Cybersecurity

RESEARCH

Open Access

Attribute-based publicly verifiable secret sharing

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

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 build an AB-PVSS scheme, we first implement a decentralized ciphertext-policy attribute-based encryption (CP-ABE) scheme, though not a fully-fledged one.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.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

Introduction

A secret sharing (SS) scheme (Shamir 1979) 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 (Feldman 1987) 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 (Ruiz and Villar 2005; Schoenmakers 1999; Heidarvand and Villar 2009; Jhanwar et al. 2014; Stadler 1996; Cascudo and David 2017, 2020; Cascudo et al. 2022; Cascudo and David 2024; Gentry et al. 2022; Fujisaki and Okamoto 1998) allows the dealer to publish shares publicly, enabling any external user to verify the dealer's honesty in a non-interactive manner. PVSS is fundamental 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 public distributed randomness beacon (Cascudo and David 2017; Syta et al. 2017), byzantine agreement (Bessani et al. 2008), blockchain consensus (Kiayias et al. 2017) and fair exchange (Avoine and Vaudenay 2004; Zhang et al. 2024).

†L. Zhang, X. Wu contributed equally to this work.

*Correspondence:
Qiuling Yue
yueqiuling@hainanu.edu.cn
[1] School of Cyberspace Security (School of Cryptology), Hainan University, Renmin Road 58, Haikou 570228, China
[2] Industrial Engineering and Decision Analytics, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong, China
[3] School of Computer Science, Fudan University, Handan Road 220, Shanghai 200433, China

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) (Shamir 1979; Beimel et al. 2005), disjunctive access structures (DAS) (Belenkiy 2008), conjunctive access structures (CAS) (Tassa 2007), and compartmented access structures (Tassa and Dyn 2009; Chen et al. 2021) 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 (Bethencourt et al. 2007)and achieve an AB-PVSS scheme based on a newly proposed lightweight decentralized CP-ABE. The decentralized CP-ABE uses secret shares only once[1] 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. Different from traditional decentralized CP-ABE schemes (Lewko and Waters 2011; Rouselakis and Waters 2015) our proposed scheme requires the ciphertext as an input to the key-generation algorithm. This design limits key reusability, where keys cannot be applied interchangeably across distinct ciphertexts. Furthermore, in order to enable encryptors to prove knowledge of plaintext, we use Sigma protocol (Damgård 2002)and Fiat-Shamir (FS) heuristic (Fiat and Shamir 1986)to obtain NIZK proofs for the proposed decentralized CP-ABE.

The contributions are summarized as follows:

- 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. Further, we define attribute-based publicly verifiable secret sharing (AB-PVSS), which not only inherit the advantage of AB-SS scheme, but also extends the functionalities of PVSS schemes.
- 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. 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.

Preliminaries

Access control policy

Definition 1 *(Access Structure* (Beimel et al. 2005)*)* 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) if $B \in \Gamma$, then $|B| \neq 0$. (2) 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.

Decentralized CP-ABE

Decentralized ciphertext-policy attribute-based encryption (CP-ABE) scheme is defined as below, slightly modified from previous schemes (Lewko and Waters 2011; Rouselakis and Waters 2015) The main modification is that partial ciphertext C_{u_i} is input of key generation algorithm for authority i.

[1] Traditional CP-ABE schemes use secret shares multiple times (Bethencourt et al. 2007; Lewko and Waters 2011; Rouselakis and Waters 2015) in the encryption algorithm, providing the opportunity to reduce ciphertext size and the numbers of cryptograhpic operations.

- $\mathsf{GP} \leftarrow \mathit{GlobalSetup}(\Lambda)$. 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 $(\mathsf{sk}_i, \mathsf{pk}_i)$.
- $C \leftarrow \mathit{Encrypt}(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$. 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 $\{\mathsf{pk}_i\}$. Let U denote all the attributes (leaf nodes' value) that appear in acp. It outputs a ciphertext $C = (C_0, \{C_{u_i}\})$, where $u_i \in U$ 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.

The security game is defined by Definition 2.

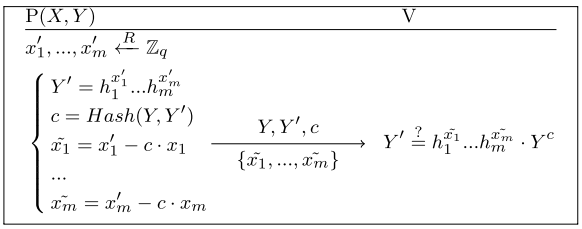
Definition 2 *(*Security Game*)* The decentralized CP-ABE security model is defined through the following game[2]:

- **Setup**: The challenger runs $\mathit{GlobalSetup}(\Lambda)$ to generate global parameters GP and obtains a key pair $(\mathsf{sk}_i, \mathsf{pk}_i)$ for each authority via $\mathit{AuthSetup}(\mathsf{GP})$ 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 $U' = \{\text{"attr}_j\text{@AUTH}_i\text{"}\}_{\forall j, \forall i}$. 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.

Sigma protocol and NIZK proof

In generic linear relationship Sigma protocol (Damgård 2002) 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 (Fiat and Shamir 1986) 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 conveys no information about X other than the validity of the statement Y.

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] In the CP-ABE defined in Sect. 2.2, 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 (Bethencourt et al. 2007; Lewko and Waters 2011)

(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 (Bethencourt et al. 2007) 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. 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. Firstly, s is attached to the root node R. Subsequently, through the top-down manner, we can calculate a binding value for each (leaf and non-leaf) node. 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 (Berrut and Trefethen 2004). Finally, s is recovered.

Functionality Decentralized CP-ABE algorithms

$\underline{GlobalSetup(\Lambda):}$

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

$\underline{AuthSetup(\mathsf{GP}):}$

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

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

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

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

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

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

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

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

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:

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

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

Fig. 1 Construction of the decentralized CP-ABE

Efficient decentralized CP-ABE

Construction

In this section, we propose an efficient decentralized CP-ABE. The notations are following those in Sect. 3. Let $\mathbb{G}_0$ be a group of prime order q, and let g_0 be generators of $\mathbb{G}_0$. Λ is the security parameter, determining the size of the groups. Figure 1 shows the proposed decentralized CP-ABE construction.

The *GlobalSetup* algorithm chooses group $\mathbb{G}_0$ of prime order q with generator g_0. Also, it defines a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ 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 Sect. 2.1, 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 u_i represents each attribute in the access control policy, $H(u_i)$ binds to the attribute u_i and $H(\mathsf{GID})$ serves to uniquely identify each CP-ABE ciphertext.[3]

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.

[3] The use of GID follows decentralized CP-ABE schemes (Lewko and Waters 2011; Rouselakis and Waters 2015).

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:

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

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 $p_R(0) = w$.

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 $(\mathsf{sk}_i, \mathsf{pk}_i)$ 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, 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.

Construction of AB-PVSS

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 can 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 (Cascudo and David 2017), an AB-PVSS scheme consists of the following three main roles:

- **Dealer** generates the encrypted share components C_{u_i} for a secret value $s \in \mathbb{G}_0$ under a given monotone access structure, using the corresponding authorities' public keys. In addition, the dealer produces a non-interactive zero-knowledge proof proofs_s attesting to the correctness of the encryption.
- **Authority (or Shareholder)** is responsible for decrypting the encrypted share C_{u_i} associated with its managed attribute u_i and subsequently returning the derived decryption value K_{u_i}.
- **User** verifies the validity of the encrypted shares C_{u_i}; collects an authorized set of decryption keys K_{u_i} to reconstruct the secret value s.

Similar to PVSS scheme (Cascudo and David 2017), 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* (Heidarvand and Villar 2009). 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 $\mathcal{A}$.
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.

Adversarial model. In our scheme, the adversary is modeled as a probabilistic polynomial-time (PPT) algorithm. For an AB-PVSS protocol defined over an access structure τ, we consider a static adversarial model in which the adversary may corrupt a subset of authorities who controlling a set of attributes Q, provided that Q does not satisfy the access structure τ.

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.* □

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

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. □

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 (Bethencourt et al. 2007).

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

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 $\mathsf{GP} \leftarrow \mathit{GlobalSetup}$ algorithm is initialized. Each authority i invokes $\mathit{AuthSetup}(\mathsf{GP})$ 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 $\mathit{Encrypt}(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$ 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.
3. **Verification** Any external verifier can check C by $\mathit{CheckCiphertext}(C, \mathsf{proofs}_s)$. If the verification 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 $\mathit{KeyGen}(\mathsf{GP}, C_{u_i}, u_i, \mathsf{sk}_i)$ algorithm for each attribute u_i to obtain K_{u_i}. Each key K_{u_i} is checked via $\mathit{CheckKey}(K_{u_i}, \mathsf{pk}_i, g_0, C_{u_i})$. After collecting an authorized key set $\{K_{u_i}\}$, any user can invoke $\mathit{Decrypt}(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$ to recover the secret s.

Functionality The proposed AB-PVSS based on CP-ABE

Setup Phase:

$\mathsf{GP} \leftarrow \mathit{GlobalSetup}(\Lambda)$

$(\mathsf{sk}_i, \mathsf{pk}_i) \leftarrow \mathit{AuthSetup}(\mathsf{GP})$

Distribution Phase:

$C = (C_0, \{C_{u_i}\}) \leftarrow \mathit{Encrypt}(s, acp, \mathsf{GID}, \mathsf{GP}, \{\mathsf{pk}_i\})$

$\mathsf{proofs}_s \leftarrow \mathsf{NIZK}(C)$

Verification Phase:

$\mathit{CheckCiphertext}(C, \mathsf{proofs}_s)$

Reconstruction phase:

$K_{u_i} \leftarrow \mathit{KeyGen}(\mathsf{GP}, C_{u_i}, u_i, \mathsf{sk}_i)$

$\mathit{CheckKey}(K_{u_i}, \mathsf{pk}_i, g_0, C_{u_i})$

$s \leftarrow \mathit{Decrypt}(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$

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

Security analysis

This section analyzes the security requirements of the AB-PVSS scheme defined in Sect. 5.1.

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* $s \leftarrow \mathit{Decrypt}(\mathsf{GID}, C, \mathsf{GP}, \{K_{u_i}\})$ *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.* □

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 1 and Table 2 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 (Schoenmakers 1999; Heidarvand and Villar 2009; Cascudo and David 2017, 2020; Cascudo et al. 2022) only enable

Table 2 Communication complexity

Ref.	Distribution		Reconstruction	
	$\mathbb{G}$	$\mathbb{Z}$	$\mathbb{G}$	$\mathbb{Z}$
SCRAPE$_{DBS}$ Cascudo and David (2017)	$2n$	0	t	0
SCRAPE$_{DDH}$ Cascudo and David (2017)	$4n$	$n+1$	$3t$	$t+1$
ALBATROSS Cascudo and David (2020)	$2n$	$n+1$	$3t$	$t+1$
HEPVSS Cascudo et al. (2022)	$3n$	$2n$	t	2
DHPVSS Cascudo et al. (2022)	$n+2$	1	$3t$	t
Ours	$2n$	$n+3$	t	0

Table 1 Computation complexity

Ref.	Distribution	Verification		Reconstruction	
	Exp	Exp	Pair	Exp	Pair
SCRAPE$_{DBS}$ Cascudo and David (2017)	$2n$	n	$2n$	$t+1$	$2t+1$
SCRAPE$_{DDH}$ Cascudo and David (2017)	$4n$	$5n$	–	$5t+3$	–
ALBATROSS Cascudo and David (2020)	$2n+1$	$2n$	–	$6t+10$	–
HEPVSS Cascudo et al. (2022)	$7n$	$4n$	–	$3t$	–
DHPVSS Cascudo et al. (2022)	$n(n-t+2)+2$	$n(n-t)+4$	–	$5t$	–
Ours	$2n+3$	$2n+2$	–	t	$2t$

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.

Implementations

We implement the decentralized CP-ABE scheme and AB-PVSS scheme with Charm-Crypto library (Akinyele et al. 2013), 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 (Bethencourt et al. 2007), LW CP-ABE (Lewko and Waters 2011) and RW CP-ABE (Rouselakis and Waters 2015). 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 (Cascudo and David 2017, 2020). The experiments are conducted on AWS Ubuntu 18.04, 4 GB RAM, with Python 3.6.9 and curve "SS512".

We then compare the performance of the proposed CP-ABE with other decentralized CP-ABE schemes, i.e., LW CP-ABE (Lewko and Waters 2011) and RW CP-ABE (Rouselakis and Waters 2015). Figure 3 and Fig. 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.

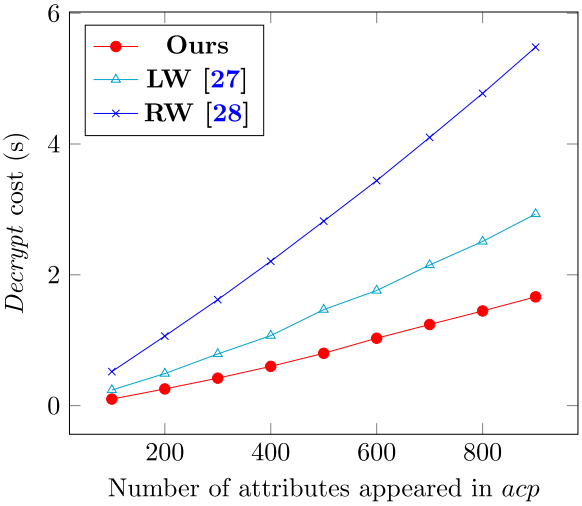


Fig. 4 *Decrypt* cost of decentralized CP-ABE

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 (Cascudo and David 2017, 2020; Cascudo et al. 2022). Figure 5, Fig. 6 and Fig. 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 1. 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

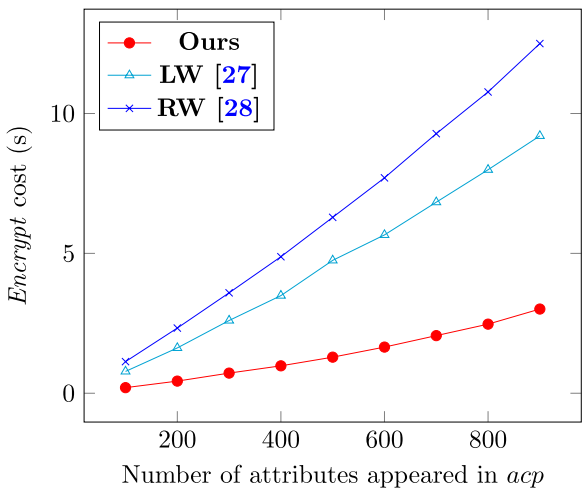


Fig. 3 *Encrypt* cost of decentralized CP-ABE

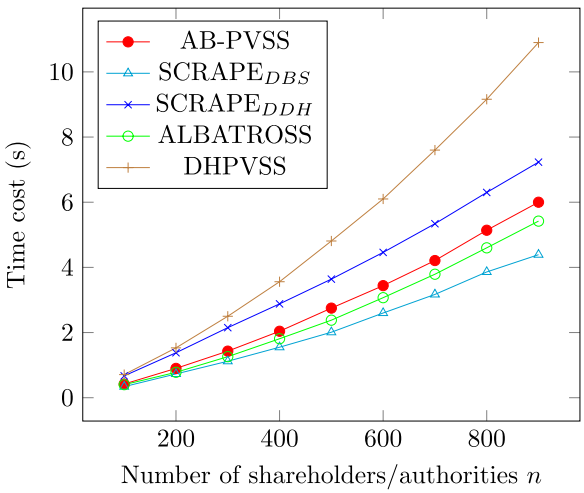


Fig. 5 Distribution cost

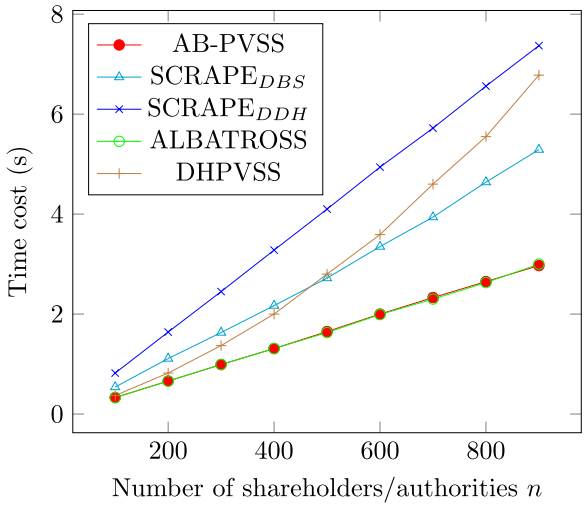


Fig. 6 Verification cost

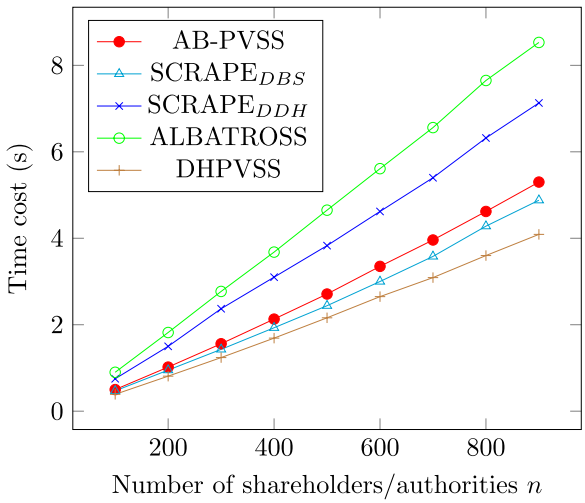


Fig. 7 Reconstruction cost

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 1.

Discussions

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 which is essential to CP-ABE schemes. And $H(\mathsf{GID})$ is used to identify each CP-ABE ciphertext, resembling previous schemes (Lewko and Waters 2011; Rouselakis and Waters 2015).
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 (Rouselakis and Waters 2015) 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. (Note that the bilinear pairing operation is required only in the proposed AB-PVSS *CheckKey* algorithm, but not in the CP-ABE scheme.)
3. As a trade-off, 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. This design leads to reduced key reusability where keys can not be used interchangeably across different ciphertexts. Even if the CP-ABE keys associated with a particular sharing instance (GID) are compromised, the confidentiality of all previously generated ciphertexts remains intact, thereby ensuring forward secrecy. 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.
5. Fundamentally, the proposed decentralized CP-ABE scheme operates as a distributed ElGamal protocol (Zhang et al. 2025) with a monotone access structure, enabling ciphertext decryption for users whose keys satisfy the specified access conditions. Further,

we implement this decentralized CP-ABE in order to uncover the connection between CP-ABE and PVSS, as presented in Sect. 7.

6. 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 (Bethencourt et al. 2007) and multi-authority RW CP-ABE (Rouselakis and Waters 2015), 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. An important direction for extending our decentralized CP-ABE design is the integration of efficient attribute-revocation mechanisms, as demonstrated in the revocable ABE schemes (Sethi et al. 2021) and Li et al. (2025). Incorporating such revocation idea into our decentralized construction would not only enhance its practicality but also facilitate the development of new secret-sharing frameworks and attribute-based publicly verifiable secret re-sharing.

Conclusion

We propose the concept of attribute-based secret sharing (AB-SS), where two favorable functionalities are acquired. 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, we formally define and implement an attribute-based secret sharing (AB-PVSS) scheme by integrating the proposed CP-ABE scheme with NIZK proofs.

Acknowledgements
We thank anonymous reviewers for helpful discussions.

Author Contributions
Liang Zhang contributed to the methodology, software implementation, and investigation, and prepared the original draft of the manuscript, as well as acquiring funding. Xingyu Wu and Qiuling Yue participated in the discussions and contributed to manuscript review and editing. Haibin Kan provided funding acquisition, resources, and supervision. Jiheng Zhang was responsible for funding acquisition and project administration. All the authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China for Young Scientists (No. 62302129), National Natural Science Foundation of China (Nos. 62272107), Hainan Province Key R&D plan project (No. ZDYF-2024GXJS030) and the HK RGC General Research Fund (Nos. 16208120 and 16214121).

Data Availability
Our manuscript has no associated data.

Declarations

Conflict of interest
The authors declare that they have no Conflict of interest.

Received: 23 September 2025 Accepted: 6 February 2026
Published online: 19 February 2026

References

Shamir A (1979) How to share a secret. Comm of the ACM 22(11):612–613
Cascudo I, David B (2024) Publicly verifiable secret sharing over class groups and applications to DKG and YOSO. In Eurocrypt'24. 216–248
Feldman PA (1987) practical scheme for non-interactive verifiable secret sharing. In FOCS'87, 427–438
Stadler M (1996) Publicly verifiable secret sharing. In Eurocrypt'96, 190–199
Cascudo I, David B (2017) SCRAPE: scalable randomness attested by public entities. In ACNS'17, 537–556
Beimel A, Tassa T, Weinreb E (2005) Characterizing ideal weighted threshold secret sharing. In TCC'05, 600–619
Belenkiy M (2008) Disjunctive multi-level secret sharing. Cryptology ePrint Archive
Tassa T (2007) Hierarchical Threshold Secret Sharing J Cryptol 20(2):237–264
Tassa T, Dyn N (2009) Multipartite secret sharing by bivariate interpolation. J Cryptol 22(2):227–258
Chen Q, Tang C, Lin Z (2021) Efficient explicit constructions of multipartite secret sharing schemes. IEEE TIT 68(1):601–631
Gentry C, Halevi S, Lyubashevsky V (2022) Practical non-interactive publicly verifiable secret sharing with thousands of parties. In Eurocrypt'22, 458–487
Syta E, Jovanovic P, Kogias EK, Gailly N, Gasser L, Khoffi I, Fischer MJ, Ford B (2017) Scalable bias-resistant distributed randomness. In SP'17, 444–460
Bessani AN, Alchieri EP, Correia M, Fraga JS (2008) DepSpace: a Byzantine fault-tolerant coordination service. In Eurosys'08, 163–176
Kiayias A, Russell A, David B, Oliynykov R (2017) Ouroboros: a provably secure proof-of-stake blockchain protocol. In Crypto'17, 357–388
Fujisaki E, Okamoto T (1998) A practical and provably secure scheme for publicly verifiable secret sharing and its applications. In Eurocrypt'98, 32–46
Fiat A, Shamir A (1986) How to prove yourself: practical solutions to identification and signature problems. In Eurocrypt'86, 186–194
Damgård I (2002) On Σ-protocols. University of Aarhus, Department for Computer Science, Lecture Notes
Schoenmakers B (1999) A Simple Publicly Verifiable Secret Sharing Scheme and Its Application to Electronic. In Crypto'99, 148–164
Ruiz A, Villar JL (2005) Publicly verifiable secret sharing from Paillier's cryptosystem. In SAC'05
Heidarvand S, Villar JL (2009) Public verifiability from pairings in secret sharing schemes. In SAC'09, 294–308
Jhanwar MP, Venkateswarlu A, Safavi-Naini R (2014) Paillier-based publicly verifiable (non-interactive) secret sharing. Des Codes Crypt 73:529–546
Bethencourt J, Sahai A, Waters B (2007) Ciphertext-Policy Attribute-Based Encryption. In SP'07, 321–334

Berrut J-P, Trefethen LN (2004) Barycentric lagrange interpolation. SIAM Rev 46(3):501–517
Cascudo I, David B, Garms L, Konring A (2022) YOLO YOSO: fast and simple encryption and secret sharing in the YOSO model. In Asiacrypt'21, 651–680
Cascudo I, David B (2020) ALBATROSS: publicly attestable batched randomness based on secret sharing. In Asiacrypt'20, 311–341
Akinyele JA, Garman C, Miers I, Pagano MW, Rushanan M, Green M, Rubin AD (2013) Charm: a framework for rapidly prototyping cryptosystems. J Cryptogr Eng 3:111–128
Lewko A, Waters B (2011) Decentralizing attribute-based encryption. In Eurocrypt'11, 568–588
Rouselakis Y, Waters B (2015) Efficient statically-secure large-universe multi-authority attribute-based encryption. In FC'15, 315–332
Zhang L, Kan H, Qiu F, Hao F (2024) A Publicly Verifiable Optimistic Fair Exchange Protocol Using Decentralized CP-ABE. Comput J 67(3):1017–1029
Avoine G, Vaudenay S (2004) Optimistic fair exchange based on publicly verifiable secret sharing. In ACISP'04
Zhang L, Wu X, Ma Y, Kan H (2025) Data exchange for the metaverse with accountable decentralized TTPs and incentive mechanisms. IEEE Trans Big Data, Accepted
Sethi K, Pradhan A, Bera P (2021) PMTER-ABE: a practical multi-authority CP-ABE with traceability, revocation and outsourcing decryption for secure access control in cloud systems. Clust Comput 24(2):1525–1550
Li J, Yan H, Koe ASV, Deng W, Zhong Z (2025) Secure and Revocable Multi-authority CP-ABE for Mobile Cloud Computing. In International conference on algorithms and architectures for parallel processing, pp. 75–84

Publisher's Note

Qiuling Yue Qiuling Yue holds a Ph.D. in Cryptography from Beijing University of Posts and Telecommunications, and has served as a visiting scholar at the University of Illinois at Urbana-Champaign and Southern Illinois University Carbondale. She is recognized as a high-level talent in Hainan Province. Her current research focuses on applied cryptography, privacy-preserving computation, blockchain, and quantum information. Recently, she has published several papers in SCI-indexed journals