcal{S}_2$, the $S_\tau - \rho^*(i^{'})$ are not zero. Thus $\mathcal{B}$ can only simulate $(u^{S_\tau}h)^{r_{x,\tau}} v^{-r_x}$ for the unauthroized attribute set. Specifically, the first part v^{-r_x} for these terms is the following:

$$v^{-\widetilde{r}_x^{'}-\theta_0 a^{q-1}} \prod_{(i,j,k)\in[n,l,n]} g^{\frac{-\theta_i M^*_{j,k} a^{q-i+k+1}}{b_j}} = \varPhi \prod_{\substack{(i,j)\in[n,l]\\ \rho^*(j)\notin S}} g^{\frac{-<\boldsymbol{\theta},\boldsymbol{M_j^*}>a^{q+1}}{b_j}}$$

Then $\mathcal{B}$ generates the $(u^{S_\tau}h)^{r_{x,\tau}}$ as following:

$$(g^{\widetilde{u}S_\tau+\widetilde{h}})^{r_{x,\tau}} \prod_{(j,k)\in[l,n]} (g^{\frac{(S_\tau-\rho^*(j))b_{i'} M^*_{j,k}\theta_i a^{k+2}}{(S_\tau-\rho^*(j))b_j^2}})^{r_{x,\tau}} = \varLambda \prod_{\substack{(i,j)\in[n,l]\\ \rho^*(j)\notin S}} g^{\frac{<\boldsymbol{\theta},\boldsymbol{M_j^*}>a^{q+1}}{b_j}}$$

2. If $x \notin Path(\theta) \cap Path(\theta^*)$, $\mathcal{B}$ knows the value g_x and can compute the following. $\mathcal{B}$ selects randoms $\widetilde{r}_x^{''}, \mu_1, \ldots, \mu_n \in \mathbb{Z}_p$ and sets $r_x = \widetilde{r}_x^{''} + \mu_0 a^q + \sum_{i\in[n]} \mu_i a^{q-1-i}$, $\mu_0 = \frac{1}{ID^2}$. Then $\mathcal{B}$ builds the long term transformation key $psk_{ID,S}$ as follows:

$$((w^{ID}k)^{r_x}/g_x)u_0^r = (g^{aID})^{\widetilde{r}_x^{''}+\sum_{i\in[n]}\mu_i a^{q-1-i}} k^{r_x} \cdot g_x^{-1} \cdot (g^{a^q})^{\widetilde{r}}$$

$g^{r_x} = g^{\widetilde{r}_x+\mu_0 a^q+\sum_{i\in[n]}\mu_i a^{q-1-i}}$. For each attribute $S_\tau \in S$, $\mathcal{B}$ randomly selects $\widetilde{r}_{x,\tau}^{''} \in \mathbb{Z}_p$ and sets $r_{x,\tau} = \widetilde{r}_{x,\tau}^{''} + \frac{\sum_{(i,i')\in[n,l],\rho^*(i')\notin S} \mu_i b_{i'} a^{q-1-i}}{S_\tau-\rho^*(i')}$. Note that $S \notin \mathbb{A}^*$, for all $S_\tau \in S$, the $S_\tau - \rho^*(i^{'})$ are not zero. Thus the common part v^{-r_x} for these terms is the following:

$$v^{-\widetilde{r}_x^{''}} (g^{\widetilde{v}})^{-r_x} \prod_{(j,k)\in[l,n]} ((g^{\frac{a^{k+2}}{b_j}})^{M^*_{j,k}})^{-\mu_0 a^q} \prod_{(i,j,k)\in[n,l,n]} g^{\frac{-\mu_i M^*_{j,k} a^{q+1-i+k}}{b_j}}$$

Thus $\mathcal{B}$ generates the $(u^{S_\tau} h)^{r_{x,\tau}}$ as following:

$$(g^{\widetilde{u}S_\tau+\widetilde{h}})^{r_{x,\tau}} \prod_{(j,k)\in[l,n]} g^{\frac{(S_\tau-\rho^*(j))b_{i'}\mu_i M^*_{j,k}a^{k+2}}{(S_\tau-\rho^*(j))b_j^2}} = \Theta \prod_{\substack{(i,j)\in[n,l]\\ \rho^*(j)\notin S}} g^{\frac{-<\boldsymbol{\mu},\boldsymbol{M_j^*}>a^{q+1}}{b_j}}$$

- **Corrupt(ID):** If there exists an entry indexed by ID in table T, then $\mathcal{B}$ obtains the entry $(ID, S, sk_{ID}, vk_{ID}, pk_{ID,S})$ and sets $D = D \cup \{(ID, S)\}$. It returns to $\mathcal{A}_1$ the private key sk_{ID}. If no such entry exists, then it returns $\perp$. Note that, for Type 1 attack, $\mathcal{A}_1$ is allowed to corrupt the target user ID^* and obtain the private key sk_{ID^*}.
- **TKeyUp(t):** If $\mathcal{A}_1$ issues a key update query on a time period t, $\mathcal{B}$ computes the key update message tuk_t as follows. For all $x \in KUNodes(BT, R, t)$, $\mathcal{B}$ fectches g_x from node x. If x is not defined, it randomly chooses g_x and stores it in the node x.

* If $x \notin Path(\theta^*)$, $\mathcal{B}$ knows g_x. Thus it chooses a random exponent $s_x \in \mathbb{Z}_p$ and computes $tuk_t = (g_x(u_1^t h_1)^{s_x}, g^{s_x})$.

* If $x \in Path(\theta^*)$, $\mathcal{B}$ does not know g_x. $\mathcal{B}$ selects a random number $\widetilde{s}_x \in \mathbb{Z}_p$ and sets $s_x = \widetilde{s}_x + \frac{a^q}{ID(t-t^*)}$. Then $\mathcal{B}$ computes

$$\begin{aligned} g_x(u_1^t h_1)^{s_x} =& g^{-\frac{a^{q+1}}{ID}+\widetilde{g}_x}((g^{a+\widetilde{u}_1})^t \cdot g^{-t^*a+\widetilde{h}_1})^{\widetilde{s}_x+\frac{a^q}{ID(t-t^*)}} \\ =& g^{\widetilde{g}_x}(g^{t\widetilde{u}_1} \cdot g^{\widetilde{h}_1})^{\frac{a^q}{ID(t-t^*)}}(u_1^t h_1)^{\widetilde{s}_x} \end{aligned}$$

- **Revoke(ID,t):** When $\mathcal{A}_1$ issues a revocation query on an identity ID at time period t, $\mathcal{B}$ adds (ID, t) to the revocation list R.

– **Challenge.** When $\mathcal{A}_1$ submits two equal length messages M_0, M_1, $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$. $\mathcal{B}$ picks a random message M_R with same length of M_0 and M_1. $\mathcal{B}$ randomly selects $\omega_R \in \mathbb{Z}_p$ and sets g^{ω_R} as the response of queries M_R to the random oracle H. Then $\mathcal{B}$ constructs

$$\begin{aligned} &Hdr^* = (T \cdot e(g, g^s)^{\widetilde{\alpha}} \cdot M_b, g^s, h_0^s = g^{\frac{sab_1}{b_q^2}} \cdot g^{s\widetilde{h}_0}), \\ &c^* = (T \cdot e(g^{a^q}, g^s)^{\widetilde{\beta}} \cdot M_R, H(M_R)^s = g^{s\omega_R}, g^s, k^s = g^{\frac{sa^q b_q}{b_1^2}} \cdot g^{s\widetilde{k}}, \\ &\qquad k_0^s = g^{sb_q}, (u_1^t h)^s = (g^s)^{\widetilde{u}_1 t^*+\widetilde{h}_1}, \{w^{\lambda_\tau} v^{s_\tau}, (u^{\rho^*(\tau)}h)^{-s_\tau}, g^{s_\tau}\}_{\tau\in l}) \end{aligned}$$

$\mathcal{B}$ sets implicitly $\overrightarrow{y} = (s, sa+\widetilde{y}_2, sa^2+\widetilde{y}_3, \ldots, sa^{n-1}+\widetilde{y}_n)$, where $\widetilde{y}_2, \ldots, \widetilde{y}_n \in \mathbb{Z}_p$. For each row $\mathcal{B}$ sets implicitly $s_\tau = -sb_\tau$ and since $\overrightarrow{\lambda} = \mathbb{M}^*\overrightarrow{y}$, we have that $\lambda_\tau = \sum_{i\in[n]} \mathbb{M}^*_{\tau,i} sa^{i-1} + \widetilde{\lambda}_\tau$. Therefore, $\mathcal{B}$ returns the challenge header and challenge ciphertext (Hdr^*, c^*) to $\mathcal{A}_1$.

– **Guess.** Finally. $\mathcal{A}_1$ outputs guess b' for the challenge bit. If $b' = b$, $\mathcal{B}$ outputs 1, it claims that the challenge term is $T = e(g, g)^{sa^{q+1}}$. Otherwise, it outputs 0 meaning $T = Z$ is random in $\mathbb{G}_T$.

When $T = Z$, the ciphertext will give no information about the simulator's choice of b. In this case, we have $\Pr[b' = b|T = Z] = \Pr[b' \neq b|T = Z] = \frac{1}{2}$, beacuse the challenge header will contain only random numbers. Since the silumator outputs a guess $T^{'} = Z$ when $b^{'} = b$, we then have $\Pr[T^{'} = T|T = Z] = \frac{1}{2}$. When $T = e(g,g)^{sa^{q+1}}$, $\mathcal{A}_1$ can sees a valid encryotion of M_b. In this situation, the $\mathcal{A}_1$'s advantage is ϵ by definition. The probability is $\Pr[b^{'} = b|T = e(g,g)^{sa^{q+1}}] = \frac{1}{2}+\epsilon$. Since the simulator outputs a guess $T = e(g,g)^{sa^{q+1}}$ of T when $b^{'} = b$, we have $\Pr[T^{'} = T|T = e(g,g)^{sa^{q+1}}] = \frac{1}{2} + \epsilon$. Hence, by putting them all together, $\mathcal{B}$'s advantage in the above game is

$$
\begin{aligned}
&\Pr[T' = T] - \frac{1}{2} \\
=&\Pr[T' = T|T = e(g,g)^{sa^{q+1}}]\Pr[T = e(g,g)^{sa^{q+1}}] + \Pr[T' = T|T = Z]\Pr[T = Z] - \frac{1}{2} \\
=&\frac{1}{2}\Pr[T' = T|T = e(g,g)^{sa^{q+1}}] + \frac{1}{2}\Pr[T' = T|T = Z] - \frac{1}{2} \\
=&\frac{1}{2}(\frac{1}{2} + \epsilon + \frac{1}{2}) - \frac{1}{2} = \frac{\epsilon}{2}
\end{aligned}
$$

Similarly, we can prove Theorem 1 for *Type 2 adversary.* In the proof of Type 2 adversary, the target user ID^* is not revoked and the adversary is not allowed to corrupt the target user's private key sk_{ID^*}. The **Initialization** and **Setup** are the same as that in the proof of type 1 adversary. The differences lie in the Phase 1 and Phase 2 queries of **Create**(ID,S) as the following: When $ID = ID^*$ for $S \in \mathbb{A}$, $\mathcal{B}$ randomly implicitly sets $r, g_x \in \mathbb{Z}_p$ and embed the unknown item $g^{a^{q+1}}$ to sk_{ID^*} due to the type 2 adversary cannot query the private key sk_{ID^*}. When $S \notin \mathbb{A}$, $\mathcal{B}$ constructs the proof similarly to the proof of type 1 adversary without distinguishing each node $x \in Path(\theta) \cap Path(\theta^*)$ or not. Since the target user ID^* is not revoked in the type 2 adversary, $\mathcal{B}$ can compute the key update message tuk_t as in the scheme for any non-revoked users.

Therefore $\mathcal{B}$ has a non negligible advantage in breaking the $q-1$ assumption in [16] and we obtain Theorem 1 and prove the security of our proposal scheme.

Theorem 2.(Selective Public Verifiability) *If the q-BDHE assumption in [4] hold then the PV-SR-ABE scheme is secure in the sense of selective public verifiability under with a challenge access policy* $(\mathbb{M}^*, \rho^*)$ *and a challenge time period* t^*.

Proof. Suppose $\mathcal{A}_2$ is an adversary with non-negligible advantage ϵ to break scheme in the selective public verifiability. We construct a simulator $\mathcal{B}$ to solve the q-BDHE assumption. Given as input the assumption instance, $\mathcal{B}$ runs $\mathcal{A}_2$ and works as follows.

- **Initialization.** $\mathcal{B}$ first choose the attribute universal set $\mathcal{U}$ and the time set $\mathcal{T}$. The selective game with $\mathcal{A}_2$ outputs a challenge access policy $(\mathbb{M}^*, \rho^*)$ and a challenge time period t^* that it intends to attack. We have that $\mathbb{M}^*$ is an $l \times n$ matrix, where $l \ll q$.
- **Setup.** To generate the system parameter, $\mathcal{B}$ picks randoms $\widetilde{\alpha}, \widetilde{\beta} \in \mathbb{Z}_p$ and sets $\alpha = \widetilde{\alpha}$, $\beta = \widetilde{\beta}$. $\mathcal{B}$ picks the random numbers $b_k, \widetilde{k}, \widetilde{w}, \widetilde{h}_0, \widetilde{u}_1, \widetilde{h}_1, \widetilde{u}, \widetilde{v}, \widetilde{h} \in \mathbb{Z}_p$, $e(u_0, g) = e(g_0, k_0)$ and sets H be the randome oracle controled by the $\mathcal{B}$. $\mathcal{B}$

defines the output of randome oracle H with M_R query is g^{b_R}, where $b_R \in \mathbb{Z}_p$. $\mathcal{B}$ randomly chooses $z_0, z_1, \ldots, z_q \in \mathbb{Z}_p$ and sets $G(\rho^*(j)) = z_0 + z_1\rho^*(j) + z_2\rho^*(j)^2 + \ldots + z_q\rho^*(j)^q$ according to the challenge access policy $(\mathbb{M}^*, \rho^*)$. Using the assumption, $\mathcal{B}$ generates the following public parameters:

$$g = g, g_0 = g^{\frac{a^q}{b_k}}, k_0 = g^{b_k}, w = g^{\widetilde{w}}, v = g^{a^q \prod_{j=1}^{l} G(\rho^*(j))},$$
$$k = g^{\widetilde{k}}, u = g^{\widetilde{u}} \cdot \prod_{(j,k)\in[l,n]} g^{M^*_{j,k}}, u_0 = g^{a^q}, u_1 = g^{a+\widetilde{u}_1},$$
$$h = g^{\widetilde{h}} \cdot \prod_{(j,k)\in[l,n]} g^{-\rho(j)^* M^*_{j,k}}, h_0 = g^{\widetilde{h}_0}, h_1 = g^{-t^*a+\widetilde{h}_1},$$

$\mathcal{B}$ then randomly chooses integers $x_1, \ldots, x_{l+\gamma} \in [0, q-1]$, where γ is the number of digits of identity ID. Aiming at the challenge access policy $(\mathbb{M}^*, \rho^*)$, for the attribute set $S = \{a_1, \ldots, a_{|S|}\}$, if $a_i = \rho^*(j), i = \{1, \ldots, |S|\}, j = \{1, \ldots, l\}$ then $\theta_i = 1$, otherwise, $\theta_i = 0$. When $|S| < l$, then $\theta_i = 0, i = \{|S|, \ldots, l\}$. At last, $\theta_i = ID[i - l + 1]$ for $i \in \{l, \ldots, l + \gamma\}$. $\mathcal{B}$ simply set $n = l + \gamma$ and $F(ID, S) = 1$ if $(\sum_{i=1}^{n} \theta_i \cdot x_i) = 0 \ mod \ q$, otherwise, $F(ID, S) = 0$. Finally, $\mathcal{B}$ gives $\mathcal{A}_1$ the master public key.

– **Challenge.** $\mathcal{B}$ generates challenge header and challenge ciphertext (Hdr^*, c^*) under $(\mathbb{M}^*, \rho^*)$ and t^*. $\mathcal{B}$ randomly chooses random messages $M, M_R \in \mathcal{M}$ and queries the random oracle H. $\mathcal{B}$ computes as the follows:

$$Hdr^* = (e(g, f)^{\widetilde{\alpha}} \cdot M, f, h_0^s = f^{\widetilde{h}_0})$$

$$c^* = (e(g^{a^q}, f)^{\widetilde{\beta}} M_R, H(M_R)^s = f^{b_R}, f, k^s = f^{\widetilde{k}}, k_0^s = f^{b_k},$$
$$(u_1^{t^*} h)^s = f^{\widetilde{u}_1+\widetilde{h}_1}, \{w^{\lambda_\tau} v^{s_\tau}, (u^{\rho^*(\tau)} h)^{-s_\tau}, g^{s_\tau}\}_{\tau\in l})$$

$\mathcal{B}$ sets implicitly $\overrightarrow{y} = (s, \widetilde{y}_2, \widetilde{y}_3, \ldots, \widetilde{y}_n)$, where $\widetilde{y}_2, \ldots, \widetilde{y}_n \in \mathbb{Z}_p$. For each row $\mathcal{B}$ sets implicitly $s_\tau \in \mathbb{Z}_p$ and since $\overrightarrow{\lambda} = \mathbb{M}^* \cdot \overrightarrow{y}$, we have that $\lambda_\tau = \sum_{i\in[n]} s\mathbb{M}^*_{\tau,i}\widetilde{\lambda}_\tau$. Therefore, $\mathcal{B}$ returns the challenge header and challenge ciphertext (Hdr^*, c^*) to $\mathcal{A}_2$.

– **Query.** $\mathcal{B}$ answers the $\mathcal{A}_2$'s queries as follows:

- **Create**(ID, S)**:** When $\mathcal{A}_2$ issues a user key query on (ID, S), $\mathcal{B}$ stores ID in the pre-assigned leaf node θ. We set the user key query to be q_1 times. $\mathcal{B}$ randomly chooses $\widetilde{r} \in \mathbb{Z}_p$ and sets $r = \widetilde{r} + F(ID, S)a$. $\mathcal{B}$ sets the private key $sk_{ID} = g^{\alpha}(u_0^{ID} h_0)^r = g^{\widetilde{\alpha}}(g^{IDa^q} g^{\widetilde{h_0}})^{\widetilde{r}+F(ID,S)a}$ and computes the public verification key vk_{ID} as $g_0^{\beta+rID} = (g^{\frac{a^q}{b_k}})^{\widetilde{\beta}+ID(\widetilde{r}+F(ID,S)a)}$. For each node $x \in Path(\theta)$, $\mathcal{B}$ randomly chooses random $g_x \in \mathbb{G}$, $r_x \in \mathbb{Z}_p$ and generates the long-term transformation key $psk_{ID,S} = ((w^{ID}k)^{r_x}/g_x)u_0^r = (g^{ID\widetilde{w}} g^{\widetilde{k}})^{r_x} \cdot g_x^{-1} \cdot (g^{a^q})^{\widetilde{r}+F(ID,S)a}$. Aiming at each attributes $S_\tau \in S$, $\mathcal{B}$ randomly chooses $r_{x,\tau} \in \mathbb{Z}_p$ and generates the $(u^{S_\tau} h)^{r_{x,\tau}} v^{-r_x}$. When a user key query on (ID, S), if $F(ID, S) \neq 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ returns the private key sk_{ID}, long-term transformation key $psk_{ID,S}$ and public verification key vk_{ID} to $\mathcal{A}_2$.

- **TKeyUp**(t)**:** If $\mathcal{A}_2$ issues a key update query on time period t, $\mathcal{B}$ computes the key update message tuk_t as the scheme.
- **Transform**(ID, S, t^*, c^*)**:** $\mathcal{A}_2$ issues a transformed query on (ID, S, c^*, t^*) and $\mathcal{B}$ computes the transformed ciphertext π as the follows. We set the above query to be q_2 times. For the query $(ID, S = (a_1, \ldots, a_{|S|}), t^*, c^*)$, $\mathcal{B}$ first sets $J(ID, S) = a_1 ID + a_2 ID^2 + \ldots + a_{|S|} a^{|S|}$ and then defines the random number $r = \frac{\prod_{j=1}^{l} G(\rho^*(j))}{J(ID,S)} + F(ID, S)a$. When $S \in \mathbb{A}^*$, for a transform query on c^* and tk^S_{ID,t^*}, if $F(ID, S) \neq 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ computes the transformed ciphertext $\pi = e(u_0^r, g^s)$ as
$$= e((g^{a^q})^{\frac{\prod_{j=1}^{l} G(\rho^*(j))}{J(ID,S)}}, f) e((g^{a^q})^{F(ID,S)a}, f) = e(v^{\frac{1}{J(ID,S)}}, f) e(u_0^{F(ID,S)a}, f)$$
- **Revoke**(ID, t)**:** When $\mathcal{A}_1$ issues a revocation query on an identity ID at time period t, $\mathcal{B}$ adds (ID, t) to the revocation list R.

– **Forgery.** $\mathcal{A}_1$ finishes its query and outputs a guess $\pi^* = e(u_0^{r^*}, f)$
$$= e((g^{a^q})^{\frac{\prod_{j=1}^{l} G(\rho^*(j))}{J(ID^*,S^*)}}, f) e((g^{a^q})^{F(ID^*,S^*)a}, f) = e(v^{\frac{1}{J(ID^*,S^*)}}, f) e(g^{a^{q+1}}, f)^{F(ID,S)}$$
which it believes to be a valid forgery. $\mathcal{B}$ computes $\frac{\pi^*}{e(v^{\frac{1}{J(ID^*,S^*)}}, f)} = e(g^{a^{q+1}}, f)$ as the solution to the q-BDHE problem instance.

According to the proof definition and simulation, there are the following ways to compute the solution to the q-BDHE assumption under $F(ID, S) = 0$ for $(q_s = q_1 + q_2)$ queries and $F(ID^*, S^*) = 1$. We set the number of queries $q = 2q_s$. From the Setup phase, we have $\sum_{i=1}^{n} \theta_i x_i \in [0, n(q-1)]$, where the range contains integers $0q, 1q, 2q, \ldots, (n-1)q$ $(n < q)$. Noted that the pairs (ID_i, S_i) and (ID^*, S^*) for any i different on at least one bit, $F(ID_i, S_i)$ and $F(ID^*, S^*)$ differ on the coefficient of at least one x_j. Thus the forged transformed ciphertext π^* is reducible with success probability $\Pr[Success]=$
$$\begin{aligned}
&\Pr[(\bigwedge_{i=1}^{q_s} F(ID_i, S_i) = 0) \cap F(ID^*, S^*) = 1] \\
=&\Pr[\bigwedge_{i=1}^{q_s} F(ID_i, S_i) = 0 | F(ID^*, S^*) = 1] \Pr[F(ID^*, S^*) = 1] \\
=&(1 - \Pr[\bigvee_{i=1}^{q_s} F(ID_i, S_i) = 1 | F(ID^*, S^*) = 1]) \Pr[F(ID^*, S^*) = 1] \\
\geq&(1 - \sum_{i=1}^{q_s} \Pr[F(ID_i, S_i) = 1 | F(ID^*, S^*) = 1]) \Pr[F(ID^*, S^*) = 1] \\
=&(1 - \frac{q_s}{q}) \cdot \frac{n}{n(q-1)+1} \\
\geq&(1 - \frac{q_s}{q}) \cdot \frac{1}{q} = \frac{1}{4q_s}
\end{aligned}$$
Therefore the adversary $\mathcal{A}_2$ will solve the q-BDHE assumption with the advantage $\frac{\epsilon}{4q_s}$. This completes the proof of Theorem 2 and conclude that the scheme is secure in the sense of selective public verifiability.

5 Conclusion

In this paper, we propose a publicly verifiable server-aided revocable attribute-based encryption scheme that enables any party to verify whether the server behaves as expected. The scheme adopts a "verify-then-decrypt" mechanism, where outsourced results are first verified before being used for decryption, thereby preventing redundant or incorrect decryption attempts. This design enhances the security, transparency, and trustworthiness of server-aided ABE systems while reducing the risk of fraud and misconduct. Nonetheless, several challenges remain, including optimizing the ciphertext size in publicly verifiable server-aided RABE, designing more efficient constructions with shorter transformation keys, and exploring its applicability in practical scenarios.

References

1. Alderman, J., Janson, C., Cid, C., Crampton, J.: Hybrid publicly verifiable computation. In: Topics in Cryptology-CT-RSA 2016 (2016)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE symposium on security and privacy (SP'07) (2007)
3. Boldyreva, A., Goyal, V., Kumar, V.: Identity-based encryption with efficient revocation. In: Proceedings of the 15th ACM conference on Computer and communications security (2008)
4. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Annual international conference on the theory and applications of cryptographic techniques (2005)
5. Cheng, L., Meng, F.: Server-aided revocable attribute-based encryption revised: Multi-user setting and fully secure. In: Computer Security–ESORICS 2021
6. Cui, H., Deng, R.H., Li, Y., Qin, B.: Server-aided revocable attribute-based encryption. In: Computer Security–ESORICS 2016
7. Cui, H., Wan, Z., Wei, X., Nepal, S., Yi, X.: Pay as you decrypt: Decryption outsourcing for functional encryption using blockchain. IEEE Transactions on Information Forensics and Security (2020)
8. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM conference on Computer and communications security. pp. 89–98 (2006)
9. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of abe ciphertexts. In: 20th USENIX Security Symposium (USENIX Security 11) (2011)
10. Hur, J., Noh, D.K.: Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Trans. Parallel Distributed Syst. (2011)
11. Lai, J., Deng, R.H., Guan, C., Weng, J.: Attribute-based encryption with verifiable outsourced decryption. IEEE Transactions on information forensics and security (2013)
12. Miao, Y., Li, F., Li, X., Ning, J., Li, H., Choo, K.K.R., Deng, R.H.: Verifiable outsourced attribute-based encryption scheme for cloud-assisted mobile e-health system. IEEE Transactions on Dependable and Secure Computing (2023)
13. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: Verifiable computation from attribute-based encryption. In: Theory of Cryptography: 9th Theory of Cryptography Conference, TCC 2012 (2012)

14. Qin, B., Zhao, Q., Zheng, D., Cui, H.: Server-aided revocable attribute-based encryption resilient to decryption key exposure. In: Cryptology and Network Security: 16th International Conference (2018)
15. Qin, B., Zhao, Q., Zheng, D., Cui, H.: (dual) server-aided revocable attribute-based encryption with decryption key exposure resistance. Information Sciences (2019)
16. Rouselakis, Y., Waters, B.: Practical constructions and new proof methods for large universe attribute-based encryption (2013)
17. Sahai, A., Waters, B.: Fuzzy identity based encryption. IACR Cryptol. ePrint Arch. (2004)
18. Xu, S., Han, X., Xu, G., Ning, J., Huang, X., Deng, R.H.: An adaptive secure and practical data sharing system with verifiable outsourced decryption. IEEE Transactions on Services Computing (2023)
19. Yang, F., Cui, H., Jing, J.: Generic constructions of server-aided revocable abe with verifiable transformation. In: International Conference on Applied Cryptography and Network Security (2023)
20. Yu, P., Wen, Q., Ni, W., Li, W., Sun, C., Zhang, H., Jin, Z.: Decentralized, revocable and verifiable attribute-based encryption in hybrid cloud system. Wireless Personal Communications (2019)