Attribute-Based Publicly Verifiable Secret Sharing

Liang Zhang[1,3], Xingyu Wu[1], Qiuling Yue*[1], Haibin Kan[2], and Jiheng Zhang[3]

[1] Hainan University, Renmin Road 58, Haikou, China
{zhangliang,22210839000029,yueqiuling}@hainanu.edu.cn
[2] Fudan University, Handan Road 220, Shanghai, China
hbkan@fudan.edu.cn
[3] Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
jiheng@ust.hk

Abstract. Can a dealer share a secret without knowing the shareholders? We provide a positive answer to this question by introducing the concept of an attribute-based secret sharing (AB-SS) scheme. With AB-SS, a dealer can distribute a secret based on attributes rather than specific individuals or shareholders. Only authorized users whose attributes satisfy a given access structure can recover the secret. Furthermore, we introduce the concept of attribute-based publicly verifiable secret sharing (AB-PVSS). An AB-PVSS scheme allows external users to verify the correctness of all broadcast messages from the dealer and shareholders, similar to a traditional PVSS scheme. Additionally, AB-SS (or AB-PVSS) distinguishes itself from traditional SS (or PVSS) by enabling a dealer to generate shares according to an arbitrary monotone access structure. To construct an AB-PVSS scheme, we first implement a decentralized ciphertext-policy attribute-based encryption (CP-ABE) scheme. The proposed CP-ABE scheme offers a smaller ciphertext size and requires fewer computational operations, although it is not fully-fledged as a trade-off. We then incorporate non-interactive zero-knowledge (NIZK) proofs to enable public verification of the CP-ABE ciphertext. Based on the CP-ABE and NIZK proofs, we construct an AB-PVSS primitive. Furthermore, we present an intuitive implementation of optimistic fair exchange based on the AB-PVSS scheme. Finally, we conduct security analysis and comprehensive experiments on the proposed CP-ABE and AB-PVSS schemes. The results demonstrate that both schemes exhibit plausible performance compared to related works.

Keywords: attribute-based secret sharing · decentralized CP-ABE · attribute-based publicly verifiable secret sharing · NIZK

1 Introduction

A secret sharing (SS) scheme [1] is a cryptographic primitive where a dealer commits to a secret, which can only be recovered by a threshold number of shareholders. However, in an SS scheme, a dealer can broadcast invalid shares to deviate from the protocol. To address this issue, a verifiable secret sharing (VSS) scheme [3] ensures that the dealer behaves honestly, as shareholders can verify the validity of the dealer's shares through corresponding proofs. Building on this, a publicly verifiable secret sharing (PVSS) scheme [4,5] allows the dealer to publish shares publicly, enabling any external user to verify the dealer's honesty in a non-interactive manner.

Traditional (PV)SS schemes enable the dealer to share a secret with specific shareholders. Some other works have extended this concept to more complex scenarios where shareholders are organized hierarchically, such as in weighted access structures (WAS) [1,6], disjunctive access structures (DAS) [7], conjunctive access structures (CAS) [8], and compartmented access structures [9,10]. However, these access structures represent particular instances of arbitrary monotone access structures when applied in secret sharing schemes. Consequently, the resulting secret sharing schemes are limited in their applicability. An arbitrary monotone access structure allows a dealer to distribute shares according to more flexible and versatile policies. The question of whether it is possible to construct (PV)SS schemes with more general attribute-based access structures remains an open problem.

In this paper, we fill the gap by proposing an attribute-based secret sharing (AB-SS) scheme and an attribute-based publicly verifiable secret sharing (AB-PVSS) scheme. AB-SS and AB-PVSS schemes adopt a general access structure, providing versatile and fine-grained access control policies. More importantly, AB-SS qualifies a dealer to share a secret according to attributes, rather than concrete shareholders. We construct an AB-SS scheme by studying how SS schemes are leveraged in BSW CP-ABE [36] and achieve an AB-PVSS scheme based on a newly proposed lightweight decentralized CP-ABE. The

decentralized CP-ABE uses secret shares only once[4] in the ciphertext. Furthermore, an encryptor can incorporate an arbitrary number of users as the authorities when generating a ciphertext, making the CP-ABE scheme decentralized. To enable encryptors to prove knowledge of plaintext, we use Sigma protocol [21] and Fiat-Shamir (FS) heuristic [19] to obtain NIZK proofs for the proposed decentralized CP-ABE.

Our contributions. We put forward the concept of attribute-based secret sharing (AB-SS), allowing a dealer to share/hide a secret according to attributes, rather than individuals or shareholders. We focus on attribute-based publicly verifiable secret sharing (AB-PVSS) scheme, a sub-topic of AB-SS. AB-PVSS not only inherits the advantage of AB-SS scheme, but also extends the functionalities of traditional PVSS scheme. To implement an AB-PVSS scheme, we propose a more efficient decentralized CP-ABE scheme. The main idea of the proposed CP-ABE is that we use secret shares only once in *Encrypt* algorithm. Moreover, the CP-ABE is proved to be secure under discrete logarithm (DL) assumption. To prove plaintext knowledge of the proposed CP-ABE ciphertext, we achieve NIZK proofs by incorporating Sigma protocol with Fiat-Shamir heuristic. Comprehensive complexity analysis and experiments are conducted for both the proposed CP-ABE scheme and AB-PVSS scheme. The results show that both schemes outperform respective related works.

2 Related Works

2.1 Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

Ciphertext-policy attribute-based encryption (CP-ABE) [36,41,42] employs access control policy (ACP) in its ciphertext, enabling one-to-many public key encryption paradigm. In most existing CP-ABE schemes, access control policy is implemented using secret sharing techniques (either Shamir secret sharing or linear secret sharing scheme (LSSS) [22]). Hence, the CP-ABE encryption algorithm performs the secret sharing phase and the CP-ABE decryption algorithm executes the secret reconstruction phase.

CP-ABE schemes enable a user to encrypt a ciphertext with attributes without caring about who are the concrete decryptors. The Sahai and Waters research team has led an overwhelming development of CP-ABE. CP-ABE originates from the Fuzzy identity-based encryption (IBE) [11] scheme which was proposed by Sahai and Waters. Fuzzy IBE supports only threshold semantics access control policy, leading to a limitation in applications. Bethencourt, Sahai and Waters (BSW) [36] presented the first expressive CP-ABE construction based on bilinear mapping and Shamir secret sharing. The method on translating an access control policy tree into multiple Shamir secret-sharing instances in their encryption algorithm motivates our work. Herranz1 et al. [13] proposed a constant size threshold CP-ABE scheme.

To eliminate the single point of failure problem in CP-ABE, an intuitive method is to decentralize the authorities who are capable of issuing decryption keys. Lewko and Waters (LW) [41] proposed a multi-authority CP-ABE, where the authorities are autonomous and anyone can become an authority by using an initial set of common reference parameters. They resolve the collusion attack by tying a user's key with a global identifier. We also inherit the idea in the proposed decentralized CP-ABE introduced in Section 5.1. Rouselakis and Waters (RW) extended Lewko's work by supporting large university attributes and attributes can be used without being enumerated during setup. Both Lewko's and Rouselakis's implementations incorporated LSSS to express access control policy in the encryption algorithm.

2.2 Publicly Verifiable Secret Sharing (PVSS)

PVSS is a fundamental primitive in secure multi-party computation (SMPC) applications, especially when fault-tolerance, public communication channels or public verifiability is required. These SMPC applications include but not limited to distributed key generation [14], public distributed randomness beacon [5,15], byzantine agreement [16] and blockchain consensus [17].

Most of the existing PVSS protocols are built upon a VSS scheme, where a public-key encryption (PKE) scheme is used to encrypt the shares. Roughly speaking, VSS is realized via SS and verifiable techniques or non-interactive zero knowledge (NIZK) proofs. Hence, shares in VSS are verifiable. Then, through encrypting shares of VSS schemes via PKE, PVSS allows shares to be delivered and verified

[4] Traditional CP-ABE schemes use secret shares multiple times [36,41,42] in the encryption algorithm, providing the opportunity to reduce ciphertext size and the numbers of cryptograhpic operations

in a public channel. It's interesting to notice that ciphertext-policy attribute-based encryption (CP-ABE) [36,41,42] combines PKE and SS techniques. By intuition, we are motivated to design a (AB-)PVSS scheme based on CP-ABE.

PVSS scheme has been studied for more than two decades. The intuition to achieve a PVSS protocol is to choose a proper encryption algorithm to encrypt VSS shares. Stadler [4] is the first to propose a PVSS protocol, based on ElGamal encryption [24] under number-theoretic settings. Fujisaki et al. [18] uses complicated proof to prove that a share is correctly encrypted via RSA. Schoenmakers [23] resolves PVSS problem using ElGamal encryption under elliptic curves setting. Ruiz et al. [25] construct a PVSS protocol using Paillier's cryptosystem [26] to encrypt shares. Later, Jhanwar et al. [29] improves the Ruiz et al.'s protocol [25] by lowering the computation cost (in both the distribution and reconstruction phase). Heidarvand et al. [27] build a PVSS protocol by leveraging bilinear pairings to prove dealer's honesty. All the above PVSS schemes [4,18,23,25,27,29] have verification complexity $O(nt)$, where n is the number of shareholders and t is the threshold. Cascudo et al. invent a SCRAPE PVSS protocol [5] that achieves verification complexity $O(n)$ using Reed-Solomon error-correcting code [30]. In SCRAPE, the authors put forward two PVSS constructions. Further, Cascudo et al. [39] propose ALBATROSS PVSS based on packed Shamir secret sharing, which aims at generating multiple secrets at a time. ALBATROSS leverages low-degree exponent interpolation (LDEI) in the sharing phase to generate NIZK proofs and $Local_{LDEI}$ in the reconstruction phase to check validity of decrypted shares. The LDEI technique is essentially Sigma protocol and Fiat-Shamir heuristic, which is similar to our construction in this paper. Both ALBATROSS[39] and our protocol hide secret shares as $pk_i^{s_i}$, and generate NIZK proofs using Sigma protocol for the hiding process. HEPVSS[38] uses ElGamal encryption to hide secret share and produce corresponding NIZK proofs. DHPVSS[38] optimizes the hiding process of HEPVSS and uses Reed-Solomon code to verify the correctness of reconstruction phase. Recently, Cascudo et al. [2] introduce qCLPVSS, a PVSS scheme based on class groups and it allows to reconstruct the original secret in a finite field. They point out the distinctive application scenario is to construct MPC protocols. Since the operational cost of class group is relatively high, they omit the concrete analysis about verification complexity. Gentry et al. [12] propose a PVSS scheme, using lattice-based encryption to protect the privacy of shares and using bulletproofs to provide non-interactive zero knowledge proofs.

Table 1 gives a brief comparison on the related works.

Table 1. Comparison of different PVSS schemes

Ref.	Verif.	Tech.	Protocol	ACP	Applications
Stadler [4]	$O(nt)$	ElGamal	PVSS	threshold	key escrow
Fujisaki et al. [18]	$O(nt)$	RSA	PVSS	threshold	key escrow
Schoenmakers [23]	$O(nt)$	ElGamal	PVSS	threshold	voting
Ruiz et al. [25]	$O(nt)$	Paillier	PVSS	threshold	-
Heidarvand et al. [27]	$O(nt)$	Pairing	PVSS	threshold	-
Jhanwar et al. [29]	$O(nt)$	Paillier	PVSS	threshold	-
Cascudo et al. [5]	$O(n)$	Reed-Solomo	PVSS	threshold	beacon
Cascudo et al. [39]	$O(n)$	LDEI	PVSS	threshold	beacon
Cascudo et al.[38]	$O(n)$	ElGamal	PVSS	threshold	resharing
Cascudo et al. [2]	$O(n)$	Class group	PVSS	threshold	DKG, MPC
Gentry et al. [12]	$O(n)$	Lattice	PVSS	threshold	-
ours	$O(n)$	CP-ABE	**AB-PVSS**	**versatile**	fair exchange

2.3 Multi-Level Secret Sharing

In real-world distributed systems, trust is not distributed uniformly over all parties. The degree of trust for distributed parties differs based on each party's authority. When designing secret sharing schemes, shareholders may have various ability to recover the dealer's secret. Hence, different kinds of access structures are employed in building secret schemes.

In the WAS-based secret sharing scheme [1,6], shareholders with higher privilege possess a greater number of shares. WAS-based secret sharing is achieved through a quantitative method. Simmons [28] introduced DAS-based and compartmented secret sharing schemes to distribute asymmetric trust among shareholders. In the DAS-based secret sharing scheme [7], the distributed parties are put in multiple levels. Any at least a threshold number of parties in a same level can recover the secret. If the number

of low-level parties is less than the threshold value, higher-level parties can participate in recovering the secret. Parties at higher levels have more importance or higher authority than the parties at lower levels. In compartmented secret sharing [9,10], parties are divided into disjoint compartments. The secret can be recovered by any threshold number of parties in a compartment. In the CAS-based secret sharing scheme [8], parties are also placed in multiple levels. Every group of a threshold number of parties in a same level must cooperate to recover the secret. Similar to DAS-based secret sharing, higher-level parties can be of help in case low-level parties do not reach the threshold value.

Benaloh et al.'s work [32] proved existence of generalized secret sharing and monotone functions. However, we give a more practical construction using attributes in our AB-SS scheme. AB-SS is a new cryptographic primitive and it provides the feature of anonymity, since a dealer does not necessarily need to know the shareholders.

3 Preliminaries

3.1 Access Control Policy

Definition 1. *(Access Structure [6]) Let* $\mathbf{A} = \{a_1, a_2, \ldots, a_n\}$ *be a set of attribute and* $\mathbb{A}$ *be its power set. A collection* $\Gamma \in \mathbb{A}$ *is an access structure, if it meets the following two conditions:*
(1) non-triviality: if $B \in \Gamma$*, then* $|B| \neq 0$*.*
(2) monotonicity: if $B \in \Gamma$*,* $B \subseteq C$*, then* $C \in \Gamma$*.*
If $B \in \Gamma$*, we call it authorized, and if* $B \notin \Gamma$*, we call it unauthorized.*

Armed with the knowledge of access structure, we will frequently use another related concept, access control policy (ACP), in the subsequent article. ACP can be regarded as an instance of access structure, enabling only qualified users to access specific resources. Access control policy *acp* can be represented using a tree structure, containing attribute strings. Each leaf node of the tree is an attribute string appeared in *acp*. Each non-leaf node represents the threshold gate, described by its direct children and a threshold value. A policy is satisfied only when enough (the threshold-gate value) attributes are combined. The collection of the qualified attributes is called an authorized attribute set. For example, $t\ of\ (attr_1, attr_2, ..., attr_n)$ is an access control policy and at least any t out of the n attributes can make the policy satisfied. In another example, i.e., $2\ of\ ((1\ of\ (attr_1, attr_2)), attr_3, attr_4)$, $[attr_1, attr_3]$ and $[attr_2, attr_4]$ are authorized attribute sets in this example.

3.2 Decentralized CP-ABE

Decentralized ciphertext-policy attribute-based encryption (CP-ABE) scheme is defined as below, slightly modified from previous schemes [41,42]. The main modification is that partial ciphertext C_{u_i} is input of key generation algorithm for authority i.

- $\mathsf{GP} \leftarrow \mathit{GlobalSetup}(\Lambda)$. It takes in the security parameter Λ and outputs global parameters GP.
- $(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathit{AuthSetup}(\mathsf{GP})$. Each authority i takes GP as input to produce a key pair (sk_i, pk_i).
- $C \leftarrow \mathit{Encrypt}(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$. The algorithm takes in GP, a message $s \in \mathbb{G}_0$, an access control policy *acp*, an identity GID, and a set of public keys $\{\mathsf{pk}_i\}$. Denote U be the all the attributes (leaf nodes' value) appeared in *acp*. It outputs a ciphertext $C = (C_0, \{C_{u_i}\})$, where u_i is the attribute value controlled by authority i.
- $K_{u_i} \leftarrow \mathit{KeyGen}(\mathsf{GP}, C_{u_i}, u_i, \mathsf{sk}_i)$. The algorithm takes in GP, an attribute u_i belonging to the authority i, a ciphertext C_{u_i} associated with the attribute u_i and an authority's secret key sk_i. It produces a decryption key K_{u_i}. If a set $\{u_i\}$ satisfies an access control policy *acp*, we say the corresponding set $\{K_{u_i}\}$ is an authorized key set.
- $s \leftarrow \mathit{Decrypt}(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$. The decryption algorithm takes in GP, the ciphertext C, and a collection of decryption keys $\{K_{u_i}\}$. Only if $\{K_{u_i}\}$ is an authorized key set for the access control policy *acp* in C, it outputs the message s.

Definition 2 (*CP-ABE Security Game*)**.** *The CP-ABE security model is defined through the following game*[5]*:*

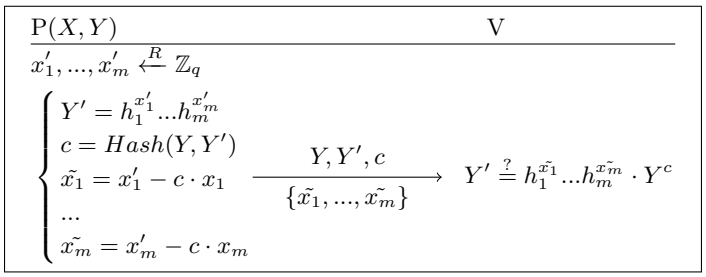
[5] In the CP-ABE defined above, ciphertext should be generated before decryption keys. It is unnecessary to define a query phase before the challenge phase, which is required in previous works [36,41].

- ***Setup***: *The challenger runs GlobalSetup(Λ) to generate global parameters* GP *and obtains a key pair* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *for each authority via AuthSetup*(GP) *algorithm. Then, it sends all public parameters to the adversary.*
- ***Challenge***: *The adversary constructs a challenge access control policy* acp^*. *Then, it sends two equal length messages* (s_0, s_1), acp^* *and* GID *to the challenger. The challenger randomly chooses* $b \in \{0, 1\}$ *and encrypts* s_b *with* acp^* *to obtain the resultant ciphertext* $C^* = (C_0, \{C_{u_i}\})$ *which is sent to the adversary.*
- ***Query***: *By constructing each attribute value* u_i=*"*$attr_j$ *@*$AUTH_i$*", the adversary queries a decryption key* K_{u_i} *from the challenger. Denote all the queried attributes as set* U' =$\{$*"*$attr_j$ *@*$AUTH_i$*"*$\}_{\forall j, \forall i}$. *After current phase, any* $S \subseteq 2^{U'}$ *does not satisfy* acp^*.
- ***Guess***: *The adversary outputs a guess* b' *of* b.

The scheme is breakable if an adversary has a non-negligible advantage in correctly guessing the bit b *in the above security game.*

3.3 Sigma Protocol and NIZK Proof

In generic linear relationship Sigma protocol [21], a prover P can prove zero knowledge of $X = \{x_1, ..., x_m\}$ for Y, where $Y = h_1^{x_1} ... h_m^{x_m}$ and $h_1, ..., h_m$ are generators of $\mathbb{G}$, as follows:



$Hash$ is modeled as a random oracle, as required by Fiat-Shamir heuristic [19]. Y' is called the commitment value, c is the challenge value and $\{\tilde{x_1}, ..., \tilde{x_m}\}$ the response value. The transcript $(Y', c, \{\tilde{x_1}, ..., \tilde{x_m}\})$ is called a conversation between P and V. The transcript is also regarded as NIZK proof proofs_X for proving knowledge of owning X.

A sigma protocol is required to achieve following security properties.

- **Correctness:** If P is honest, honest V always outputs $True$.
- **Knowledge soundness:** Given two correct conversations $(Y', c, \{x_i\})$ and $(Y', c', \{x'_i\})$ where $c \neq c'$, it is efficient to extract the private value X.
- **Special honest verifier zero knowledge (HVZK):** The proof proofs_X reveals nothing information about X.

4 Attribute-based Secret Sharing

Definition 3. *(Attribute-based Secret Sharing) An attribute-based secret sharing scheme (AB-SS) is defined with following two phases.*

(1) ***Distribution*** *Phase: The dealer chooses an ACP* Γ *and takes a secret* $s \in \mathbb{Z}_q$ *as input. Then using a randomized algorithm* $\mathsf{Share}(\Gamma, s) \longrightarrow \{s_1, s_2, \cdots, s_{|\Gamma|}\}$ *to output shares, where* $|\Gamma|$ *is the number of leaf nodes in* Γ.

(2) ***Reconstruction*** *phase: Using a deterministic algorithm* $\mathsf{Recon}(\Gamma, S) \longrightarrow s$ *to reconstruct the secret, if* S *is an authorized attribute set for* Γ, *i.e.,* $S \in \Gamma$.

AB-SS is a secret sharing scheme that allows a dealer to share a secret based on attributes, not individuals or shareholders. We give an AB-SS instance which is inspired by the BSW CP-ABE construction [36], as below.

In the **Distribution** phase, the dealer constructs an ACP tree Γ to share a secret s. Denote U be the set containing all the attribute values of leaf nodes in Γ. Each non-leaf node has a pre-defined threshold value. Then, each (leaf and non-leaf) node in Γ is attached with a value, which is calculated

in a top-down manner. s is attached to the root node R. For each node x, define a polynomial p_x with degree d_x, where d_x is one less than the threshold value. Next, set $p_x(0) = p_{\mathsf{parent}(x)}(\mathsf{index}(x))$ for any other node x, where the parent function returns x's parent and the index function represents x's index value in its parent. Finally, the secret share for attribute u_i is defined as $p_{u_i}(\mathsf{H}(u_i))$ and the values of non-leaf node are discarded. For simplicity, H maps the $|U|$ attributes to integers belong to $[1, |U|]$.

In the **Reconstruction** phase, given an authorized attribute set $S \in \Gamma$, the dealer's secret is recovered in a down-top manner. For each non-leaf node, its value is recovered by its direct children nodes' values, using Lagrange interpolation. Finally, s is recovered.

5 Efficient Decentralized CP-ABE

In this section, we propose an efficient decentralized CP-ABE. The notations are following those in Section 4.

5.1 Construction of the Decentralized CP-ABE

Let $\mathbb{G}_0$ be bilinear groups of prime order q, and let g_0 be generators of $\mathbb{G}_0$. Besides, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$ denotes the bilinear map. Λ is the security parameter, determining the size of the groups. Figure 1 shows the proposed decentralized CP-ABE construction.

Functionality Decentralized CP-ABE algorithms

GlobalSetup(Λ) :

$\mathsf{GP} \leftarrow \mathit{GlobalSetup}(\Lambda) = (g_0, \mathbb{G}_0, H)$

AuthSetup(GP) :

$\mathsf{sk}_i \xleftarrow{R} \mathbb{Z}_q$

$\mathsf{pk}_i \leftarrow g_0^{\mathsf{sk}_i}$

Encrypt$(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$:

$w \xleftarrow{R} \mathbb{Z}_q$

$$C = \begin{cases} C_0 = s \cdot g_0^{w \cdot H(\mathsf{GID})}, \\ \{C_{u_i} = \mathsf{pk}_i^{p_{u_i}(0) \cdot H(u_i)}\}_{\forall u_i \in U} \end{cases}$$

KeyGen$(\mathsf{GP}, C_{u_i}, u_i, \mathsf{sk}_i)$:

$K_{u_i} = C_{u_i}^{1/\mathsf{sk}_i} = g_0^{p_{u_i}(0) \cdot H(u_i)}$

Decrypt$(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$:

for leaf node z:

$F_z = K_{u_i}^{H(\mathsf{GID})/H(u_i)} = g_0^{p_z(0) \cdot H(\mathsf{GID})}$

for non-leaf node x:

$F_x = \prod_{y \in S_x} F_y^{\mu(S'_x)} = ... = g_0^{p_x(0) \cdot H(\mathsf{GID})}$

$s = C_0 / g_0^{p_R(0) \cdot H(\mathsf{GID})}$, where $p_R(0) = w$

Fig. 1. Construction of the decentralized CP-ABE

The *GlobalSetup* algorithm chooses bilinear groups $\mathbb{G}_0$ of prime order q with generator g_0. Also, it defines a hash function $H : \{0,1\}^* \to \mathbb{Z}_q$ which is modeled as a random oracle. The function maps an arbitrary value to a random element in $\mathbb{Z}_q$.

The *AuthSetup* algorithm takes in the global parameters $\mathsf{GP} = \{g_0, \mathbb{G}_0, H\}$, and authority i randomly chooses $\mathsf{sk}_i \in \mathbb{Z}_q$ and calculates the corresponding public key $\mathsf{pk}_i = g_0^{\mathsf{sk}_i}$.

The *Encrypt* algorithm takes in the a secret/plaintext s, an access control policy acp, the global parameters GP, a global identifier GID and public keys $\{\mathsf{pk}_i\}$. Denote T be the access control policy acp tree. Each non-leaf node of T has a pre-defined threshold value. In the algorithm, each node of the access control policy tree is attached to a value and the value is calculated in a top-down manner. As

clarified in Section 3.1, the secret sharing phase of the SS scheme is conducted for each non-leaf node in the *Encrypt* algorithm. Denote U be the set containing all the values of leaf nodes in *acp*. For each node (or attribute value) x, define a polynomial p_x with d_x, where d_x is one less than the threshold value. s is the random value for the root node R. Then, set $p_x(0) = p_{\mathsf{parent}(x)}(\mathsf{index}(x))$ for any other node x, where the parent function returns x's parent and the index function represents x's index value in its parent. Finally, computes the ciphertext $C = (C_0 = s \cdot g_0^{w \cdot H(\mathsf{GID})}, \{C_{u_i} = \mathsf{pk}_i^{p_{u_i}(0) \cdot H(u_i)}\}_{\forall u_i \in U})$, where GID is used to identify current encryption instance.[6] Obviously, each C_{u_i} in the ciphertext corresponds to a unique leaf node.

The *KeyGen* algorithm invoked by authority i generates its key K_{u_i} for attribute value u_i as follows: $K_{u_i} = C_{u_i}^{1/\mathsf{sk}_i} = g_0^{p_{u_i}(0) \cdot H(u_i)}$. Since we propose a decentralized CP-ABE, multiple authorities exist. Here, u_i is used to represent that the attribute value is controlled by authority i.

As opposed to *Encrypt* algorithm, the secret reconstruction phase of the SS scheme is included in the *Decrypt* algorithm, taking ciphertext C, GP and an authorized key set $\{K_{u_i}\}$ as the input. Define $\mu(Z) = \prod_{j,k \in Z, j \neq k} \frac{k}{k-j}$ be the Lagrange coefficient. The *Decrypt* algorithm is a recursive operation from down to top with the following two rules:

- For any leaf node x with attribute value u_i, set recovered value $F_x = K_{u_i}^{H(\mathsf{GID})/H(u_i)} = g_0^{p_x(0) \cdot H(\mathsf{GID})}$. p_x means the randomly chosen polynomial of node x.
- For a non-leaf node x with arbitrary child node z, denote F_z be the recovered value for node z, S_x be an arbitrary authorized attribute set for node x, S'_x is defined as $S'_x = \{\mathsf{index}(z) : z \in S_x\}$. If $\{K_{u_i}\}$ does not comprise of an authorized key set, return $\perp$ for the *Decrypt* algorithm. Otherwise, calculate:

$$\begin{aligned} F_x &= \textstyle\prod_{z \in S_x} F_z^{\mu(S'_x)} \\ &= \textstyle\prod_{z \in S_x} (g_0^{p_z(0) \cdot H(\mathsf{GID})})^{\mu(S'_x)} \\ &= \textstyle\prod_{z \in S_x} (g_0^{p_{\mathsf{parent}(z)}(\mathsf{index}(z)) \cdot H(\mathsf{GID})})^{\mu(S'_x)} \\ &= g_0^{p_x(0) \cdot H(\mathsf{GID})} \end{aligned}$$

Hence, we recursively obtain $g^{w \cdot H(\mathsf{GID})}$ for the root node of tree T. Finally, calculate plaintext $M = C_0 / g_0^{p_R(0) \cdot H(\mathsf{GID})}$, since $p_R(0) = w$.

5.2 Security Analysis

Theorem 1. *Under the DL assumption, the proposed CP-ABE scheme is secure against a static probabilistic polynomial time adversary.*

Proof: We say that a CP-ABE scheme is secure if for any polynomial time adversary, whose attributes set U' do not satisfy the access control policy acp^*, has a negligible advantage in the security game (by Definition 2) played against a challenger. Suppose the adversary can break the DL assumption with advantage of η. The security game goes as follows:

- **Setup**: The challenger runs *GlobalSetup*(Λ) to generate global parameters GP and invokes *AuthSetup*(GP) to obtain a key pair (sk_i, pk_i) for each authority. Then, it sends all public parameters to the adversary.
- **Challenge**: The adversary constructs a challenge access control policy acp^*. Then, it sends two equal length messages (s_0, s_1), acp^* and GID to the challenger. The challenger randomly chooses $b \in \{0, 1\}$ and encrypts s_b with acp^*. The corresponding ciphertext is $C^* = (C_0 = s_b \cdot g_0^{w \cdot H(\mathsf{GID})}, \{C_{u_i} = \mathsf{pk}_i^{p_{u_i}(0) \cdot H(u_i)}\})$ which is sent to the adversary.
- **Query**: By constructing each attribute value u_i, the adversary queries a decryption key $K_{u_i} = \{g_0^{p_{u_i}(0) \cdot H(u_i)}\}$, where i and j are parameters. Denote all the queried attributes as set $U' = \{\text{“attr}_j\text{@AUTH}_i\text{”}\}_{\forall j, \forall i}$. After the current phase, acp^* is satisfied by none of set $S \subseteq 2^{U'}$. These decryption keys $\{K_{u_i}\}_{u_i \in U'}$ are sent to the adversary.
- **Guess**: The adversary makes a guess of b'.

Since w is randomly chosen, $Pr[C_0 = s_0 \cdot g_0^{w \cdot H(\mathsf{GID})}] = Pr[C_0 = s_1 \cdot g_0^{w \cdot H(\mathsf{GID})}] = 1/2$. If the adversary wants to distinguish s_b, it needs to compute $g_0^{w \cdot H(\mathsf{GID})}$. The adversary will succeed if it is able to recover $g_0^{p_R(0)}$ for the root node R given an acp^*. Due to the fact that the calculation of g_0^w is a process from bottom to top of acp^*. For each non-leaf node x, it is associated with a $(t-1)$-degree polynomial p_x,

[6] The use of GID follows decentralized CP-ABE schemes [41,42].

where t is the threshold number required to recover $g_0^{p_x(0)}$. Since $U' \notin acp^*$ after the **Query** phase, there exists a non-leaf node x where less than t decryption keys are provided for the adversary. As is known, less than t points interpolate infinite $(t-1)$-degree polynomials, making it infeasible to defer $g_0^{p_x(0)}$ at node x. Therefore, the adversary cannot recover g_0^w where $w = p_R(0)$ and R denotes the root of acp^*.

Then, the last chance to obtain g_0^w is by breaking the DL assumption so that $p_{u_i}(0)$ can be obtained directly from C_{u_i}. Hence, the probability that the adversary succeeds in guessing $Pr[b' = b]$ is $\frac{1}{2} + \eta$, where η is negligible.

5.3 Complexity of Decentralized CP-ABE schemes

In the CP-ABE *Encrypt* algorithm, C has two parts C_0 and $\{C_{u_i}\}$. C_0 costs 1 exponentiation on $\mathbb{G}_0$ and each C_{u_i} costs n exponentiations on $\mathbb{G}_0$. In the CP-ABE *KeyGen* algorithm, K_{u_i} is generated with only 1 exponentiation on $\mathbb{G}_0$. In the CP-ABE *Decrypt* algorithm, each (leaf or non-leaf) node performs 1 exponentiation on $\mathbb{G}_0$. Table 2 summarizes the computation complexity on decentralized CP-ABE schemes [41,42]. With regard to ciphertext size, our scheme takes up $n+1$ elements on $\mathbb{G}_0$. Table 3 summarizes the ciphertext size on decentralized CP-ABE schemes [41,42], where $\mathbb{G}_1$ is the bilinear mapping group, i.e., $\mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$.

Table 2. Computation complexity

Ref.	*Encrypt*		*KeyGen*	*Decrypt*	
	Exp.	Pair	Exp.	Exp.	Pair
Lewko et al. [41]	$5n+1$	0	2	n	$2n$
Rouselakis et al. [42]	$6n+1$	0	4	n	$3n$
Ours	$n+1$	0	1	$2n-1$	0

Note: n is the number of leaf nodes in the access control policy tree. As a binary tree, the number of non-leaf nodes is $n-1$.

Table 3. Ciphertext size

Ref.	*Decrypt*	
	$\mathbb{G}_0$	$\mathbb{G}_1$
Lewko et al. [41]	$2n$	$n+1$
Rouselakis et al. [42]	$3n$	$n+1$
Ours	$n+1$	0

Note: n is the number of leaf nodes in the access control policy tree.

6 Construction of AB-PVSS

6.1 AB-PVSS Definition

Definition 4. *(Attribute-based Publicly Verifiable Secret Sharing) Let $\Gamma \in \mathbb{A}$ be an access control policy, where $\mathbb{A} = 2^{\{a_1,a_2,\ldots,a_n\}}$. An attribute-based publicly verifiable secret sharing scheme (AB-PVSS) contains four phases, i.e.,* ***Setup***, ***Distribution***, ***Verification***, ***Reconstruction****:*

(1) ***Setup*** *Phase: On input security parameter Λ, global parameters $\mathsf{GP} = \{g_0, \mathbb{G}_0, H\}$ is generated. Each authority generates his key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$. The dealer collects all public keys $\{\mathsf{pk}_i\}_{i\in[1,n]}$.*

(2) ***Distribution*** *Phase: The dealer chooses a Γ and takes a random value $s \in \mathbb{G}_0$. The dealer picks $w \in \mathbb{Z}_q$ and calculates and utilizes a randomized algorithm $\mathsf{Share}(\Gamma, w) \longrightarrow \{w_1, w_2, \cdots, w_{|\Gamma|}\}$ to output shares for each leaf node in Γ. The dealer encrypts s with w to C and encrypts w_j with the corresponding authority's public key pk_i to C_{u_i}, where u_i is the attribute value of a leaf node. The whole result is denoted by C. Also, the dealer generates an NIZK proof proofs_s for proving the correctness of the encryption.*

(3) ***Verification*** *phase: Any external user can verify that C correctly contains valid shares of some secret non-interactively.*

(4) ***Reconstruction*** *phase: Firstly, each authority decrypts each C_{u_i} with his private key sk_i to obtain a decryption key K_{u_i}. Note that any user should be check whether K_{u_i} is correctly computed or not. With enough decryption keys collected to be an authorized key set, a user can recover the secret value s.*

Similar to PVSS scheme [5], an AB-PVSS scheme should satisfy the following three security requirements:

- *Correctness.* If the dealer and the authorities are honest, then all check in **Verification** and **Reconstruction** phases will pass and the secret can be reconstructed in the **Reconstruction** phase with any authorized key set.
- *IND2-Secrecy [27].* Without an authorized key set, no one can learn any information about the secret before **Reconstruction**. It is formally defined by Definition 5.
- *Verifiability.* If the **Verification** phase passes, the C is a valid sharing of some secret with high probability. If the verification in the **Reconstruction** phase passes, K_{u_i} is a correct decryption key generated for attribute u_i.

Definition 5 (**IND2-Secrecy Game**). *An AB-PVSS has **IND2-Secrecy** if for any polynomial time adversary $\mathcal{A}$ corrupting some authorities who cannot produce an authorized key set, $\mathcal{A}$ has negligible advantage in a game with a challenger $\mathcal{C}$.*

1. ***Setup**: $\mathcal{C}$ runs the PVSS **Setup** phase and sends $(\mathsf{GP}, \mathsf{pk}_i, \mathsf{sk}_i)$ to each uncorrupted shareholder P_i. $\mathcal{C}$ sends public information and corrupted authorities' private keys $\{\mathsf{sk}_i\}$ to $\mathcal{A}$.*
2. ***Challenge**: The adversary $\mathcal{A}$ sends two equal length secrets (s_0, s_1) to $\mathcal{C}$. $\mathcal{C}$ randomly chooses $b \leftarrow \{0,1\}$ and runs the **Distribution** phase with secret s_b. It sends all the output to $\mathcal{A}$, along with s_{1-b}.*
3. ***Query**: The adversary $\mathcal{A}$ queries a set of decryption keys, and the whole set should be unauthorized.*
4. ***Guess**: $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$.*

$\mathcal{A}$'s advantage over the game is defined as $|Pr[b = b'] - 1/2|$.

The game is actually similar to the proposed CP-ABE security model in Definition 2.

6.2 NIZK Proofs for CP-ABE Ciphertext

In this section, we demonstrate how to achieve proof of plaintext knowledge for the proposed CP-ABE ciphertext using the Sigma protocol and FS heuristic. Suppose a prover encrypts a secret $s \in \mathbb{G}_0$ to obtain C using the CP-ABE algorithm, as Equations (1) show.

$$C = Encrypt(s, acp, \mathsf{GID}, \mathsf{GP}, \{pk_i\}) = \begin{cases} C_0 = s \cdot g_0^{w \cdot H(\mathsf{GID})}, \\ \{C_{u_i} = \mathsf{pk}_i^{p_{u_i}(0) \cdot H(u_i)}\}_{u_i \in U} \end{cases} \tag{1}$$

Then, the prover composes the commitment value C', which is encrypted from $s' \xleftarrow{R} \mathbb{G}_0$, as Equations (2) show.

$$C' = Encrypt(s', acp, \mathsf{GID}, \mathsf{GP}, \{pk_i\}) = \begin{cases} C'_0 = s' \cdot g_0^{w' \cdot H(\mathsf{GID})}, \\ \{C'_{u_i} = \mathsf{pk}_i^{p'_{u_i}(0) \cdot H(u_i)}\}_{u_i \in U} \end{cases} \tag{2}$$

where $w' (\neq w)$ is randomly chosen from $\mathbb{Z}_q$; p'_R is a randomly chosen polynomial for root node R, and $p'_R(0) = w'$. Next, the prover calculates the Sigma protocol challenge value $c = H_1(C', C)$, where H_1 is a hash function that maps data to an element in $\mathbb{Z}_q$. Then, the response value includes:

$$\tilde{s} = s'/s^c, \tilde{w} \quad = w' - cw, \{\tilde{p}_{u_i}(0) = p'_{u_i}(0) - c \cdot p_{u_i}(0)\}_{u_i \in U}$$

Thus, the NIZK proof $\mathsf{proofs}_s \leftarrow \mathsf{NIZK}(C) = (C', c, (\tilde{s}, \tilde{w}, \{\tilde{p}_{u_i}(0)\}_{u_i \in U}))$.

Any honest external verifier can be convinced that the prover has plaintext knowledge of s, if *CheckCiphertext*, defined by Equations (3), outputs true:

$$\underline{CheckCiphertext(C, \mathsf{proofs}_s):} \\ \begin{cases} C'_0 \stackrel{?}{=} \tilde{s} \cdot g_0^{\tilde{w} \cdot H(\mathsf{GID})} C_0^c \\ \{C'_{u_i} \stackrel{?}{=} \mathsf{pk}_i^{\tilde{p}_{u_i}(0) \cdot H(u_i)} \cdot C_{u_i}^c\}_{u_i \in U} \\ \tilde{w} \stackrel{?}{=} interpolate(\{\tilde{p}_{u_i}(0)\}_{u_i \in U}) \end{cases} \tag{3}$$

The last equation in Equations (3) provides binding relationship of s in C_0 and $\{C_{u_i}\}$. *interpolate* implements the Lagrange polynomial interpolation process from bottom to top according to the *acp* tree.

Lemma 1 (**Completeness**). *A dealer can use the CheckCiphertext algorithm to prove knowledge of the secret s.*

Proof. Given the CP-ABE ciphertext and an NIZK proofs_s, then the Equations (3) is proved to hold as follows.

$$\begin{cases} C'_0 = s' \cdot g_0^{w' \cdot H(\mathsf{GID})} = \tilde{s} \cdot s^c \cdot g_0^{(\tilde{w}+cw)\cdot H(\mathsf{GID})} = \tilde{s} \cdot g_0^{\tilde{w}\cdot H(\mathsf{GID})} C_0^c \\ \{C'_{u_i} = \mathsf{pk}_i^{p'_{u_i}(0)\cdot H(u_i)} = \mathsf{pk}_i^{(\tilde{p}_{u_i}(0)+c\cdot p_{u_i}(0))\cdot H(u_i)} = \mathsf{pk}_i^{\tilde{p}_{u_i}(0)\cdot H(u_i)} \cdot C_{u_i}^c\}_{u_i \in U} \\ \tilde{w} = w' - cw = interpolate(\{p'_{u_i}(0)\}_{u_i \in U}) - c \cdot interpolate(\{\tilde{p}_{u_i}(0)\}_{u_i \in U}) = interpolate(\{\tilde{p}_{u_i}(0)\}_{u_i \in U}) \end{cases}$$

Lemma 2 (**Special knowledge soundness**). *Given two correct conversations with the same commitment and different challenge value, it is efficient to calculate the plaintext s.*

Proof. Given two accepting conversations (C, proofs_s) and (C, proofs'_s), where $\mathsf{proofs}_s = (C', c, (\tilde{s}, \tilde{w}, \{\tilde{p}_{u_i}(0)\}_{u_i \in U}))$ and $\mathsf{proofs}'_s = (C', c', (\tilde{s'}, \tilde{w'}, \{\tilde{p'}_{u_i}(0)\}_{u_i \in U}))$. Note that the two conversations share the same sigma protocol value C'. With $\begin{cases} \tilde{w} = w' - cw, \\ \tilde{w'} = w' - c'w \end{cases}$, one can calculate $w = \frac{\tilde{w'} - \tilde{w}}{c - c'}$. Thus, s can be calculated as : $s = \frac{C}{g_0^{w \cdot H(\mathsf{GID})}}$.

Lemma 3 (**Special HVZK**). *The proof proofs_s reveals nothing information about s.*

Proof. The special HVZK is proved with a *simulator*. We need to prove that the *simulator* can always generate a conversation that is identical with real conversation between P and V. The *simulator* can generate the conversation in arbitrary order. Upon receiving the CP-ABE ciphertext $C = \{C_0, C_{u_i}\}$ and challenge value c, the *simulator* randomly chooses response values $\hat{s} \in \mathbb{G}_0, \hat{w} \in \mathbb{Z}_q$ and multiple polynomials according to the access control policy Γ in C, where $\hat{w}$ invokes the AB-SS $\mathsf{Share}(\Gamma, \hat{w})$ algorithm to obtain $\{\hat{p}_{u_i}(0)\}_{u_i \in U}$ for each leaf node in Γ. Then, generates a conversation as:

$$\mathsf{proofs}'_s = (C' = \{C'_0, C'_{u_i}\}, c, (\hat{s}, \hat{w}, \{\hat{p}_{u_i}(0)\}_{u_i \in U}))$$

where $C'_0 \leftarrow \hat{s} \cdot g_0^{\hat{w} \cdot H(\mathsf{GID})} C_0^c$ and $\{C'_{u_i} \leftarrow \mathsf{pk}_i^{\hat{p}_{u_i}(0)\cdot H(u_i)} \cdot C_{u_i}^c\}_{u_i \in U}$. Obviously, proofs'_s always represents an accepting conversation, as required. Furthermore, since Share is a random algorithm and $\hat{s}, c, \hat{w}, \{\hat{p}_{u_i}(0)\}$ are uniformly distributed in $\mathbb{G}_0$ and $\mathbb{Z}_q$, C'_0 and $\{C'_{u_i}\}$ are uniformly distributed in $\mathbb{G}_0$. That means the *simulator* can always output a proof proofs'_s and the distribution is identical to the real randomized conversation. Hence, proofs_s constructs an NIZK proofs for s in C.

6.3 Construction of AB-PVSS

In this section, we introduce how to build an AB-PVSS scheme based on the proposed CP-ABE algorithm. Firstly, we introduce an algorithm *CheckKey* to check whether a CP-ABE decryption key K_{u_i} is correctly generated with attribute u_i. The *CheckKey* algorithm takes in K_{u_i}, pk_i and u, then outputs true or false. The algorithm costs constant time, i.e., two bilinear pairings.

$$\frac{CheckKey(K_{u_i}, \mathsf{pk}_i, g_0, C_{u_i}) :}{e(K_{u_i}, \mathsf{pk}_i) \stackrel{?}{=} e(C_{u_i}, g_0)}$$

For convenience, we introduce three entities in the AB-PVSS scheme, namely the dealer, authorities and an external verifier/user. The dealer can share a secret using attribute values. The authorities are responsible for generating keys according to attributes. The external verifier/user checks whether the dealer or an authority is honest or not. If secret recovery is required, the external verifier/user acts as the role to collect decryption keys from authorities.

Figure 2 depicts the diagram of data flow in four phases.

1. **Setup** Given the decentralized CP-ABE $\mathsf{GP} \leftarrow$ *GlobalSetup* algorithm is initialized. Each authority i invokes *AuthSetup*(GP) to obtain the key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$. The dealer collects all public keys $\{\mathsf{pk}_i\}$.
2. **Distribution** The dealer constructs an access control policy *acp*. Then, the dealer encrypts his secret s by invoking *Encrypt*$(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$ and obtains ciphertext C. At the same time, the corresponding NIZK proofs $\mathsf{proofs}_s \leftarrow \mathsf{NIZK}(C)$ is attached. Next, the dealer publishes C, proofs_s in the public channel.

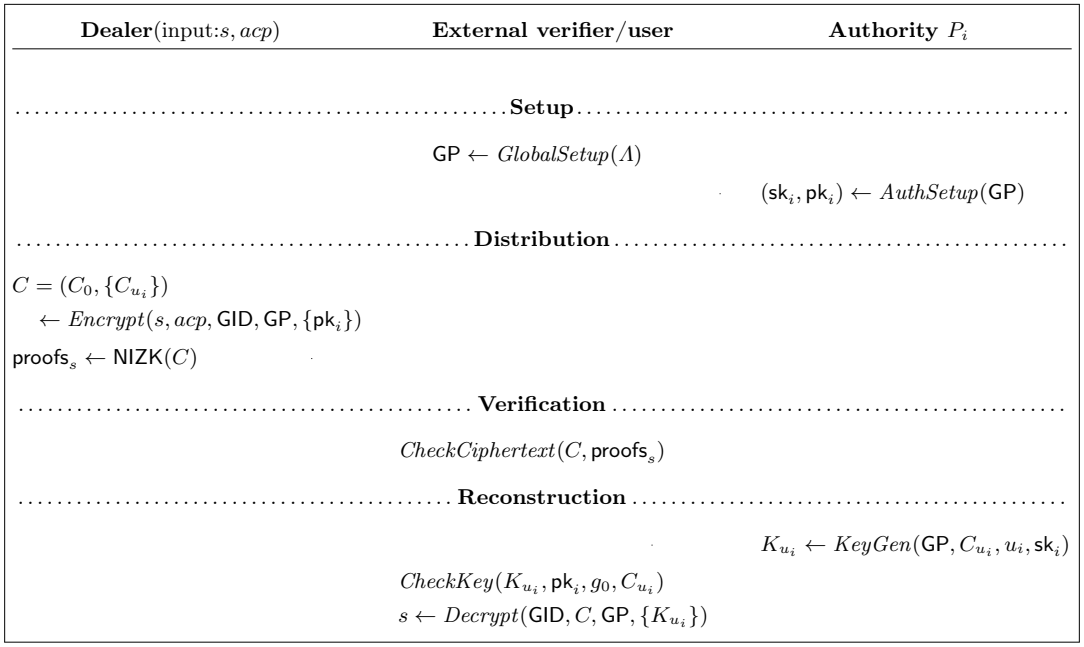


Fig. 2. The proposed AB-PVSS protocol based on CP-ABE

3. **Verification** Any external verifier can check C by $CheckCiphertext(C, \mathsf{proofs}_s)$. If the verfication result is true, the verifier is sure that s is indeed encrypted to C but learns nothing about s.
4. **Reconstruction** Each authority i runs the $KeyGen(\mathsf{GP}, C_{u_i}, u_i, \mathsf{sk}_i)$ algorithm for each attribute u_i to obtain K_{u_i}. Each key K_{u_i} is checked via $CheckKey(K_{u_i}, \mathsf{pk}_i, g_0, C_{u_i})$. After collecting an authorized key set $\{K_{u_i}\}$, any user can invoke $Decrypt(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$ to recover the secret s.

6.4 Security Analysis

This section analyzes the security requirements of the AB-PVSS scheme defined in Section 6.1.

Theorem 2 (*Correctness*)**.** *If the dealer and authorities are honest,* ***Verification*** *phase outputs* true *and* ***Reconstruction*** *phase outputs the dealer's secret s for any honest external verifier/user.*

Proof: In the **Distribution** phase, the honest dealer computes C by encrypting a secret s under access control policy acp and generate NIZK proofs proofs_s. proofs_s will always makes the **Verification** outputs true for any honest external verifier/user due to completeness of Sigma protocols, as Lemma 1 shows. In the **Reconstruction** phase, honest authorities issue correct CP-ABE decryption keys to the external user. Then, the decryption keys $\{K_{u_i}\}$ form an authorized key set, guaranteeing that attribute set $\{u_i\} \in acp$ and $s \leftarrow Decrypt(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$ is successfully recovered.

Theorem 3 (*IND2-secrecy*)**.** *The proposed AB-PVSS is IND2-secrect against a probabilistic polynomial time adversary $\mathcal{A}$, without an authorized key set under the DL assumption and random oracle model.*

Proof. By Lemma 3, we prove that $\mathcal{A}$ has negligible advantage to obtain the secret s from the NIZK proofs_s. Moreover, we prove the $\mathcal{A}$ has negligible advantage in the CP-ABE security game by Theorem 1. The proving process of Theorem 1 is also applicable to the IND2-Secrecy game, since the behaviors of $\mathcal{A}$ are the same in both games given CP-ABE ciphertext. Thus, $\mathcal{A}$ also has negligible advantage in learning information about plaintext s.

Theorem 4 (*Verifiability*)**.** *The protocol is (publicly) verifiable, i.e., the dealer is verifiable in* ***Distribution*** *and authorities are verifiable in* ***Reconstruction****.*

Proof: Theorem 2 has shown that **Verification** phase outputs true if the dealer is honest. If the dealer is dishonest, it can be uncovered and the output is false by the soundness of Sigma protocols, as Lemma 2 shows. Hence, the dealer is verifiable in the **Distribution** phase. We introduce *CheckKey* algorithm to check whether a CP-ABE decryption key K_{u_i} is valid or not. The *CheckKey* is based on bilinear group

pairing, i.e., $e(K_{u_i}, \mathsf{pk}_i) \stackrel{?}{=} e(C_{u_i}, g_0)$. It is infeasible to find a invalid decryption key $K'_{u_i} \neq K_{u_i}$ for a dishonest authority, owing to one-wayness of bilinear mapping. Thus, the authorities are verifiable in the **Reconstruction** phase.

6.5 Complexity of the Proposed AB-PVSS

PVSS scheme usually contains only one instance of secret sharing, which can be expressed with a one-level threshold secret sharing. However, our protocol is attribute-based, enabling multi-level secret sharing. To compare the computation and communication complexity with PVSS schemes, the below analysis only considers a one-level threshold access control policy. Hence, n is the number of authorities/shareholders, t is the threshold value.

Computation Complexity: In the **Distribution** phase, the dealer invokes *Encrypt* algorithm to generate C. It costs $n+1$ exponentiations to produce a ciphertext. The NIZK proofs generation algorithm $\mathsf{NIZK}(C)$ generates $C', c, (\tilde{s}, \tilde{w}, \{\tilde{p}_{u_i}(0)\}_{u_i \in U})$, where C' also takes $n+1$ exponentiations and $\tilde{s}$ takes 1 exponentiation. Hence, the **Distribution** phase takes $2n+3$ exponentiations. In the **Verification** phase, the *CheckCiphertext* costs 2 exponentiations for verifying C_0 and $2n$ exponentiations for verifying all $\{C_{u_i}\}$. Therefore, the **Distribution** phase takes $2n+2$ exponentiations. In the **Reconstruction** phase, the *CheckKey* costs 2 pairings for each decryption key. Besides, the *Decrypt* algorithm is used for recovering secret s, costing t exponentiations. Therefore, the computation complexity of the **Reconstruction** phase costs t exponentiations and $2t$ pairings in total.

Communication Complexity: In the **Distribution** phase, the dealer publishes the ciphertext C of s and the corresponding NIZK proofs $\mathsf{proofs}_s = (C', c, (\tilde{s}, \tilde{w}, \{\tilde{p}_{u_i}(0)\}_{u_i \in U}))$. The proposed CP-ABE ciphertext contains n elements on $\mathbb{G}_0$ and 1 element on $\mathbb{G}_1$. Hence, the **Distribution** contains $2n+3$ elements on $\mathbb{G}$ in total and $n+2$ elements on $\mathbb{Z}_q$. In the **Reconstruction** phase, each authority i publishes a CP-ABE decryption key K_{u_i} for each attribute u. Moreover, only t valid keys are enough for the CP-ABE *Decrypt* algorithm. Hence, **Reconstruction** phase costs t elements on $\mathbb{G}$ for an external user to recover the secret.

Table 4. Computation complexity

Ref.	**Distribution**	**Verification**		**Reconstruction**	
	Exp.	Exp.	Pair	Exp.	Pair
SCRAPE$_{DBS}$[5]	$2n$	n	$2n$	$t+1$	$2t+1$
SCRAPE$_{DDH}$[5]	$4n$	$5n$	–	$5t+3$	–
ALBATROSS[39]	$2n+1$	$2n$	–	$6t+10$	–
HEPVSS[38]	$7n$	$4n$	–	$3t$	–
DHPVSS[38]	$n(n-t+2)+2$	$n(n-t)+4$	–	$5t$	–
Ours	$2n+3$	$2n+2$	–	t	$2t$

Table 5. Communication complexity

Ref.	**Distribution**		**Reconstruction**	
	$\mathbb{G}$	$\mathbb{Z}$	$\mathbb{G}$	$\mathbb{Z}$
SCRAPE$_{DBS}$[5]	$2n$	0	t	0
SCRAPE$_{DDH}$[5]	$4n$	$n+1$	$3t$	$t+1$
ALBATROSS[39]	$2n$	$n+1$	$3t$	$t+1$
HEPVSS[38]	$3n$	$2n$	t	2
DHPVSS[38]	$n+2$	1	$3t$	t
Ours	$2n$	$n+3$	t	0

Table 4 and Table 5 compare the computation and communication complexity of our protocol with state-of-the-art ($O(n)$ verification) PVSS schemes. To further underscore our contribution beyond complexity, it is important to note that previous PVSS protocols [23,27,5,39,38] only enable a dealer to distribute shares among individuals or shareholders. In contrast, our protocol is attribute-based, allowing a dealer to share a secret using attribute values. This capability enables arbitrary monotone access control, making our protocol applicable to more general and diverse scenarios.

7 Discussion

7.1 Notes on the Decentralized CP-ABE

1. The mask of using $H(\mathsf{GID})$ and $H(u_i)$ seems useless in C, since u_i and GID are public and adversaries can invert a value without these masks. However, u_i is an attribute in the access control policy policy which is essential to CP-ABE schemes. And $H(\mathsf{GID})$ is used to identify each CP-ABE ciphertext, resembling previous schemes [41,42].
2. Our CP-ABE outperforms related works, because we use secret shares only once and remove bilinear mapping. Let's take decentralized RW CP-ABE [42] as an example to show how we reduce ciphertext size and computation cost. A random value t_x, two tuple of secret shares $\{\lambda_x\}$ and $\{\omega_x\}$ are generated and used for attribute x in ciphertext. Another random value t is used in two fields $K_{GID,u}$ and $K'_{GID,u}$ in $KeyGen$ algorithm. Hence, the random values $\{t_x\}$ and t can be eliminated by bilinear pairings in $Decrypt$ algorithm. However, in our CP-ABE implementation, the ciphertext C_{u_i} uses each secret share $p_{u_i}(0)$ only once in $Encrypt$ algorithm. Then, the decryption key K_{u_i} inputs the ciphertext C_{u_i} and no new random value is generated in $KeyGen$ algorithm. Thus, we do not need bilinear pairing operation to get rid of randomness in $Decrypt$ algorithm.
3. As a sacrifice, our $KeyGen$ algorithm requires the ciphertext C_{u_i} as an input. That means traditional CP-ABE $Encrypt$ and $KeyGen$ algorithms are independent and enables a decryption key to be useful for future-generated ciphertext. However, our CP-ABE require $Encrypt$ to be invoked before $KeyGen$ is executed for a plaintext. Hence, our scheme is a relaxation of functionality, which is the weakness compared with related works. Strictly speaking, this may violate the concept of CP-ABE for some researchers. We neglect the accuracy of the concept of CP-ABE, because our primary contribution is to introduce AB-(PV)SS and its particularity. But it does impact its usage and security in implementing our AB-PVSS scheme.
4. Although the encryptor can generate keys for decryptors without authorities, our decentralized CP-ABE is still non-trivial in some distributed scenarios. These scenarios include those where a commitment scheme or threshold decryption is required. We depict a fair exchange application based on the decentralized CP-ABE or AB-PVSS in Section 7.3.
5. We implement this decentralized CP-ABE in order to uncover the connection between CP-ABE and PVSS, as presented in Section 7.2.

7.2 Relationship Between AB-PVSS and CP-ABE

To our knowledge, AB-PVSS can be obtained by any CP-ABE along with NIZK. Actually, we have also successfully constructed AB-PVSS with single-authority BSW CP-ABE [36] and multi-authority RW CP-ABE [42], which are less efficient than the proposed AB-PVSS in this paper. That also explains why we construct the more efficient decentralized CP-ABE. As a sacrifice, ciphertext has to be an input of the $KeyGen$ algorithm, indicating restrictions in some applications. AB-PVSS construction might also be obtained based on traditional PVSS and multi-level ACP. Hence, CP-ABE is not a necessity in building AB-PVSS schemes. In the future, we will investigate more about new constructions of AB-PVSS schemes.

7.3 Applications Based on AB-(PV)SS

AB-SS can be useful in applications where SS can be applied, such as key escrow, distributed Elgamal encryption, distributed key generation, distributed randomness beacon or other MPC protocols. However, AB-SS has another plausible property for shareholders, i.e., anonymity. AB-SS allows a dealer to share secret without knowing who are the potential shareholders. For example, in a business company, the CEO's secret, which will impact the company's strategy, needs to be shared to all of the company's stockholders. The CEO can issue shares according to a stockholder's attributes, such as the job field, the nationality, the amount of stocks, the stock holding time, etc.

Also, the AB-PVSS protocol can be used in secure multi-party computation algorithms, such as beacon protocol [5,39], distributed key generation (DKG) [2], where PVSS can apply. To more specifically, the AB-PVSS scheme is suitable in applications where commitments in distributed environment are required. Then, we describe a practical example, optimistic fair exchange (OFE) [31,35], based on AB-PVSS primitive to highlight our contribution. In OFE protocol, two players, Alice (with secret $x \in \mathbb{G}_T$) and Bob (with secret $y \in \mathbb{G}_T$), want to exchange their secrets fairly. It's known that the fairness is hard to guarantee without a trusted third party (TTP) [33]. One TTP may suffers the single point of failure

problem and we can solve the problem by employing decentralized TTPs [34]. In this example, Alice and Bob can choose arbitrary n users as decentralized TTPs, of whom at least $n/2$ are honest. The OFE protocol can be described with two communication rounds. In round-1, Alice and Bob obtains X and Y by executing the AB-PVSS **Distribution** algorithm to hide x and y, respectively, where:

$$acp = 2\ of\ (attr1@Alice, attr1@Bob, n/2\ of\ (attr1@TTP_1, ..., attr1@TTP_n))$$

If both Alice and Bob are honest, each can obtain the other's secret, since **Reconstruction** algorithm can be successfully invoked with Alice's key (generated with attribute "attr1@Alice") and Bob's key (generated with attribute "attr1@Bob") in round-2. If one player (say Bob) is malicious, honest TTPs can be of help for Alice to recover y, since the *acp* can be satisfied by the set which is composed of honest TTPs' and Alice's attributes. In this example, AB-PVSS is used to build decentralized TTPs with property of autonomity (the TTPs do not need to negotiate with each other), optimism (the TTPs are involved in only when arbitration requires) and even statelessness (the TTPs do not need to store variables for an OFE instance).

8 Implementations

We implement the decentralized CP-ABE scheme and AB-PVSS scheme with Charm-Crypto library [40], which is a framework for constructing cryptographic schemes. It provides Python programming language interfaces. The Charm-Crypto framework relies on the GMP (GNU multiple precision) arithmetic library and the PBC (pairing-based cryptography) library written in C language. Charm-Crypto also provides classic cryptographic primitives as its built-in examples, including the BSW CP-ABE [36], LW CP-ABE [41] and RW CP-ABE [42]. Based on the built-in example, we first make it compatible with the threshold-based access control policy. Then we implement our proposed decentralized CP-ABE. Further, we implement our AB-PVSS and some of above mentioned PVSS schemes [5,39]. The experiments are conducted on AWS Ubuntu 18.04, 4 GB RAM, with Python 3.6.9 and curve "SS512".

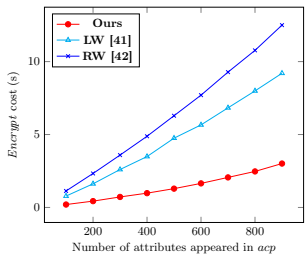


Fig. 3. *Encrypt* cost of decentralized CP-ABE

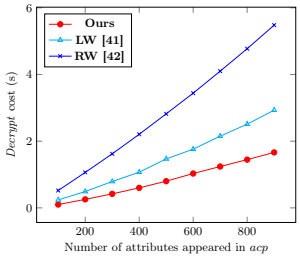


Fig. 4. *Decrypt* cost of decentralized CP-ABE

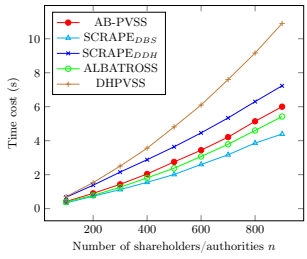


Fig. 5. Distribution cost

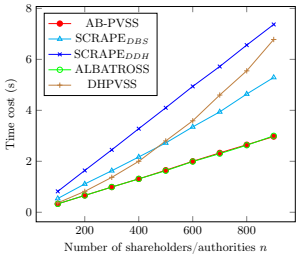


Fig. 6. Verification cost

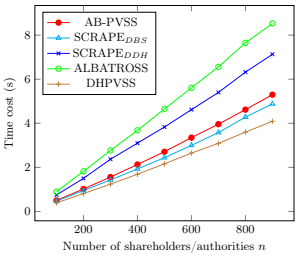


Fig. 7. Reconstruction cost

We then compare the performance of the proposed CP-ABE with other decentralized CP-ABE schemes, i.e., LW CP-ABE [41] and RW CP-ABE [42]. Figure 3 and Figure 4 depict the *Encrypt* and the *Decrypt* time cost, respectively. Though our decentralized CP-ABE scheme is not fully-fledged, these figures indicate that our scheme outperforms previous constructions.

We then evaluate the performance with the proposed AB-PVSS scheme by downgrading the AB-PVSS to a PVSS scheme and compare it with other PVSS schemes [5,39,38]. Figure 5, Figure 6 and Figure 7 show the concrete computation overhead of the **Distribution** phase (by the dealer), the **Verification** phase (by a verifier), and the **Reconstruction** phase (by a user), respectively. SCRAPE$_{DBS}$ has the lowest distribution time cost, which is identical to Table 4. It can be seen that our AB-PVSS and ALBATROSS have the lowest verification overhead. However, ALBATROSS has the highest reconstruction overhead. DHPVSS has the lowest reconstruction overhead, but it requires superlinear complexity in the distribution and verification phase due to point evaluations with a random (n-t-1)-degree polynomial. HEPVSS is not shown in the figures, as it does not appear to be optimized in either phase by Table 4.

9 Conclusion

We achieve two favorable functionalities in the field of secret sharing schemes by proposing the concept of AB-SS. They are: 1) a dealer can share a secret with an arbitrary monotone access structure; 2) a dealer also can share a secret without knowing the shareholders. We give the definition of AB-SS rigorously and present an AB-SS scheme by adopting some ideas from BSW CP-ABE. Then, we build an efficient decentralized CP-ABE by reducing the times of secret shares usage in *Encrypt* algorithm. Further, NIZK proofs are attached to prove plaintext knowledge for the proposed CP-ABE ciphertext. The NIZK proofs are obtained by leveraging generic linear Sigma protocol and Fiat-Shamir heuristic. Finally, AB-PVSS scheme is formally defined and an AB-PVSS scheme is constructed by incorporating the proposed CP-ABE scheme and NIZK proofs.

References

1. Shamir, A. How to share a secret. *Comm. of the ACM*, 1979, 22(11), 612–613.
2. Cascudo, I., & David, B. Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In *Eurocrypt'24*, 2024, pp. 216–248.
3. Feldman, P. A practical scheme for non-interactive verifiable secret sharing. In *FOCS'87*, 1987, pp. 427–438.
4. Stadler, M. Publicly verifiable secret sharing. In *Eurocrypt'96*, 1996, pp. 190–199.
5. Cascudo, I., & David, B. SCRAPE: Scalable randomness attested by public entities. In *ACNS'17*, 2017, pp. 537–556.
6. Beimel, A., Tassa, T., & Weinreb, E. Characterizing ideal weighted threshold secret sharing. In *TCC'05*, 2005, pp. 600–619.
7. Belenkiy, M. Disjunctive multi-level secret sharing. *Cryptology ePrint Archive*, 2008.
8. Tassa, T. Hierarchical threshold secret sharing. *Journal of cryptology*, 2007, 20(2), 237–264.
9. Tassa, T., & Dyn, N. Multipartite secret sharing by bivariate interpolation. *Journal of Cryptology*, 2009, 22(2), 227–258.
10. Chen, Q., Tang, C., & Lin, Z. Efficient explicit constructions of multipartite secret sharing schemes. *IEEE TIT*, 2021, 68(1), 601–631.
11. Sahai, A., & Waters, B. Fuzzy identity-based encryption. In *Eurocrypt'05*, 2005, pp. 457–473.
12. Gentry, C., Halevi, S., & Lyubashevsky, V. Practical non-interactive publicly verifiable secret sharing with thousands of parties. In *Eurocrypt'22*, 2022, pp. 458–487.
13. Herranz, J., Laguillaumie, F., & Ràfols, C. Constant size ciphertexts in threshold attribute-based encryption. In *PKC'10*, 2010, pp. 19–34.
14. Fouque, P. A., & Stern, J. One round threshold discrete-log key generation without private channels. In *PKC'01*, 2001, pp. 300–316.
15. Syta, E., Jovanovic, P., Kogias, E. K., Gailly, N., Gasser, L., Khoffi, I., Fischer, M. J., & Ford, B. Scalable bias-resistant distributed randomness. In *SP'17*, 2017, pp. 444–460.
16. Bessani, A. N., Alchieri, E. P., Correia, M., & Fraga, J. S. DepSpace: a Byzantine fault-tolerant coordination service. In *Eurosys'08*, 2008, pp. 163–176.
17. Kiayias, A., Russell, A., David, B., & Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Crypto'17*, 2017, pp. 357–388.
18. Fujisaki, E., & Okamoto, T. A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In *Eurocrypt'98*, 1998, pp. 32–46.
19. Fiat, A., & Shamir, A. How to prove yourself: Practical solutions to identification and signature problems. In *Eurocrypt'86*, 1986, pp. 186–194.
20. Zhang, L., Qiu, F., Hao, F., & Kan, H. 1-Round Distributed Key Generation With Efficient Reconstruction Using Decentralized CP-ABE. *IEEE TIFS*, 2022, 17, 894–907.
21. Damgård, I. On Σ-protocols. *Lecture Notes, University of Aarhus, Department for Computer Science*, 2002.
22. Beimel, A., Ben-Efraim, A., Padró, C., & Tyomkin, I. Multi-linear secret-sharing schemes. In *TCC'14*, 2014, pp. 394–418.

23. Schoenmakers, B. A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic. In *Crypto'99*, 1999, pp. 148–164.
24. ElGamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE TIT*, 1985, 31(4), 469–472.
25. Ruiz, A., & Villar, J. L. Publicly verifiable secret sharing from Paillier's cryptosystem. In *SAC'05*, 2005.
26. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt'99*, 1999, pp. 223–238.
27. Heidarvand, S., & Villar, J. L. Public verifiability from pairings in secret sharing schemes. In *SAC'09*, 2009, pp. 294–308.
28. Simmons, G. J. How to (really) share a secret. In *Asiacrypt'98*, 1988, pp. 390–448.
29. Jhanwar, M. P., Venkateswarlu, A., & Safavi-Naini, R. Paillier-based publicly verifiable (non-interactive) secret sharing. *Designs, Codes and Cryptography*, 2014, 73, 529–546.
30. McEliece, R. J., & Sarwate, D. V. On sharing secrets and Reed-Solomon codes. *Communications of the ACM*, 1981, 24(9), 583–584.
31. Zhang, L., Kan, H., Qiu, F., & Hao, F. A Publicly Verifiable Optimistic Fair Exchange Protocol Using Decentralized CP-ABE. *The Computer Journal*, 2024, 67(3), 1017–1029.
32. Benaloh, J., & Leichter, J. Generalized secret sharing and monotone functions. In *Crypto'88*, 1990.
33. Pagnia, H., & Gärtner, F. C. On the impossibility of fair exchange without a trusted third party. Technical Report, Darmstadt University of Technology, Department of Computer Science, Darmstadt. 1999.
34. Küpçü, A. Distributing trusted third parties. *ACM SIGACT News*, 2013, 44(2), 92–112.
35. Asokan, N., Shoup, V., & Waidner, M. Optimistic fair exchange of digital signatures. In *Eurocrypt'98*, 1998, pp. 591–606.
36. Bethencourt, J., Sahai, A., & Waters, B. Ciphertext-Policy Attribute-Based Encryption. In *SP'07*, 2007, pp. 321–334.
37. Berrut, J.-P., & Trefethen, L. N. Barycentric lagrange interpolation. *SIAM Review*, 2004, 46(3), 501–517.
38. Cascudo, I., David, B., Garms, L., & Konring, A. YOLO YOSO: fast and simple encryption and secret sharing in the YOSO model. In *Asiacrypt'21*, 2022, pp. 651–680.
39. Cascudo, I., & David, B. ALBATROSS: publicly attestable batched randomness based on secret sharing. In *Asiacrypt'20*, 2020, pp. 311–341.
40. Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M., & Rubin, A. D. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, 2013, 3, 111–128.
41. Lewko, A., & Waters, B. Decentralizing attribute-based encryption. In *Eurocrypt'11*, 2011, pp. 568–588.
42. Rouselakis, Y., & Waters, B. Efficient statically-secure large-universe multi-authority attribute-based encryption. In *FC'15*, 2015, pp. 315–332.