Secure Power Data Sharing with Fine-grained Control: A Multi-strategy Access Tree Approach

Zhuo Yun JIANG [1], Jia Wei ZHANG [2], Hao Jie YANG [3], Peng GENG [3]*

[1] School of Mathematical Sciences, Soochow University, Suzhou, 215006, China
[2] School of Electrical and Electronic Engineering, Northumbria University, Newcastle NE1 8ST, United Kingdom
[3] School of Information and Communication Engineering, Nanjing Institute of Technology, Nanjing, 211167, China

gengpeng@njit.edu.cn

Submitted August 20, 2024 / Accepted October 12, 2024 / Online first November 12, 2024

Abstract. *The current data sharing schemes mainly employ Attribute-Based Encryption (ABE) technique to achieve one-to-many access control for power data. However, these schemes suffer from issues such as low encryption efficiency and vulnerability to user attribute tampering. To address these problems, a power data sharing scheme based on multi-strategy access trees is proposed. By combining ABE with symmetric encryption algorithms, specifically employing the Advanced Encryption Standard (AES) in conjunction with Ciphertext-Policy ABE (CP-ABE), a hybrid encryption mechanism is adopted. Building upon an encryption algorithm rooted in multi-strategy access trees, data visitors are categorized into security levels according to roles and regions. Then, a time-constrained attribute encryption scheme is proposed for designated personnel, thereby achieving confidentiality and fine-grained access control for power data. Analysis results indicate that the proposed scheme enables secure sharing of power data and is highly suitable for resource-limited power terminal devices.*

Keywords

Attribute-based encryption, access control, fine granularity, threshold trees, smart grid

1. Introduction

Electric power data, comprising production data, customer data, operational data, and financial data, mirrors the development trends of the economy and society as well as electricity usage behaviors of enterprises, serving as a crucial component in advancing the construction of the digital economy. As social and economic interconnections strengthen, the external sharing of power data emerges as a vital means to dismantle industry barriers, enhance the dynamism of socioeconomic operations, and reduce the operating costs across various economies [1]. Although data sharing technologies facilitate breaking down data silos and promoting data flow, the sensitivity of corporate data poses a paramount issue that requires careful attention throughout the process of external data sharing.

Currently, most electric power enterprises adopt cloud-based data sharing methods to facilitate the storage and analysis of massive data volumes. However, these approaches are accompanied by challenges concerning data confidentiality and user privacy safeguards. Attribute-Based Encryption (ABE) [2] presents a solution that integrates data privacy protection with access control, making it a widely applied technology in power data sharing scenarios. As a development of public-key cryptography and identity-based cryptography, ABE supports detailed access permissions and enforces a comprehensive encrypted access framework, enhancing the security and manageability of shared data in the power industry.

ABE is primarily categorized into two types depending on how access rules associate with attributes: Key-Policy ABE (KP-ABE) [3] and Ciphertext-Policy ABE (CP-ABE) [4]. For KP-ABE, the access control rule governing the encrypted data is embedded in the decryption key of the recipient. This implies that the user's decryption key defines an access pattern based on the set of attributes they possess, and decryption of a ciphertext can occur only when the set of attributes carried by the ciphertext aligns with the pattern specified in the decryption key. CP-ABE embeds the access control policy directly into the ciphertext. At encryption, a policy is specified, and only those users whose collection of attributes in their decryption key comply with this rule can decrypt the data. Both offer high levels of security and flexibility in data sharing, with the choice between them hinging on specific application requirements and contexts. KP-ABE is better suited for scenarios requiring centralized management of access permissions, such as internal company data distribution, where permissions can be predetermined and relatively static. Fadlullah et al. [5] employed KP-ABE to implement access permission rule for protected transmission in an intelligent grid context, where the command station broadcasts a unified secure information to targeted recipients. Members of the targeted recipient individually decrypt the message using predefined key policies. The drawbacks of this scheme include its inadequacy for granular access permission rule and the fact that it results in a significant amount of keys due to the command station generating keys for each user, lacking the capability for user

revocation. Huang et al. [6] proposed a searchable encryption method wherein data owners can ensure that their data is accessible for querying only by authorized users. However, this scheme requires all information to be transmitted over a secure channel, imposing high demands on practical application scenarios.

Unlike KP-ABE, CP-ABE is suitable for decentralized environments, such as object storage service, as data owners can specify access policies at the time of encryption without needing prior knowledge of the precise attributes of all potential recipients. This flexibility accommodates evolving user populations and dynamic access demands, making CP-ABE particularly apt for access control applications in smart grid scenarios, including cloud storage and fine-grained data sharing.

Zhang et al. [7] proposed a verifiable multi-entity access control system using CP-ABE for intelligent grid, supporting granular access permission rule for real-time shared information and restricting the control of malicious entities over private keys. Addressing the security issues in data sharing among different entities in smart grids, Sitharthan et al. [8] employed the Linear Secret Sharing Scheme (LSSS) proposed in [9] to transform access trees into LSSS matrices. Every line of the LSSS matrix corresponds to an attribute, effectively safeguarding the data privacy of different entities. However, this approach falls short in expressing complex access structures and incurs substantial computational overhead.

Regarding data security from smart meters, the authors in [10] first segmented the data collected at the frontend and then employed access tree encryption, effectively prevent unauthorized users from accessing the data maliciously. Addressing the vulnerability of conventional ABE schemes to attacks, researchers have proposed numerous approaches. For instance, Liu et al. [11] proposed a decentralized access control system with user revocation capabilities to maintain data confidentiality, and Zuo et al. [12] utilized blockchain and CP-ABE to construct a cloud data sharing scheme, wherein all data uploaded onto the blockchain is immutable. Regarding the security challenges of large datasets, Xiong et al. [13] proposed a searchable encryption scheme for a cloud storage environment, which provides a robust solution for overcoming the obstacles related to data security and retrieval efficiency for large datasets. Related work by Jiang et al. [14] introduced a peer-to-peer blockchain-based power trading system, though it encountered privacy leakage issues. To tackle this issue, Zhao et al. [15] put forward a transaction model leveraging CP-ABE algorithms that safeguard privacy within the Internet of Things (IoT). The authors in [16] employed a light-weight encryption algorithm for smart metering data authentication, thereby increasing the difficulty of large integer factorization; however, this approach suffers from slower computation speeds, rendering it unsuitable for handling massive data.

Additionally, in the field of cryptography, power analysis, as a type of side-channel attack, infers encryption keys or other sensitive information by monitoring the power consumption patterns of devices during encryption operations, as demonstrated by an extensive comparison of commonly used machine learning algorithms such as support vector machine, decision trees, and new approaches based on k-NN in [17].

From the above analysis, the granular control of data access is manifested in data owners' ability to formulate access policies based on the attributes of accessible users when publishing shared data, and subsequently encrypting the data according to these policies. These access policies, when determining user access rights, only rely on the attributes inherent to the users themselves, overlooking some essential external factors. In the context of smart grids, for example, not only are there various types of intelligent terminal devices but also users with different roles are required to perform time-constrained access operations on these devices. Therefore, this paper targets the complex environment of multi-role users and multiple devices in smart grids, proposing an authentication scheme that combines time-bound attribute-based encryption and dual-factor security to achieve the integration of authentication, authorization, and access control in intelligent grid, ensuring the safe storage and retrieval of power data.

The key contributions of this paper are:

(1) **Time-constrained fine-grained access control**. Time constraint is introduced for specific access personnel within the multi-policy access tree secret sharing access structure, achieving more fine-grained access control requirements. Data owners can autonomously determine the accessible users for shared data and their respective permission release times.

(2) **Access structure with time attributes.** For different access users, an access structure with time-sensitive access needs is designed, enabling efficient management of user permissions.

The organization of the remainder of this paper is: Section 2 describes the system model and security model, Section 3 gives detailed construction of the proposed scheme, and Section 4 evaluates the proposed scheme in terms of its security and performance. Finally, Section 5 provides a summary.

2. System Architecture and Security Model

2.1 System Architecture

The system architecture of the proposed scheme comprises four entities: a trusted Authorization Center (AC), a database, a Data Service Managers (DSM), and many Data Visitors (DVs). Figure 1 illustrates the relationships between these entities.

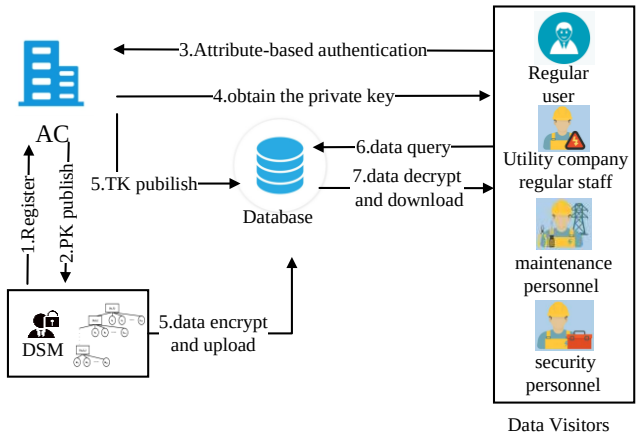

Fig. 1. System architecture.

As shown in Fig. 1, the entities that constitute the system architecture are outlined as follows:

(1) The AC is the control center of the system and the only trusted entity, managing all attributes. Its main tasks include system initialization, assigning attributes based on user information, and generating necessary system parameters. It also functions as the private key issuance center, generating attribute-specific private keys for users upon receiving their private key requests.

(2) The database stores the encrypted power data provided by the DSM and handles access requests for the encrypted power data from Data Visitors.

(3) The DSM defines access policies according to the types of user attributes available, embeds these policies into the power data, encrypts it, and uploads it to the database.

(4) The DVs (or users) have different roles such as regular users, grid maintenance personnel, and security personnel. Data Visitors apply to the authorization center for attribute private keys. They can access the encrypted power data only when their private keys, associated with their attributes, fulfill the access criteria constructed by the DSM during encryption. The DVs need to be classified according to their role, region, and time constraints, allowing them to access only the data that meets their specific access policy.

As shown in Fig. 1, DSM transmits ciphertexts to the database, where users can query any ciphertext. The AC maintains security by performing the following two functions:

(1) It distributes attribute-related private keys related to the attributes of users.

(2) At every point in time, it publishes time constraints, which are used to control the timely release of access permissions to users.

2.2 Security Model

The proposed scheme is designed to be semantically secure against both chosen-plaintext attacks and chosen-access structure attacks, and this section describes the security game process between an attacker A and a challenger C:

Initialization: The challenger C selects a security parameter ξ and excutes the *Setup* procedure. Then, C sends the obtained system public parameters *PK* to the attacker A while maintaining the confidentiality of the master secret key *MSK*. Additionally, the challenger C sets up an empty key set *SK*.

Stage 1: The attacker A chooses some authorization units *Atts*[] and issues private key queries $Q_i(i = 1,2,\ldots,m)$ to C. In response, C runs *KeyGen* procedure to create private keys $SK_{Atts[]}$, then stores $SK_{Atts[]}$ in set *SK* and delivers them to A.

Challenge: The attacker A selects two plaintexts m_0, m_1 of equal length and a challenge access tree Γ, and sends them to C, ensuring that the authorization units queried by attacker A in Stage 1 do not meet the access tree Γ. The challenger C randomly selects a bit $b \in \{0,1\}$, and encrypts m_b using Γ to produce a challenge ciphertext CT^*. The challenger C sends CT^* to A, but does not reveal the value of *b*.

Stage 2: Carry out the process of Stage 1 again, but this time the authorization units queried by the attacker A must not satisfy the challenge access structure Γ.

Guess: The attacker A provides a prediction $b' \in \{0,1\}$. If $b' = b$, then the attacker A succeeds in the experiment.

The success probability of the attacker in the experiment is specified as $\left| P_r[b' = b] - \frac{1}{2} \right|$.

2.3 Time-constrained Access Structure

In this paper, Γ is defined as a threshold access tree containing nodes (leaf nodes and non-leaf nodes) and carrying some time traps, denoted as *TS*, which represent the permission release times for relevant user groups. All leaf nodes are represented by attribute values or related to attributes, while non-leaf nodes are represented by threshold (*p*,*n*), where *n* indicates the quantity for child nodes under the current node, and *p* indicates the minimum count of conditions that need to be satisfied. Time attributes are attached to the required nodes to determine whether they are within the allowed access period, thereby implementing time constraints.

This threshold access tree is similar to traditional CP-ABE, but from an algorithmic perspective, the time trap *TS* can be attached to any node (leaf nodes, non-leaf nodes, or even the root node), binding the time at which corresponding users can start accessing the data. In Fig. 2, the time trap TS_1 is attached to a leaf node, restricting the permission release time for the single attribute Att_1, while TS_2 is associated with a non-leaf node, it helps define the sub-policy "$Att_2 \wedge Att_3$".

The time constraints correspond to *TS*, and they are uniformly published by the AC at specific times, which change the state of the corresponding trapdoors. The state of the trapdoor determines whether the user can retrieve the data. In the proposed scheme, the trapdoor conditions are categorized as "released" and "non-released". If a trapdoor is in a non-released state, the corresponding user cannot obtain the associated secret through that trapdoor. A released

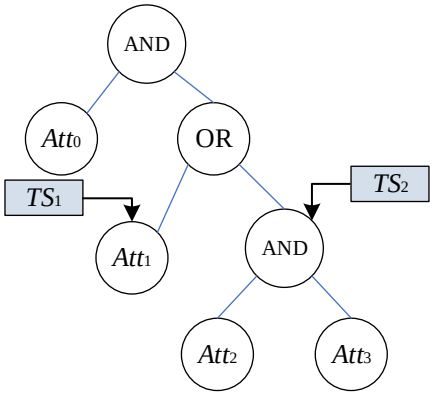

Fig. 2. A time-constrained access scenario.

trapdoor allows users to obtain secret information bottom-up through the trapdoor, and released time traps are denoted as *TS′*.

2.4 Formal Definition

The formal definitions of the four polynomial-time algorithms are described as:

(1) **System Setup Algorithm.** The AC establishes the foundational parameters required for the system. It takes a bilinear group file, denoted as *PairingFile*, and a security parameter ξ as inputs, and produces a public key *PK* and a master secret key *MSK*. The *PK* is utilized for encrypting data and defining access policies, while *MSK* is used internally for generating private keys for individual users. Once generated, *PK* and *MSK* are made available to the system. Its formal definition is

$$Setup\left(PairingFile,\xi\right)\rightarrow\left(PK,MSK\right). \tag{1}$$

(2) **Key Generation Algorithm.** Data visitors provide their attribute sets to the AC for authentication. The AC invokes *KenGen* algorithm to produce a *SK* aligned with the user's attributes. This algorithm takes the *MSK*, *PK* and the visitor's attributes *Atts*[] as inputs, and outputs *SK* related to the visitor's attributes. Its formal definition is

$$KeyGen(PK,MSK,Atts[])\rightarrow SK. \tag{2}$$

(3) **Plaintext Encryption Algorithm.** The DSM first performs the AES symmetric encryption on the plaintext to be uploaded, then encrypts the symmetric key using the public key. This algorithm takes *PK*, the plaintext *M*, and a custom access policy Γ as inputs, and outputs the ciphertext *CT*. Its formal definition is

$$KemEncrypt\left(PK,M,\Gamma\right)\rightarrow CT. \tag{3}$$

(4) **Ciphertext Decryption Algorithm.** The DV (or user) executes this algorithm to gain access to the ciphertext. This algorithm considers the data visitor's attribute set associated *SK* and the *CT* as inputs. If the user's attribute satisfies the access policy defined by the DSM and the attribute time falls within the allowed time constraints, then the plaintext *M* is output; otherwise, decryption fails and an error message is output. Its formal definition is

$$Decrypt\left(SK,CT\right)\rightarrow M\ \text{or}\ \perp. \tag{4}$$

where ⊥ denotes failure or an error message when decryption is not possible due to an unsatisfied access policy or invalid time constraints.

3. Implementation

This section provides a detailed description of the specific implementation for secure and efficient access to power data, combining the efficiency of AES encryption and decryption with the security of CP-ABE key management. This ensures the safety of power data and keys during network transmission and server storage.

3.1 Setup

The setup process is carried out by AC. AC selects two bilinear groups G_1 and G_T of the same prime order *p*, as well as a generating element *g* of G_1, hash functions $H_1(\cdot):\{0,1\}^*\rightarrow G_1^*$, and $H_2(\cdot):G_T^*\rightarrow Z_p^*$. AC generates the element $A = [p, G_1, G_T, g, e, H_1, H_2, F_T]$, where F_T represents the uniform format for system time.

Specifically, the AC generates the pairing-related public parameters $\langle e, g, G_1, G_T, Z_r\rangle$ based on the bilinear group parameters *PairingFile* and a security parameter ξ. It then randomly selects $\alpha, \beta, \gamma \in Z_r$, and generates the system public key as follows:

$$PK=\left(A,h=g^{\beta},f=g^{\gamma},e\left(g,g\right)^{\alpha}\right). \tag{5}$$

The system master secret key *MSK* is defined as $(\beta, \gamma, g^{\alpha})$, where *f* and γ are used for time release.

3.2 Key Generation

Key generation algorithm

$KeyGen(PK, MSK, Atts[]) \rightarrow SK$. The AC maps each attribute in *Atts*[] to a corresponding group element in the group G_1, and performs a hash operation $H(i)$. It then selects a random number $\omega \in Z_r$ for generating a private key for the DV that is related to attributes as follows:

$$SK=\left\{g^{\alpha}g^{\beta\omega},g^{\omega},\left\{H\left(i\right)^{\omega}\right\}_{i\in Atts[]}\right\}. \tag{6}$$

3.3 Encryption

Ciphertext encryption algorithm

$KemEncrypt(PK, M, \Gamma) \rightarrow CT$. This algorithm includes two processes:

(1) Firstly, the DSM encrypts the plaintext *M* using AES. It selects a random number *s* from the cyclic group Z_r to be used as the root node secret value and an element *t* from the group G_T. Using these values, it computes the ciphertext components $\{C = te(g, g)^{\alpha s}, C'=g^s\}$. The element *t* from G_T

is hashed using the function $H(t)$ to derive a symmetric key. The original plaintext M and the derived symmetric key are then encrypted using AES, resulting in the symmetric ciphertext *symmetric cp*, which is then stored.

$$E\left(M, symmetric\ key\right) \rightarrow symmetric\ cp. \quad (7)$$

(2) Secondly, perform public-key encryption on the symmetric key. An access tree Γ is defined based on the accessible user attribute set U. For each node i, an index value index(i) is defined in breadth-first traversal order. The threshold value of non-leaf nodes is denoted as (k,n). This function is responsible for generating shares for each node in the access tree Γ based on Shamir's Secret Sharing scheme. For each non-leaf node i, a random set of polynomial parameters $\{s_n\}$ is selected over the integer group Z_r. A polynomial $P_i(x) = s + s_1x + s_2x^2 + \ldots + s_{k-1}x^{k-1}$ is generated, with the constant term set to the secret value to be shared, denoted as $P_i(0) = s$. Secret sharing is performed along the access tree Γ. If a node is a threshold node, its threshold index value x = index(i) is used as the x-coordinate, substituted into the parent node's polynomial to obtain a result that serves as the leading coefficient for constructing a polynomial $P_i(x)$ of degree $k-1$. Setting the polynomial equal to 0 yields the corresponding threshold node secret share λ_i. The recursive function *nodeshare*() is then called again to continue sharing the node secrets. Specifically, for each child node j of node i: (1) If node j is a threshold node, repeat the polynomial generation and substitution process described above; (2) If node j is a leaf node, its index(j) value is used as the x-coordinate and substituted into the parent node's polynomial to calculate the leaf node secret share λ_j, and the secret sharing stops at the leaf node. By recursively applying this process, the secret shares are distributed to all nodes in the access tree Γ, ensuring that only authorized users can reconstruct the original secret.

Perform a hash operation $H(U_i)$ on each leaf node attribute $U_{i\in \text{leaf noodes}}$, and randomly select $r_i \in Z_r$ for each attribute, then construct ciphertext components as:

$$\left\{C_{i_1} = g^{\beta\lambda_i} H\left(U_i\right)^{-r_i}, C_{i_2} = g^{r_i}\right\}. \quad (8)$$

At this point, the ciphertext is generated as follows:

$$CT = \left\{C, C', \left\{C_{i_1}, C_{i_2}\right\}_{i_1, i_2 \in U_i}\right\}. \quad (9)$$

For each trapdoor TS_i, the associated permission release time $tt \in F_T$, and the node secret parameter is q_i^{τ}. The DSM chooses a random value r_{tt}, and the trapdoor generation process is as follows:

$$TS_i = \left(A_i = g^{r_i}, B_i = q_i^{\tau} + H_2\left(e\left(H_1\left(t\right), f\right)^{r_{tt}}\right)\right). \quad (10)$$

3.4 Time Token Generation and Trapdoor Release

At each time point $tt \in F_T$, the time constraints generated and published by the AC are:

$$TC_{tt} = H_1\left(tt\right)^{\gamma}. \quad (11)$$

When a specific time point tt corresponding to a trapdoor TS_i is reached, the database retrieves the relevant time constraint TC_{tt} from the content published by the AC. Then, it queries all access structures in the data for time-bound trapdoors associated with tt. For each trapdoor, the following computation is performed to transform the trapdoor state:

$$TS_i^{'} = B_i - H_2\left(e\left(TC_t, A_i\right)\right). \quad (12)$$

After this process is correctly executed, the database stores TS_i' instead of TS_i in the relevant ciphertexts.

3.5 Decryption

Decryption algorithm *Decrypt* (SK, CT) $\rightarrow M$. If the attribution set S of the DV overlaps with the set of leaf node attributes U in the access tree Γ at attribute i, then the pairing result of the ciphertext component and key component corresponding to i is treated as a secret share as

$$\begin{aligned} &P_i = e\left(C_{i_1}, g^{\omega}\right) e\left(C_{i_2}, H\left(i\right)^{\omega}\right) = \\ &e\left(g^{\beta\lambda_i} H\left(i\right)^{-\lambda_i}, g^{\omega}\right) e\left(g^{r_i}, H\left(i\right)^{\omega}\right) = e\left(g,g\right)^{\beta\omega\lambda_i}. \end{aligned} \quad (13)$$

Using the recovery of the leaf node secret shard P_i as an example, define node z as the parent of node i, then call *noderecovery*() function to perform Lagrange interpolation for non-leaf nodes, resulting in:

$$\begin{aligned} &P\left(z\right) = \prod_{i \in S_z} P_i^{\Delta x', s_z^{'}(0)} = \prod_{i \in S_z} \left(e\left(g,g\right)^{\beta\omega P_i(0)}\right)^{\Delta x', s_z^{'}(0)} = \\ &= \prod_{i \in S_z} \left(e\left(g,g\right)^{\beta\omega P_{\text{parent}(i)} index(i)}\right)^{\Delta x', s_z^{'}(0)} = e\left(g,g\right)^{\beta\omega P_z(0)} \end{aligned} \quad (14)$$

where $x' = index(i)$, and $S_z' = \{index(i) : i \in S_z\}$. For the root node, the secret value is ultimately recovered in the form of $e(g,g)^{\beta\omega s}$. Then, the randomly generated element t in the group G_T is recovered according to:

$$\frac{C}{\left(\frac{e\left(C', g^{\alpha} g^{\beta\omega}\right)}{e\left(g,g\right)^{\beta\omega s}}\right)} = \frac{\left(te\left(g,g\right)^{\alpha s}\right)}{\left(\frac{e\left(g^{s}, g^{\alpha} g^{\beta\omega}\right)}{e\left(g,g\right)^{\beta\omega s}}\right)} = \frac{\left(te\left(g,g\right)^{\alpha s}\right)}{e\left(g,g\right)^{\alpha s}} = t. \quad (15)$$

Next, determine whether the attribute is within the valid time constraint. Let the start and end times of the time constraint be T_0 and T_1, respectively. If the current time $T \in (T_0, T_1)$, then perform a hash operation $H(t)$ on the recovered element t to obtain a symmetric key, which subsequently decrypts the symmetrically encrypted ciphertext *symmetric cp* using AES, yielding the plaintext:

$$E\left(symmetric\ cp, symmetric\ key\right) \rightarrow M. \quad (16)$$

Otherwise, if the current time $T \notin (T_0, T_1)$, it is not possible to derive the symmetric key for AES decryption, which means the decryption will fail.

4. Evaluation

The hardware/software used in our experiments is: the software environment utilizes Java, while the hardware platform consists of an AMD Ryzen 5 5600H CPU with 16 GB of RAM. The AC selects a security parameter ξ to define the cryptographic strength of the system. Typically, ξ is set to 128 bits to achieve a recommended level of security strength. Additionally, the AC selects two bilinear groups G_1 and G_2 of the same prime order p, along with a generating element g of G_1, hash functions H_1 and H_2, and generates the element T, where T represents the uniform format for system time. The link to the relevant source code is: https://github.com/ZiYi-Wang-git/ABE.

4.1 Security Evaluation

A security proof is offered for the attribute-based encryption approach proposed in this work. Firstly, it is demonstrated that the proposed attribute-based encryption system is secure under the general group model, and subsequently, the scheme's resilience against collusion attacks is examined.

Theorem 4.1: The scheme proposed in this paper is provably secure under the general group model.

Proof: In the model defined in Sec. 2.2, for an attacker A to win the game, they need to:

(1) Distinguish between $m_0 e(g, g)^{\alpha s}$ and $e(g, g)^t$;

(2) Distinguish between $m_1 e(g, g)^{\alpha s}$ and $e(g, g)^t$, where t is an element randomly chosen from Z_r. At this point, the advantage of the attacker A is at least $\varepsilon/2$.

Let γ_0 and γ_1 represent two random encodings over Z_r. Through oracle requests, operations can be performed over the groups G_1 and G_T, and bilinear map operations e: $G_1 \times G_1 \rightarrow G_T$ can be accessed. Let G_1 be a general bilinear group. Let $\gamma_0(1)$ denote g, $\gamma_0(\sigma)$ denote g^1, and $\gamma_0(x)$ denote g^x. Similarly, let $\gamma_1(1)$ denote $e(g, g)$ and $\gamma_1(y)$ denote $e(g, g)^y$. In the following simulation, the simulator B acts as the challenger C for the entire process.

Initialization: The simulator B chooses a security parameter ξ and accesses oracles to compute the system public parameters. Then, B sends the obtained PK to the attacker A. Additionally, B establishes an empty key set SK.

The attacker A selects some authorization units and issues private key queries $Q_i (i = 1,2,\ldots,m)$ to B. B runs *KeyGen* algorithm to create private key components for the authorization units; the corresponding private keys are denoted by $SK_{(Atts[])}$; the challenger stores the private keys $SK_{(Atts[])}$ in the set SK and sends them to A.

Challenge: The attacker A chooses two plaintexts m_0 and m_1 of equal length, along with an access tree Γ and presents them to the simulator B, where the authorization units queried by the attacker A in the first stage do not meet Γ. B randomly selects a bit $b \in \{0,1\}$ and then encrypts m_b with Γ, generating ciphertext components $C = e(g,g)^t$, $C' = g^s$.

The simulator B sends these to the attacker A. The ciphertext can only effectively decrypted if the keys for the authorization units that satisfy the access structure Γ are obtained by the attacker A. But this cannot happen in this game. Since A cannot obtain the leaf node secret shares $e(g, g)^{\beta\omega\lambda_i}$ they cannot recover the element t from the root node secret value $e(g, g)^{\beta\omega s}$. Therefore, the proposed approach is shown to be secure.

Theorem 4.2: The proposed scheme can resist collusion attacks.

Proof: In the proposed method, when AC generates attributes for each user, it randomly selects $s \in Z_r$. In this case, since each user's random factor s is different, other users cannot obtain the correct decryption key. For each authorization unit, s is randomly split into s_i and s_t, which are respectively bound to the identity attributes and time attributes. This effectively prevents users from attempting to combine their own identity and time attributes within their authorization units to gain unauthorized access.

4.2 Performance Evaluation

A performance analysis of the proposed scheme is conducted in terms of access control, confidentiality, resistance to collusion attacks, and time overhead.

(1) **Fine-grained Access Control.** Data owners can assign different access permissions according to the attributes of accessors. Only when the attributes of the data accessor meet the access rule can they access the power data. Additionally, data owners can impose time interval restrictions on any attribute. In specific scenarios, they can change the access policy and time limits, preventing data leakage to unauthorized users or malicious accessors.

(2) **Confidentiality.** If a malicious accessor's attribute set does not match the leaf node attribute sets of ciphertext access structure, they cannot obtain the leaf node secret shares $e(g, g)^{\beta\omega\lambda_i}$. Consequently, they cannot recover the element t using the root node secret value $e(g, g)^{\beta\omega s}$ to derive the symmetric key. Therefore, malicious accessors cannot obtain the symmetric key by cracking the AES algorithm, ensuring power data's security.

(3) **Resistance to Collusion Attacks.** If malicious accessors with partial decryption keys attempt to act together and merge their decryption keys to gain access, their efforts will be unsuccessful. During the private key generation phase, different visitors generate distinct private key components $H(i)^{\omega}$. Malicious accessors can only compute the secret values $e(g, g)^{\beta\omega P_z(0)}$ corresponding to non-leaf nodes, but they cannot decrypt the plaintext M by piecing together these values. Thus, the independence between user keys is maintained, making the scheme resistant to collusion attacks.

(4) **Time Overhead**. Based on the given personnel attributes in the aforementioned encryption and decryption simulations, the effectiveness of ABE algorithm is evaluated. The results of the CP-ABE based threshold tree algorithm [18] with the proposed hybrid encryption algorithm are compared, as shown in Fig. 3 and 4, respectively.

Entities	Proposed scheme	Scheme in [20]
AC	$(3+N_C)\|g\|$	$(5+N_C)\|g\|$
DMS	$(4+2N_D)\|g\|$	$(2+2N_D)\|g\|$
User	$(2+3N_U)\|g\|$	$(2+3N_U)\|g\|$

Tab. 1. Storage overhead.

Interaction	Proposed scheme	Scheme in [20]
User registration	$(2+3N_U)\|g\|$	$(1+2N_U)\|g\|$
Data update	$(4+2N_D)\|g\|$	$(4+4N_D)\|g\|$
Data access	$(2+3N_U)\|g\|$	$(1+2N_U)\|g\|$

Tab. 2. Transmission overhead.

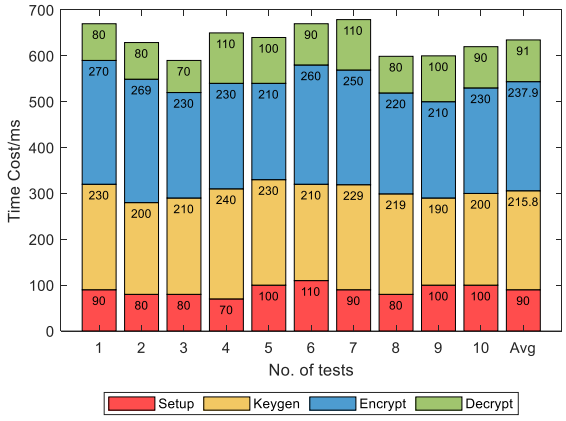

Fig. 3. Time overhead of the threshold tree algorithm.

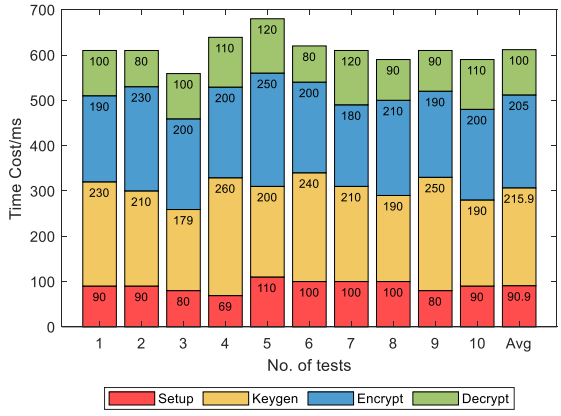

Fig. 4. Time overhead of the proposed multi-strategy threshold tree algorithm.

From Figs. 3 and 4, the time overhead for generating public settings is determined solely by the bit length of the elliptic curve [19], thus the time overhead for both algorithms is largely the same. During the key generation phase, both algorithms have an equal number of attributes, so the time overhead is also basically identical. During the encryption phase, due to the introduction of the AES algorithm in this paper, it only requires generating a symmetric key from a randomly generated element t in the G_T group and creating a symmetric ciphertext from the plaintext M, therefore, the encryption time overhead of the proposed algorithm is notably less than that of threshold tree algorithm. In the decryption phase, to obtain the element t, one needs to compute the ciphertext components of each node and then decrypt them using AES to reconstruct M. Thus, the decryption time of our algorithm is slightly greater than that of the threshold tree method. From the overall comparison of time overhead, it can be concluded that our algorithm has a relatively balanced time overhead, with an average value lower than that of the threshold tree algorithm. Moreover, our algorithm combines high efficiency and security in encryption and decryption, with a flexible structure, making it more suitable for smart grid scenarios.

The proposed scheme is compared with the scheme in [20] in terms of storage overhead and transmission overhead, and the results are shown in Tab. 1 and Tab. 2, respectively. In these tables, $|g|$ represents the size of elements in G and G_T, N_C is the total number of attributes in the proposed scheme, N_D represents the number of attributes held by the DSM, and N_U represents the number of attributes held by the user. The storage overhead is shown in Tab. 1. It can be seen from Tab. 1, compared to the scheme in [20], the proposed scheme has lower storage overhead for the AC, the same storage overhead for users, and slightly higher storage overhead for the DMS. Overall, the storage overhead between the two schemes is not significantly different. The transmission overhead is shown in Tab. 2. Due to the addition of time attributes, it can be observed that in the user registration phase, the proposed scheme incurs slightly higher transmission overhead compared to the scheme in [20]. In the data update phase, the proposed scheme has lower overhead than the scheme in [20]. In the data access phase, users in the proposed scheme need to send the ciphertext to the database, whereas in [20], users hold the keys. Therefore, the overhead in the data access phase is higher for the proposed scheme.

Furthermore, the proposed scheme is compared with the solution presented in [20] and CP-ABE based solutions. Figures 5 and 6 depict the time overhead for AC in releasing functionality on schedule as the number of users and the number of files awaiting authorization increase, respectively. The proposed scheme, due to the time token TK_t being a globally uniform parameter for all users, allows AC to

compute and publish a single token at each time point. If time is handled as an attribute, such as in CP-ABE schemes, AC would need to distribute time-related keys to each user at every instance, leading to linearly increasing time overhead with the number of users, making it less suitable for large-scale systems. In the scheme presented in [20], although AC incurs no additional overhead for scheduling functionality releases, the decryption task's overhead grows linearly with the scale of associated data, as shown in Fig. 6. The proposed scheme exhibits lower overhead and can meet the needs of large-scale systems.

(5) **Overhead for the data owner.** When the data owner uploads their shared file, the communication overhead depends on the size of the corresponding ciphertext package. If we only consider the impact of the number of intended access users, in the scheme presented in [20], the overhead for the data owner is $O(|N|)$, where N is the number of intended access users. In contrast, both the proposed scheme and CP-ABE-based solutions have an overhead of $O(|N_{attr}|)$, where N_{attr} is the number of attributes in the access policy. Typically, the growth rate of N_{attr} is much slower than that of N. Figure 7 illustrates the relationship between the overhead incurred by the data owner when encrypting a shared file and the number of intended access users. It can be seen from Fig. 6 that both the proposed scheme and the scheme in [20] significantly reduce the burden on the data owner.

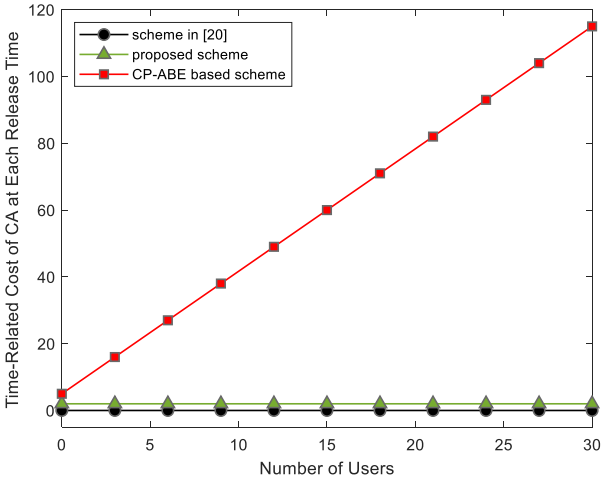

Fig. 5. Cost of AC versus number of users.

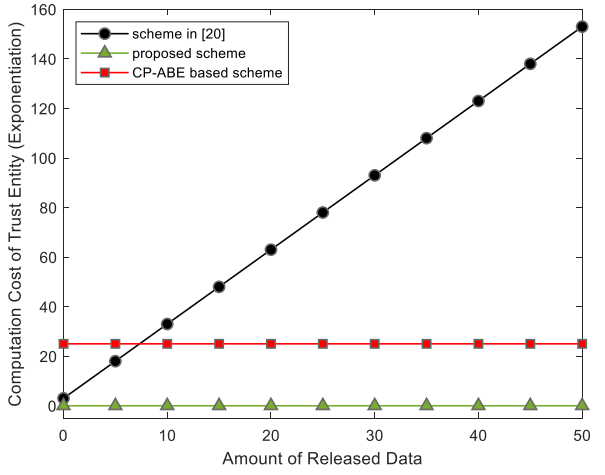

Fig. 6. Computation overhead versus amount of released data.

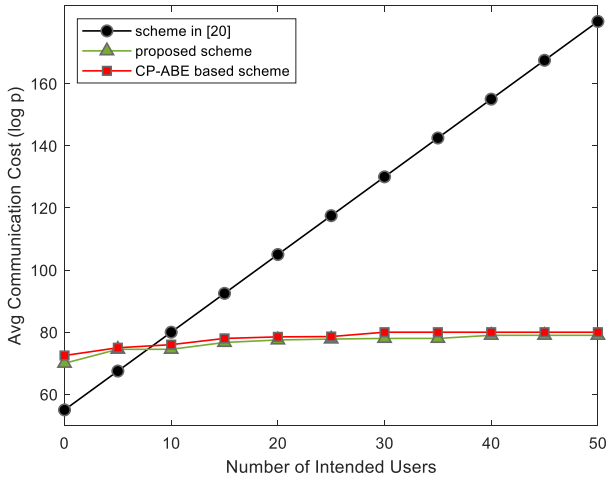

Fig. 7. Cost of owners versus number of intended users.

5. Conclusions

To address confidentiality concerns in power data, this paper presents a new approach enhancing data protection through a time-sensitive access control scheme. The proposed solution integrates a hybrid encryption mechanism that combines CP-ABE and AES. The core innovation lies in introducing temporal dimensions to the attribute-based encryption process, which is traditionally based on secret sharing mechanisms. By incorporating time attributes, the system allows for dynamic adjustments to access policies and time constraints according to specific application scenarios. This flexibility significantly enhances the richness of user permission management. The hybrid encryption mechanism ensures that power data remains securely stored by leveraging the strengths of both CP-ABE and AES. CP-ABE enables access control with fine granularity, allowing for policy-based decryption of ciphertexts by users who possess the required attributes. Meanwhile, AES provides robust symmetric key encryption for securing data at rest and in transit.

This combination not only guarantees the confidentiality and integrity of power data but also supports efficient and scalable access control. It meets the stringent requirements for managing large volumes of data while ensuring approved access within specified time frames. Thus, the proposed scheme achieves a harmonious blend of security and usability, rendering it appropriate for various scenarios in the smart grid.

Future work will focus on enhancing security mechanisms against advanced attacks and optimizing performance for resource-constrained environments. Additionally, we plan to develop practical implementations and deploy the scheme in real-world scenarios to ensure its effectiveness and usability.

Acknowledgments

This research was financially supported by the National Natural Science Foundation of China (61901211), Jiangsu Science and Technology Think Tank Program Project (JSKX24085), Jiangsu Key Research and Development Program (Social Development) (BE2022789), and the Science and Technology Special Fund (Social Development R&D Plan) of Jiangsu Province (SBE2022740885).

References

[1] SAIDI, A., NOUALI, O., AMIRA, A. SHARE-ABE: An efficient and secure data sharing framework based on ciphertext-policy attribute-based encryption and Fog computing. *Cluster Computing*, 2022, vol. 25, p. 167–185. DOI: 10.1007/s10586-021-03382-5

[2] ANCY, P. R., KRISHNA, A. V. N., BALACHANDRAN, K. B. M. An efficient access policy with multi-linear secret-sharing scheme in ciphertext-policy attribute-based encryption. *Journal of Theoretical and Applied Information Technology*, 2022, vol. 100, no. 5, p. 1404–1411. ISSN: 1992-8645

[3] XIE, S., ZHANG, L., WU, Q., et al. Flexibly expressive and revocable multi-authority KP-ABE scheme from RLWE for Internet of Medical Things. *Journal of Systems Architecture*, 2024, vol. 152, p. 1–12. DOI:10.1016/j.sysarc.2024.103179

[4] MISHRA, B., JENA, D., PATNAIK, S. Fine-grained access control of files stored in cloud storage with traceable and revocable multi-authority CP-ABE scheme. *International Journal of Grid and Utility Computing*, 2023, vol. 14, no. 4, p. 320–338. DOI: 10.1504/IJGUC.2023.10057953

[5] FADLULLAH, Z. M., KATO, N., LU, R., et al. Toward secure targeted broadcast in smart grid. *IEEE Communications Magazine*, 2012, vol. 50, no. 5, p. 150–156. DOI: 10.1109/MCOM.2012.6194396

[6] HUANG, Q., YAN, G., WEI, Q. Attribute-based expressive and ranked keyword search over encrypted documents in cloud computing. *IEEE Transactions on Services Computing*, 2023, vol. 16, no. 2, p. 957–968. DOI: 10.1109/TSC.2022.3149761

[7] ZHANG, L. Y., YANG, G., SONG, C., et al. Accountable multi-authority attribute-based data access control in smart grids. *Journal of King Saud University - Computer and Information Sciences*, 2023, vol. 35, no. 7, p. 1–12. DOI: 10.1016/j.jksuci.2023.101597

[8] SITHARTHAN, R., VIMAL, S., VERMA, A., et al. Smart microgrid with the Internet of Things for adequate energy management and analysis. *Computers & Electrical Engineering*, 2023, vol. 106, p. 1–11. DOI: 10.1016/j.compeleceng.2022.108556

[9] BEIMEL, A., FARRAS, O., MINTZ, Y., et al. Linear secret-sharing schemes for forbidden graph access structures. *IEEE Transactions on Information Theory*, 2022, vol. 68, no. 3, p. 2083–2100. DOI: 10.1109/TIT.2021.3132917

[10] BHAVANI, N. G., KUMAR, R., BHAWANI, S. P., et al. Design and implementation of IoT integrated monitoring and control system of renewable energy in smart grid for sustainable computing network. *Sustainable Computing: Informatics and Systems*, 2022, vol. 35, p. 1–10. DOI: 10.1016/j.suscom.2022.100769

[11] LIU, K. P., WANG, C. F., ZHOU, X. T. Decentralizing access control system for data sharing in smart grid. *High-Confidence Computing*, 2023, vol. 3, no. 2, p. 1–8. DOI: 10.1016/j.hcc.2023.100113

[12] ZUO, Y. T., KANG, Z. Z., XU, J., et al. BCAS: A blockchain-based ciphertext-policy attribute-based encryption scheme for cloud data security sharing. *International Journal of Distributed Sensor Networks*, 2021, vol. 17, no. 3, p. 1–16. DOI: 10.1177/1550147721999616

[13] XIONG, Y., LUO, M. X. Searchable encryption scheme for large data sets in cloud storage environment. *Radioengineering*, 2024, vol. 33, no. 2, p. 223–235. DOI: 10.13164/re.2024.0223

[14] JIANG, X., SUN, A., SUN, Y., et al. A trust-based hierarchical consensus mechanism for consortium blockchain in smart grid. *Tsinghua Science and Technology*, 2023, vol. 28, no. 1, p. 69–81. DOI: 10.26599/TST.2021.9010074

[15] ZHAO, M., DING, Y., TANG, S., et al. A blockchain-based framework for privacy-preserving and verifiable billing in smart grid. *Peer-to-Peer Networking and Applications*, 2023, vol. 16, p. 142–155. DOI: 10.1007/s12083-022-01379-4

[16] KAMAL, N. F., AL-ALI, A. K., AL-ALI, A., et al. LPPDA: A light-weight privacy-preserving data aggregation protocol for smart grids. *IEEE Access*, 2023, vol. 11, p. 95358–95367. DOI: 10.1109/ACCESS.2023.3311140

[17] MARTINASEK, Z., ZEMAN, V., MALINA, L., et al. k-nearest neighbors algorithm in profiling power analysis attacks. *Radioengineering*, 2016, vol. 25, no. 2, p. 365–382. DOI: 10.13164/RE.2016.0365

[18] PREMKAMAL, P. K., PASUPULETI, S. K., ALPHONSE, P. J. A. Efficient escrow-free CP-ABE with constant size ciphertext and secret key for big data storage in cloud. *International Journal of Cloud Applications and Computing (IJCAC)*, 2020, vol. 10, no. 1, p. 28–45. DOI: 10.4018/IJCAC.2020010103

[19] NANJO, Y., SHIRASE, M., KODERA, Y., et al. Efficient final exponentiation for cyclotomic families of pairing-friendly elliptic curves with any prime embedding degrees. *International Journal of Networking and Computing*, 2022, vol. 12, no. 2, p. 317–338. DOI: 10.15803/ijnc.12.2_317

[20] WANG, B. W., LI, W., XIONG, N. N. Time-based access control for multi-attribute data in Internet of Things. *Mobile Networks and Applications*, 2021, vol. 26, p. 797–807. DOI: 10.1007/s11036-019-01327-2

About the Authors…

Zhuo Yun JIANG is an undergraduate majoring in Financial Mathematics in Soochow University, with a primary research focus on numerical analysis and data processing.

Jia Wei ZHANG is an undergraduate majoring in Electrical and Electronic Engineering in Northumbria University, with a primary research focus on power data analysis.

Hao Jie YANG is an undergraduate majoring in Electronic Information Engineering in Nanjing Institute of Technology, with a primary research focus on intelligence algorithms.

Peng GENG (corresponding author) is an Associate Professor of Nanjing Institute of Technology. He received his Master's degree in Communication and Information System in 2007 from Guilin University of Electronic Science and Technology, Guilin, Guangxi, China. His research interests include complex systems and deep learning.