Distributed Attribute-Based Signature With Attribute Dynamic Update for Smart Grid

Qianqian Su, Rui Zhang, Rui Xue, You Sun, and Sheng Gao

***Abstract*—Smart grid is gaining more and more attention as one of the typical applications of Internet of Things. However, in a distributed environment, how to guarantee the privacy of users in electricity trading while ensuring the efficiency of the transactions is one of the urgent issues to be solved. In this article, we propose a distributed attribute-based signature (DABS) scheme for distributed electricity trading, which can support users' free choice of trade objects without revealing their real identities. We construct a signature generation and verification method by taking advantage of the open and hard-to-tamper properties of blockchain to achieve signature verifiability independent of dynamic changes in attributes.To improve the update efficiency, we propose an improved scheme that enables the update complexity to be reduced from $O(n)$ to $O(\log n)$, where n is the number of users. Finally, performance analysis and simulation experiments demonstrate the security and practicality of the DABS.**

***Index Terms*—Attribute-based signature, blockchain, electricity trading, multiple attribute authorities (AAs), smart grid.**

I. Introduction

With the rapid development of Internet of Things (IoT) technology and the popularity of IoT devices [1], [2], an era of the interconnection of everything has arrived. A prominent example of the IoT is the smart grid [3], which can be seen as the next generation of the electricity grid [4]. Smart grids consist of multiple smart components that sense measurements and send this data over the network, thus enabling real-time feedback on energy consumption. The widespread use of smart grid technologies is due to the popularity of the use of heterogeneous energy sources such as renewable and nonrenewable resources and the need for simultaneous energy production and consumption. Smart grid has new features such as dynamic and transparent electricity prices, free electricity trading between energy producers and energy consumers [5], etc. Specifically, households, communities, or factories cannot only meet their own electricity needs by generating electricity by themselves, but also make profits by selling excess electricity to nearby users at a suitable price through the smart grid [6], [7].

Such a distributed electricity trading scenario requires a trading platform that can support it, which can realize free and efficient distributed trading while supporting the dynamics of users, the traceability of trading, and the anonymity of users. Thanks to the development of blockchain technology [8], [9], such distributed electricity trading can be enabled by blockchain. There are already many researchers combining blockchain with smart grids to realize distributed electricity trading and management. In [10], the author used the consortium blockchain to build a management system for on-demand resource transactions on the Internet of Vehicles. Balance the transaction process between buyers and sellers through the Stackelberg game. In [11], the author proposed a electricity trading platform that can balance the efficiency of servers and users based on the hierarchical Stackelberg game. However, these electricity trading platforms do not perform well in dealing with an on-demand selection of trading partners [12], [13]. Most of these schemes require centralized agents and have not investigated how to select trading parties based on various attributes.

While, in the aforementioned distributed electricity trading scenario, due to the particularity of electricity transactions, in addition to ensuring the anonymity of users, both parties to the transaction also need to know some of the other party's information, such as distance, reputation, and the amount of electricity that needs to be purchased, to prevent huge losses caused by long-distance transmission and malicious destruction of the electricity market. One of the challenges faced when applying blockchain to solve the aforementioned problems is that blockchain cannot provide users with sufficient and comprehensive privacy protection. This is due to the fact that the

Manuscript received 7 July 2022; revised 7 October 2022; accepted 7 December 2022. Date of publication 12 December 2022; date of current version 24 July 2023. This work was supported from the Beijing Natural Science Foundation under Grant M21036; in part by Natural Science Foundation of Shandong Province under Grant ZR2022QF102; in part by National Natural Science Foundation of China under Grant 62072487; and in part by Zhejiang Provincial Natural Science Foundation of China under Grant LD22F020002. Paper no. TII-22-2991. *(Corresponding authors: Rui Zhang; Sheng Gao.)*

Qianqian Su is with the School of Computer Science and Technology, Qingdao University, Qingdao 266071, China (e-mail: qdusuqianqian@163.com).

Rui Zhang and Rui Xue are with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China, and also with the School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: zhangrui@iie.ac.cn; xuerui@iie.ac.cn).

You Sun is with the Information Science Academy, China Electronics Technology Group Corporation, Beijing 10008, China (e-mail: sunyou@iie.ac.cn).

Sheng Gao is with the School of Information, Central University of Finance and Economics, Beijing 10008, China (e-mail: sgao@cufe.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3228688.

Digital Object Identifier 10.1109/TII.2022.3228688

relevant information in the blockchain is public for all nodes, and the sharing of redundant ledgers by multiple nodes brings great privacy protection risks while improving the system reliability and avoiding single point of failure. Obviously, the pseudonym and elliptic curve digital signature algorithm (ECDSA) adopted by Bitcoin cannot meet the needs of distributed electricity trading. It needs a signature algorithm that can protect user privacy and obtain some user attributes.

In this distributed electricity trading, users do not want to reveal their real identity, but it is acceptable to expose some information, such as reputation, location, etc. Compared with other traditional signature algorithms (RSA and ELGamal), the attribute-based signature algorithm can identify the user's identity with a set of attributes, so attribute-based signature is suitable for distributed electricity trading scenarios and can meet the user's identity privacy needs. According to our analysis, the attribute signature algorithm is a potential signature algorithm that can be used in this scenario, in which the signer uses the attribute information he holds instead of identity information to sign the message, and the verifier verifies the identity of the signer by verifying whether the signer has the claimed attributes. According to the characteristics of the attribute signature algorithm, we can represent the user's reputation and distance as attributes and use the corresponding private key to sign the transaction so that anyone can verify the attributes of the signer. Fortunately, the concept of a decentralized attribute signature scheme has been proposed in [14]. The scheme can be used in a distributed networks to support privacy-preserving electricity trading.

However, another issue to be considered is that user attributes always change dynamically over time. For example, the user's reputation will change as his trading behavior is good or bad. Therefore, the distributed attribute-based signature (DABS) algorithm can support the dynamic change of attributes. In addition, the computation efficiency of the key update that occurs when the attribute changes should not be low.

In a distributed electricity trading system, it not only requires peer-to-peer trading between users, but also need users to choose their counterparty according to their needs. However, most of the existing schemes require the participation of agents when conducting electricity trading, which makes it difficult to truly realize distributed electricity trading. In addition, most of the existing schemes only consider the impact of electricity price on the transaction, and do not rely on information such as the geographical location and reputation of the user. In summary, in this distributed electricity trading scenario, a suitable signature algorithm that can be used in the blockchain should have the following properties:

1) anonymity but support for attribute verification;
2) distributed and does not require the participation of central authority;
3) able to support dynamic update of attributes;
4) has high computational efficiency.

In this article, we propose a DABS algorithm, which can meet the aforementioned four requirements to well support distributed electricity trading.

A. Our Contribution

We outline the main contributions as the following points.

1) We propose a blockchain-assisted and privacy-preserving smart grid electricity trading scheme to support the free choice of trading users. In the proposed scheme, we embed the blockchain's timestamp into the signing key and design a signature verification method to achieve the goal that the verifiability of the signature is not affected by attribute changes. With blockchain's public and hard-to-tamper properties, distributed free electricity trading is realized and provides good support for trading auditing.
2) We construct a DABS to support electricity trading in the smart grid. In the DABS, we define a new definition for the generation of time period Time. The generation and change of time period are achieved by using the blockchain timestamp field. The DABS achieves signature unforgeability and resistance to collusion attacks by embedding the user's identity, Time, and attribute information into the attribute signing key. In terms of construction, the forward verifiability of the signature is achieved by leveraging the public and immutable nature of the blockchain.
3) To improve the efficiency of updating the attribute signing key, we extend the basic DABS scheme proposed to an advanced scheme. The update efficiency is reduced from $O(n)$ to $O(\log n)$, where n represents the number of users in the system.
4) We demonstrate the security and practicality of the DABS through performance analysis and simulation experiments.

B. Related Work

Attribute-based signature (ABS) [15], [16] is a signature algorithm that allows users to sign messages using any attribute issued from the attribute authority. The ABS was first constructed in [17], in which a scheme was proposed to support predicates described by a monotone span program. In order to improve the flexibility of the ABS, a threshold ABS was introduced in [18]. In the d threshold signature, if the signature attribute set A and the verified attribute set B have d intersections, then the signature can be successfully verified. In [16], the authors used a cloud server to propose an ABS with revocation. Specifically, if the signer is not revoked, the server will generate a signature for it, and once the signer is revoked, the server will refuse to generate a signature for it. In order to avoid the single point of failure of a single attribute authority (AA) and better implement ABS in a distributed network environment, the concept of multiattribute authorities (AAs) ABS was proposed. In AAs-based ABS, each AA controls a subset of attributes and is in charge of distributing keys for the attributes within the subset. The AA is completely honest and responsible for maintaining the system's attribute set, generating attribute public/private key pairs and attribute signing keys for the user. In [14], the authors presented the construction of the decentralized multiauthority ABS (DMA-ABS) scheme for the first time. Recently, in [19], the authors presented an attributed-based signature with multiauthorities to

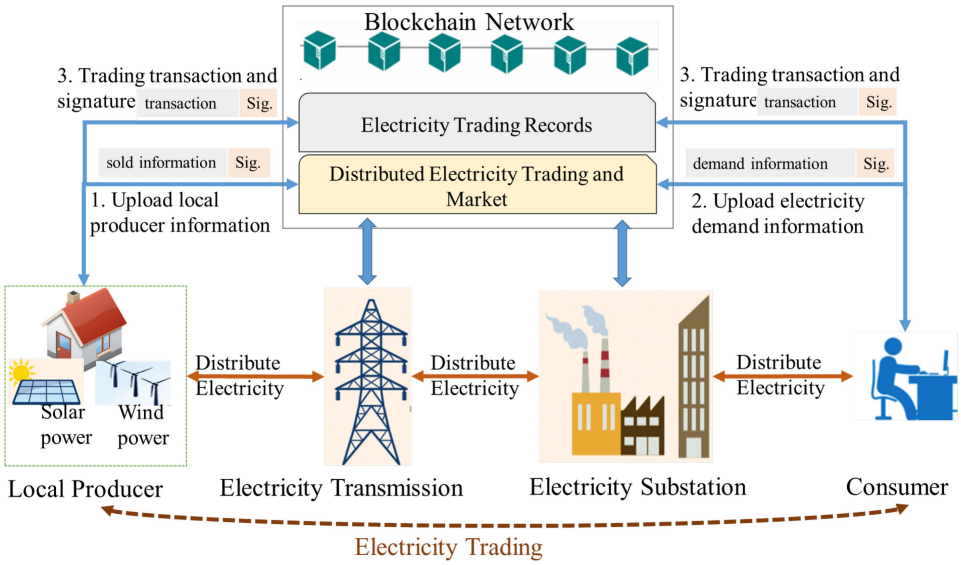

Fig. 1. System model.

guarantee the validity of Electronic Health Records encapsulated in blockchain. In [20], the author applied the decentralized ABS scheme to the healthcare blockchain to solve the secure storage of healthcare data. In [21], the authors proposed a revocable ABS for the blockchain-based healthcare system. In [22], the author applied attribute-based encryption (ABE) to solve the privacy protection problem of intelligent medical systems. With the assistance of edge computing, part of the encryption and decryption is outsourced to edge nodes, thereby improving the computing efficiency of resource-constrained devices. In [23], for the application of the vehicular network (VANET), the author combined ABE with blockchain technology to solve the access control problem of information in VANET. In [24], based on AAs and ABE technology, the authors constructed a data access control scheme supporting policy changes to address data security issues in cloud storage. In [25], the authors combined ABE and AAs to propose a keyword search scheme supporting attribute revocation to address multiowner and visited data sharing scenarios.

However, these solutions still have challenges in the face of blockchain and attribute changes. The frequency of key updates is not well defined, and the cost of the key updates is huge. The verifiability of the signature after attribute change has not been well explored. Therefore, the existing solutions are difficult to directly use in a distributed smart grid. In this article, we design a decentralized ABS with a new time period definition. By applying this signature, users in the smart grid are able to choose their counterparties based on a customized trading policy without revealing identity privacy.

II. System Model and Framework

A. System Model

As illustrated in Fig. 1, the system model consists of three entities: the AAs, the seller, and the consumer. Users (local Producer and consumer) in the system are assigned different attributes (price, location, electricity, electricity price, etc.). When the system is initialized, AAs assign different signing keys to users according to their attributes. A user's attributes can be verified with the attribute public key. Local producer publishes their electricity sale information with a signature signed with their signing key related to their attributes, such as price, location, and reputation, on a market. Consumers select trading objects according to their own needs (price, geographic location, reputation, etc.), and initiate transactions. Specifically, the transaction initiator also needs to sign the content of the transaction with their attribute signing key, and then, publish the signed transaction to the blockchain. The transaction can only be completed if both parties' transaction policies are satisfied. Once the transaction is verified and confirmed by most nodes in the blockchain network, the transaction is considered to be successful, and the Seller of the transaction needs to provide a power supply service that matches the content of the transaction.

1) *Attribute authorities (AAs):* The AAs are fully honest and responsible for maintaining the attribute set of the system (such as price, location, and reputation), generating attribute public/private key pairs, and attribute signing keys.
2) *Local producer:* The local producer is a user who wants to sell electricity. The local producer can define a fine-grained control policy to represent the attribute requirements for the counterparty. The local producer signs the transaction information and the transaction policy and uploads them to the public network.
3) *Consumer:* The consumer is a user who wants to buy electricity. They would like to select trading objects according to their own needs (price, geographic location, reputation, etc.), and initiate transactions with a signature related to their attributes. Once the local producer' information is received, the consumer performs a signature verification operation.

In our design, we use the consortium blockchain to build the electricity trading scheme. The motivation behind it is summarized as follows.

1) Authorized access: Considering the special nature of electricity trading, it is necessary to ensure that only authorized participants can participate in electricity trading to avoid malicious users' participation. Compared with the public blockchain, the consortium blockchain can be set up as a system in which only authorized users can participate.
2) High transaction throughput: There is a need for high throughput as well as low latency for electricity transactions. Compared with the public blockchain, only preselected nodes perform the consensus process in the consortium blockchain, which makes reaching consensus faster, thus increasing transaction throughput and reducing transaction latency.
3) Resource constraints: Individual users are not willing to waste extra computational resources for public chain mining (i.e., proof of work), but they have the ability to participate in the resources of the consortium blockchain.

Therefore, consortium blockchain are more feasible in our system.

In this model, we assume that entities such as AA and blockchain are fully trusted. An attacker can participate in the maintenance of the blockchain, and thus, obtain information about the transactions on the chain. Considering the consensus mechanism of the blockchain, it is therefore difficult for the attacker to tamper with the transaction information independently. In addition, an attacker or an adversary can broadcast any message with the purpose of sending messages that may be received by other users. Finally, an adversary may be able to learn the user's identity knowledge by monitoring the transactions.

B. Framework

Definition 1 (DABS): The DABS consists of the following algorithms: *GSetup*, *ASetup*, *AKeyGen*, *SignGen*, *SignVer*, and *Update*.

1) $GP \leftarrow GSetup(\lambda)$: The input is the security parameter λ, and the outputs is the global parameter *GP*.
2) $(\text{apk}, \text{ask}) \leftarrow ASetup(GP)$: The input is the global parameter *GP*. The outputs are the master key pairs apk and ask.
3) $\text{sk}^{\text{Time}}_{\text{att}_i,\text{GID}} \leftarrow AKeyGen(\text{GP}, \text{GID}, \text{Time}, \text{att}_i, \text{ask})$: The inputs are the global parameter *GP*, an identity *GID*, a time period *Time*, an attribute att_i and the attribute master private key ask of att_i. The output is the attribute signing key $\text{sk}^{\text{Time}}_{\text{att}_i,\text{GID}}$.
4) $\sigma \leftarrow SignGen(\text{GP}, m, (\mathbf{M}_{l\times n}, \rho), \text{Time}, \{\text{sk}^{\text{Time}}_{\text{att}_i,\text{GID}}\})$: The inputs are the global parameter *GP*, a message *m*, an access matrix $\mathbf{M}_{l\times n}$ with ρ mapping each row of matrix $\mathbf{M}_{l\times n}$ to an attribute, and the attribute signing keys $\{\text{sk}^{\text{Time}}_{\text{att}_i,\text{GID}}\}$. The output is the signature σ.
5) $0/1 \leftarrow SignVer(\text{GP}, m, \sigma, \{\text{apk}\}, (\mathbf{M}_{l\times n}, \rho), \text{Time})$: The inputs are the global parameter *GP*, a message *m*, a signature σ, an access matrix $\mathbf{M}_{l\times n}$ with ρ mapping each row of matrix $\mathbf{M}_{l\times n}$ to an attribute, the attribute master public keys $\{\text{apk}\}$ for the relevant attributes and *Time*. The output is $0/1$.
6) $0/1 \leftarrow \text{Update}(\text{GP}, \text{att}_i, \text{Time}, \text{GID})$: Inputs the global parameter *GP*, an attribute att_i, the timestamp Time and the revoked user identity GID, this algorithm outputs 0 or 1.

III. Proposed Scheme

A. Overview

When trading electricity in a smart grid, users want to be able to freely choose their trading partners without revealing their identities. To achieve this goal, we first try to use the ABS scheme. But the original ABS requires a central authority, which is not available in distributed systems. Motivated by the idea in the work of [26], which proposes an ABE scheme that periodically updates the attribute key. We consider using the idea of AAs and periodic updates to achieve this. However, if the idea of [26] is directly adopted, the following problems will arise. First, the revocation process is equivalent to regenerating the attribute signing key for the users, which is computationally expensive. Using the periodic key update approach also results in the signature that has been completed before the update not being verified correctly, thus resulting in a waste of computational and storage resources for the user. In addition, due to the distributed nature of the users, the synchronization of time is also a problem that needs to be solved.

In response to the aforementioned problems, we use the block structure and the nontamperable characteristics of the blockchain to propose a new time period calculation method to achieve a balance between efficiency and availability. Different from the fixed update period in work [26], our method is to use the information in the timestamp field of the block structure in the blockchain to realize the definition of the time update period. Here, we use an example to illustrate.

Example: Suppose there is a blockchain marked as $\mathbf{B}=\{B_0, B_1, B_2, \ldots, B_{n-1}\}$, where B_0 is the genesis block, each block B_i has a blockheader and a blockbody. The detailed description of the blockchain structure can be found in [27]. We use B_i.Timestamp() to represent the timestamp recorded in the blockheader of block B_i. Assume the latest block in the current blockchain is B_n, the current time period $\text{Time}_{\text{current}}$ can be expressed as: $\text{Time}_{\text{current}} = B_{n-1}.\text{Timestamp}()$.

This setting method can alleviate the key overhead problem caused by frequent updates, and it is easy to implement in a blockchain-based system. In addition, thanks to the nontampering feature of the blockchain, it can provide a reliable guarantee for the verification of signed transactions.

In terms of constructure, we first use the concept of AAs and the proposed time period to propose a distributed ABS scheme, which has unforgeability and user privacy. Compared with a single AA setting, AAs cannot only avoid single failures and centralized dependence problems, but also relieve the computational burden of a single attribute authority. To prevent collusion attacks between users, the hash value $H_1(\text{GID}||\text{Time})$ is calculated, where H_1 is a collision-resistant hash function and GID is the user's global identity. By embedding $H_1(\text{GID}||\text{Time})$ in the signature generation process, it not only guarantees that the users cannot collude, but also ensures that any party cannot

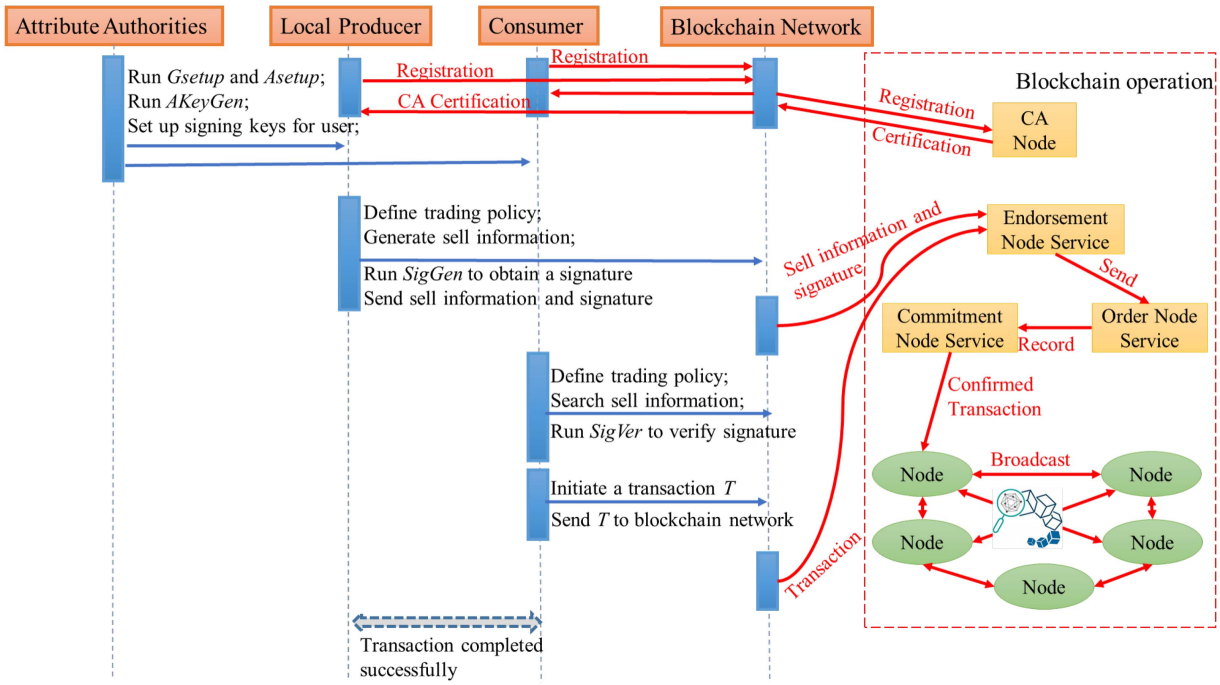

Fig. 2. Flow of the DABS scheme in the distributed electricity trading.

recover GID from the signature. In addition, the timestamp of the block is used as the time information in the generation and verification of the signature, thereby, the signature generated by the user in the previous time period is still valid and verifiable. However, in terms of updating, the first proposed scheme still requires a large computational cost. Motivated by the idea in the work of [21], we later used the KUNode algorithm to improve the update efficiency.

B. Basic Scheme of the DABS

The flow of the DABS scheme in building a distributed trading platform is shown in Fig. 2. We use a consortium blockchain (Hyperledger Fabric) as the platform, which contains certification authority (CA), endorser, committer, and order node types . When the system is initialized, the appropriate attribute secret key and the CA certificate are provided to the user. When the producer sends a sales request, the producer initializes the sales information and sends it to the endorser node. The endorser node verifies the signature as well as the information. If the sales request signature and information are correct, they are propagated to the order node and waits to be recorded in the blockchain. On the consumer side, the consumer's consumption request will also perform the aforementioned process to complete the record. In the trading phase, the consumer can check whether it meets the purchase requirement by the producer's signature. If it does, both parties make the trading transaction. Then, they submit the transaction containing the trading information and their signatures to the blockchain for recording. Next, we will show the detail of the algorithms included in the basic DABS scheme.

1) *GSetup:* This algorithm chooses two bilinear groups G and G_T, the group order of G and G_T is p, g_1 is one of a generators of G, a map function $e : G \times G \to G_T$ and two hash functions $H_0 : \{0,1\}^* \to Z_p$, $H_1 : \{0,1\}^* \to G$. The global public parameter is $\mathrm{GP} = (G, G_T, p, g_1, e, H_0, H_1)$.
2) *ASetup:* Each AA runs this algorithm and outputs attribute private key ask and attribute public key apk for each attribute maintained by it. For attribute att_i belonging to the AA_i, the AA_i randomly chooses $\alpha_i \in Z_p$, $y_i \in Z_p$ and computes $e(g_1, g_1)^{\alpha_i}$ and $g_1^{y_i}$. Set the attribute private key for att_i to $\mathrm{ask}_i = \{\alpha_i, y_i\}$. Set the attribute public key for att_i to $\mathrm{apk}_i = \{\mathrm{apk}_i^{(1)} = e(g_1, g_1)^{\alpha_i}, \mathrm{apk}_i^{(2)} = g_1^{y_i}\}$.
3) *AKeyGen:* Assume AA_i is responsible for attribute att_i. This algorithm is run by AA_i to generate or update the signing key $\mathrm{sk}_{\mathrm{att}_i,\mathrm{GID}}^{\mathrm{Time}}$ for GID with attribute $\{\mathrm{att}_i\}$ at period Time. AA_i computes: $\mathrm{sk}_{\mathrm{att}_i,\mathrm{GID}}^{\mathrm{Time}} = \{\mathrm{sk}^{(0)} = g_1^{\alpha_i H_0(\mathrm{Time})}$, and $\mathrm{sk}^{(1)} = H_1(\mathrm{Time}||\mathrm{GID})^{y_i}\}$. The attribute signing key for user GID with attribute $\{\mathrm{att}_i\}$ at period Time is $\mathrm{sk}_{\mathrm{att}_i,\mathrm{GID}}^{\mathrm{Time}}$.
4) *SignGen:* Define M_x as the xth row of the matrix $\mathbf{M}_{l\times n}$. To sign a message m under $(\mathbf{M}_{l\times n}, \rho)$, the signer randomly selects s in Z_p, a vector $\vec{v} = (v_1, \ldots, v_n)^{\mathrm{T}}$, a vector $\vec{w} = (w_1, \ldots, w_n)^{\mathrm{T}}$. Then, the singer sets s as the first element of the vector $\vec{v}$ and 0 as the first element of the vector $\vec{w}$. Let v_x denote $A_x\vec{v}$ and w_x denote $A_x\vec{w}$. Meanwhile, the signer randomly selects a number r_x in Z_p for each A_x. The signature generated by signer GID is $\sigma = \{\sigma_0, \sigma_{1,x}, \sigma_{2,x}\}$, where $\sigma_0 = g_1^{sH_0(m)}$, $\sigma_{1,x} = H_1(\mathrm{GID}||\mathrm{Time})^{r_x}$, and $\sigma_{2,x} = \frac{H_1(\mathrm{GID}||\mathrm{Time})^{w_x} g_1^{v_x}}{\mathrm{sk}^{(0)}(\mathrm{sk}^{(1)})^{r_x}}$.

5) *SignVer:* First, for each x, the verifier computes c_x such that: $\sum_x c_x A_x = (1, 0, \ldots, 0)$. Second, the verifier uses the hash function H_0 to obtain $H_0(m)$, where m is the signed message, and uses the hash function H_0 to obtain $H_0(\text{Time})$, where Time is the time period. Then, the verifier checks

$$e(g_1, \sigma_0)^{\frac{1}{H_0(m)}} = \prod_x \left(\left(\text{apk}_{\rho(x)}^{(1)} \right)^{H_0(\text{Time})} \cdot e\left(\sigma_{1,x}, \text{apk}_{\rho(x)}^{(2)} \right) \cdot e(g_1, \sigma_{2,x}) \right)^{c_x}. \tag{1}$$

If the aforementioned equation holds, *SignVer* outputs 1 to indicate that σ is valid. Otherwise, σ is invalid.

6) *Update:* In this scheme, the traditional update method is required to handle the attribute update problem, i.e., AAs need to reexecute the *ASetUp* and *AKeyGen* algorithms. It is obvious that the update complexity of this scheme is proportional to the number of users to be updated.

Remark Assuming there is a signature σ_{t_1} generated with time period Time_1 for a transaction Trans_1, the current time period is Time_c and $\text{Time}_1 < \text{Time}_c$, the verifier can verify σ_{t_1} at the current time period Time_c. At the time period Time_c, all transactions generated at the previous time period have been gathered into blocks, and the verifier can get the time period Time_1 of transaction Trans_1 by tracing the timestamp of the corresponding block in the blockchain. From (1), we can get the conclusion that as long as the attribute public key remains unchanged, the signature generated in the previous time period can also be verified. In the comparison schemes, the control of time period is managed by AAs, which are not only difficult to use in distributed scenarios, but also the verifiability of signed signatures is difficult to achieve. In contrast, we innovatively use the blockchain's timestamp field for the generation of the time period and use it in signature generation. Due to the invariance and public accessibility of the blockchain, the verifiability of the signature is not affected by the change in the time period.

C. Advanced Scheme of the DABS

To improve the efficiency of the key updates when user's attributes change, we extend the basic DABS scheme based on the idea of [21] and [28] to obtain an advanced scheme. The detailed construction of the advanced scheme is as follows.

1) *GSetup:* The process is the same as that in the *GSetup* phase of the basic DABS scheme.
2) *ASetup:* The process is the same as that in the *ASetup* phase of the basic DABS scheme except that each AA builds a binary tree BT_i with a state st_i and maintains a revocation list rl_i for attribute att_i.
3) *AKeyGen:* Assume AA_j is responsible for attribute att_i and have a corresponding binary tree BT_i with a state st_i. On input the global parameter *GP*, a binary tree BT_i with state st_i, a GID, an attribute att_i belonging to AA_j and attribute private key ask_i. AA_j selects a node η from the undefined leaf nodes of BT_i to store GID. For all $((\theta \in \text{Path}(\eta)) \bigwedge (\theta \text{ is undefined}))$, it chooses $r_{i,\theta} \in Z_p$ and stores $r_{i,\theta}$ in θ. For all $((\theta \in \text{Path}(\eta)) \bigwedge (\theta \text{ is defined}))$, it extracts $r_{i,\theta}$ directly. For attribute att_i, it updates the state st_i of BT_i and sends $\{\text{sk}^{(0)}, \text{sk}^{(1)}\}$ to GID, where $\text{sk}^{(0)} = \prod_\theta g_1^{r_{i,\theta}+\alpha_i}$, and $\text{sk}^{(1)} = H_1(\text{GID})^{y_i}$.
4) *UpKeyGen:* The inputs of this algorithm are as follows: the public parameter par, the attribute secret key ask_i, a time period Time, a revocation list rl_i, and a state st_i. For each node θ included in $Y = \text{KUNodes}(\text{BT}_i, \text{rl}_i, \text{Time})$, the AA chooses $r_{i,\theta} \in Z_p$ and stores $r_{i,\theta}$. Otherwise, it extracts $r_{i,\theta}$ directly. For calculating the update key, it chooses $s_{i,\theta} \in Z_p$ to computes: $uk^{(1)} = g^{\sum_\theta r_{i,\theta} H_0(\text{Time}) + \alpha_i}$, and $uk^{(2)} = H_1(\text{Time})^{\sum_\theta s_{i,\theta}}$. It sends $\{uk^{(0)}, uk^{(1)}\}$ to the user GID and updates the state st_i. Meanwhile, it makes $\text{Tpk}_i = g^{\sum_\theta s_{i,\theta}}$ public as the update public key at time Time for attribute att_i.
5) *SignGen:* The process is the same as that in the *SignGen* phase of the basic DABS except that the signature generated by signer GID is $\sigma = \{\sigma_0, \sigma_{1,x}, \sigma_{2,x}, \sigma_{3,x}\}$, where $\sigma_0 = g_1^{sH_0(m)}$, $\sigma_{1,x} = H_1(\text{GID})^{r_x}$, $\sigma_{2,x} = H_1(\text{Time})^{r_x}$, and $\sigma_{3,x} = \frac{H_1(\text{GID})^{w_x} g_1^{v_x} (\text{sk}^{(0)})^{H_0(\text{Time})}}{uk^{(0)^{-1}} uk^{(1)} \text{sk}^{(1)} g_1^{r_x}}$.
6) *SignVer:* The process is the same as that in the *SignVer* phase of the basic DABS except that checks

$$e(g_1, \sigma_0)^{\frac{1}{H_0(m)}} = \prod_x \left(\left(\text{apk}_{\rho(x)}^{(1)} \right)^{H_0(\text{Time})-1} e(g_1, \sigma_{3,x}) \cdot e\left(\sigma_{1,x}, \text{apk}_{\rho(x)}^{(2)} \right) e\left(\sigma_{2,x}, \text{Tpk}_{\rho(x)} \right) \right)^{c_x} \tag{2}$$

If (2) does not hold, it outputs 0 to indicate that this signature is invalid. Otherwise, it outputs 1, which demonstrates that this signature is valid.

D. Security Analysis

Our security concerns focus on the privacy of user identities and the distributed consensus of transactions, where the distributed consensus of transactions is obtained by using the blockchain. The security of the proposed scheme is guaranteed in two aspects: cryptographic algorithm and blockchain. For the security of the cryptographic algorithm, we formally prove the privacy and unforgeability of the proposed algorithm (please refer to Theorems 1 and 2). In addition, we discuss how the proposed algorithm can resist collusion attacks (please refer to Section III-D1). For the consensus problem of transactions in distributed scenarios, by using blockchain technology, the proposed scheme can resist the problem of single point of failure as well as the problem of distributed and consistent management of the transaction (please refer to Section III-D2).

Theorem 1: The proposed scheme is private if discrete logarithm (DL) [29] assumption holds in G .

Proof: The adversary $\mathcal{A}$ gives GID_0, GID_1 with attribute set $\{\text{att}_i\}$, a transaction Trans and the access structure $(\mathbf{A}, \rho)$ at time period Time to the challenger C, and the challenger C chooses a random toss $b \in \{0, 1\}$, and generates σ_b. Give σ_b to the

adversary $\mathcal{A}$, where $\sigma_b = \{\sigma_{0(b)} \| \sigma_{1,x(b)} \| \sigma_{2,x(b)}\}$, $b \in \{0,1\}$. If the advantage that the adversary $\mathcal{A}$ with polynomial computation ability to guess the correct b is negligible, the proposed scheme is said to be private.

Suppose that adversary $\mathcal{A}$ can break the privacy of the proposed scheme, and then, it means the adversary $\mathcal{A}$ has a nonnegligible advantage to guess the correct b in the above statement. More specifically, given GID_0 and GID_1 with attribute set $\{\text{att}_i\}$, the transaction Trans and the access structure $(\mathbf{A}, \rho)$ at time period Time, the adversary $\mathcal{A}$ has a nonnegligible advantage to distinguish the tuple σ_0 from σ_1. From the proposed scheme, we have $\sigma_0 = \{\sigma_{0(0)} \| \sigma_{1,x(0)} \| \sigma_{2,x(0)}\}$ and $\sigma_1 = \{\sigma_{0(1)} \| \sigma_{1,x(1)} \| \sigma_{2,x(1)}\}$, where $\sigma_{0(b)} = (g_1)^{s_b H_0(m)}$, $\sigma_{1,x(b)} = H_1(\text{GID}_b || \text{Time})^{r_{x(b)}}$, $\sigma_{2,x(b)} = \frac{H_1(\text{GID}||\text{Time})(g_1)^{v_x}}{\text{sk}^{(0)}\text{sk}^{(1)^{r_x}}}$, and $r_{x(b)}, w_{x(b)}$, and $v_{x(b)}$ are randomized values for $b = \{0,1\}$.

If the adversary $\mathcal{A}$ has the ability to distinguish σ_0 from σ_1, it means $\mathcal{A}$ breaks the DL problem, and it contradicts with DL assumption. Thus, it is impossible for $\mathcal{A}$ to distinguish σ_0 from σ_1 with a nonnegligible probability, and the proposed scheme is private. □

Theorem 2: The proposed scheme is unforgeable with the DL [29] assumption holding in G.

Proof: Suppose that an adversary $\mathcal{A}$ has a nonnegligible advantage in breaking the unforgeable of the proposed DABS-DC scheme, then, there exists a simulator $\mathcal{S}$ with nonnegligible advantage in breaking DL assumption. The simulator $\mathcal{S}$ is given an instance of the DL problem (a, g_1^a), where $a \in Z_p^*$, and is used to compute x. The detailed simulation is proceeded as follows.

Setup: A challenge $(\mathbf{A}, \rho)$ with attribute set $\{\text{att}_i\}^*$ is selected by the adversary $\mathcal{A}$. It sends a list of corrupted authorities L_A, $(\mathbf{A}, \rho)$ and $\{\text{att}_i\}^*$ to the simulator $\mathcal{S}$. Then, $\mathcal{S}$ gives g_1^a to $\mathcal{A}$.

Query: The simulator $\mathcal{S}$ initializes an empty list L, the adversary $\mathcal{A}$ is allowed to issue queries as follows.

1) *H-Query:* The simulator $\mathcal{S}$ maintains a list of L_H to gather the answers to the hash function oracle. For the queries the random oracle for $H_1(\text{GID}||\text{Time})$ form adversary $\mathcal{A}$, if $(\text{GID}, \text{Time}, H_1(\text{GID}||\text{Time})) \notin L_H$, $\mathcal{S}$ randomly chooses $x \in Z_p^*$, and set $H_1(\text{GID}||\text{Time}) = (g_1^a)^\gamma$. Then, $\mathcal{S}$ returns $H_1(\text{GID}||\text{Time})$ to adversary $\mathcal{A}$ after adding the tuple into L_H.
2) *Signing Key Query:* The simulator $\mathcal{S}$ maintains a list of L_S to gather the answers to the private signing key oracle. For the queries $(\text{att}_i, \text{Time}, \text{GID})$ to the AKGen & Update oracle from adversary $\mathcal{A}$, $\mathcal{S}$ response as follows. If $(\text{att}_i, \text{Time}, \text{GID}) \notin L_S$, $\mathcal{S}$ computes $\text{sk}_{\text{att}_i,\text{GID}}^{\text{Time}} = \{g_1^{\alpha_i H_0(\text{Time})}, (((g^a))^\gamma)^{y_i}$. Finally, $\mathcal{S}$ adding the entry $(\text{att}_i, \text{Time}, \text{GID}, \text{sk}_{\text{att}_i,\text{GID}}^{\text{Time}})$ in L_S and return $\text{sk}_{\text{att}_i,\text{GID}}^{\text{Time}}$ to $\mathcal{A}$.
3) *SignGen-Query:* The simulator $\mathcal{S}$ maintains a list of L_G to gather the answers to the signature generation oracle.The adversary $\mathcal{A}$ gives Trans^*, $(\mathbf{A}, \rho)^*$ and Time^* to the simulator $\mathcal{S}$, and the simulator $\mathcal{S}$ returns the signature $\sigma^* = \{\sigma_0^*, \sigma_{1,x}^* \sigma_{2,x}^*\}$ back to the adversary $\mathcal{A}$.
First, the simulator $\mathcal{S}$ selects a random number s^* in Z_p, a vector $\vec{v}^*$ with the first entry is s in Z_p^l, a vector $\vec{w}^*$ with the first element is 0 in Z_p^l. Let v_x^* denote $A_x \vec{v}^*$ and w_x^* denote $A_x \vec{w}^*$. Second, the simulator $\mathcal{S}$ selects a random number r_x^* in Z_p for each A_x, and then, calculates: $\sigma_0 = g_1^{s^* H_0(m)}$, $\sigma_{1,x} = ((g^a)^\gamma)^{r_x^*}$, $\sigma_{2,x} = \frac{(((g^a))^\gamma)^{w_x^* g_1^{v_x^*}}}{g_1^{\alpha_i H_0(\text{Time})}(((g^a))^\gamma)^{r_x^*}}$. At last, the simulator $\mathcal{S}$ returns the signature $\sigma^* = \{\sigma_0^*, \sigma_{1,x}^* \sigma_{2,x}^*\}$ back to adversary $\mathcal{A}$.

Forgery: The adversary $\mathcal{A}$ outputs a forged signature $\sigma^* = \{\sigma_0^*, \sigma_{1,x}^* \sigma_{2,x}^*\}$ on $(m, (\mathbf{A}, \rho)^*, \text{Time}^*)$. If the scheme is forgeable, then the submitted signature σ^* satisfies the verification, which means that

$$e(g_1, \sigma_0^*)^{\frac{1}{H_0(m)}} = \prod_x \left(\left(\text{apk}_{\rho(x)}^{(1)} \right)^{H_0(\text{Time})} e\left(\sigma_{1,x}^*, \text{apk}_{\rho(x)}^{(2)} \right) \cdot e(g_1, \sigma_{2,x}^*)^{c_x^* H_0(m)} \right) \tag{3}$$

where $c_x^* \in Z_p$ satisfies $\sum_x c_x^* A_x = (1, 0, \ldots, 0)$.

Assume that the adversary generate a signature σ, which can pass the verification. Then, it can generate another forged signature σ^*. We have

$$\sigma_0^{\frac{1}{H_0(m)}} = \prod_x \left(\left(\text{apk}_{\rho(x)}^{(1)} \right)^{H_0(\text{Time})} \cdot e\left(\sigma_{1,x}, \text{apk}_{\rho(x)}^{(2)} \right) \cdot \sigma_{2,x} \right)^{c_x} \tag{4}$$

where $c_x \in Z_p$ satisfies $\sum_x c_x A_x = (1, 0, \ldots, 0)$.

Considering the forged signature σ^* and a valid signature σ, let $\Delta s = s - s^*$, $\Delta r_x = r_x - r_x^*$, $\Delta w_x = w_x - w_x^*$, $\Delta\gamma = \gamma - \gamma^*$. For $\sum_x c_x A_x = (1, 0, \ldots, 0)$ and $\sum_x c_x^* A_x = (1, 0, \ldots, 0)$, we have $c_x^* = c_x$. Now dividing (4) by (3), we obtain $e(g,g)^{\Delta s} = e(g_1, g_1)^{\sum_x c_x \Delta v_x} e(g_1, g_1)^{\sum_x a \Delta\gamma c_x \Delta w_x}$. Thus, we can know that

$$e(g,g)^{\Delta s} = e(g_1, g_1)^{\sum_x c_x \Delta v_x + \sum_x a \Delta\gamma c_x \Delta w_x}. \tag{5}$$

From (5), we can know that $a = \frac{\Delta s - \sum_x c_x \Delta v_x}{\sum_x \Delta\gamma c_x \Delta w_x}$. Note that the probability of failure is the same as the probability of $\sum_x \Delta\gamma c_x \Delta w_x = 0 \mod p$. The probability of $\sum_x \Delta\gamma c_x \Delta w_x = 0 \mod p$ is $1/p$, which is negligible since p is a large prime. Therefore, we can solve the DL problem with a probability of $1 - 1/p$, which contradicts the assumption that the DL problem in G is computationally infeasible. Thus, it is impossible for $\mathcal{A}$ to forge a signature with a nonnegligible probability, and the proposed scheme is unforgeable with the DL assumption holding in G. □

1) Collusion Resistance: In our scheme, the identifier GID of a user is combined into an attribute signing keys from different AAs. For the collusion attack of users in the system, assume that there are two users GID_1 and GID_2. The attribute sets they own are S_1 and S_2, respectively, and they want to merge their attribute signing keys to forge a signature of the attribute set S, where $S \in \{S_1 \cup S_2\}$. Then, they need to compute $\text{sk}_{\text{att}_i,\text{GID}}^{\text{Time}}$ for each $i \in S$, where $\text{sk}_{\text{att}_i,\text{GID}}^{\text{Time}} = \{g_1^{\alpha_i H_0(\text{Time})}, H_1(\text{GID}||\text{Time})^{y_i}\}$. But, for each $i \in S_1$, the user with GID_1 only has $\text{sk}_{\text{att}_i,\text{GID}_1}^{\text{Time}} = \{g_1^{\alpha_i H_0(\text{Time})}, H_1(\text{GID}_1||\text{Time})^{y_i}\}$. For each $i \in S_2$, the user with GID_2 only has $\text{sk}_{\text{att}_i,\text{GID}_2}^{\text{Time}} = \{g_1^{\alpha_i H_0(\text{Time})}, H_1(\text{GID}_2||\text{Time})^{y_i}\}$. This will cause them to be unable to get the signature on the attribute set S. As a result,

TABLE I
COMPARISON OF PROPERTIES

Properties	[30]	[31]	[32]	DBAS
Decentralized	×	×	✓	✓
Privacy	×	✓	✓	✓
Unforgeable	✓	✓	✓	✓
Resistance Collusion	×	×	✓	✓
Update verifiability	×	×	×	✓

the collusion attack cannot be completed. Another situation is the collusion attack of the authority. Suppose the user's attribute signing keys come from m attribute authorities, when m attribute authorities are corrupted, they can forge a signature of an access structure with these attributes. However, when the number of authorities that are corrupted is less than or equal to $m-1$, the collusion attack cannot be completed. But this scheme cannot provide a good solution to the collusion attack between users and attribute authorities.

2) Transaction Security: The role of blockchain in our system is to provide an open and transparent trading platform. The security provided by blockchain can be divided into two aspects as follows.

a) Transaction records: In addressing the issue of transaction records, the consortium blockchain can ensure that the transactions will not leak outside the consortium, which can be achieved by establishing channels in simulated experiments. Second, each consensus node in the blockchain system stores a copy of the transaction record, which can ensure that the collapse of some nodes will not trigger the loss of transaction records. Therefore, the introduction of blockchain can avoid the risk of transaction records leaking outside the consortium system and also resist the single node failure problem, which exists in systems where blockchain is not introduced.

b) Transaction consensus: Consensus mechanism in the blockchain system can provide consistency guarantee of transaction records for distributed and central agency-free transaction scenarios. The Kafka consensus mechanism we use supports crash tolerance. Theoretically, as long as there is an order service node in the system that can run, the consensus of the system can be achieved. Therefore, the power blockchain-based trading platform built on Hyperledger Fabric has good crash tolerance. Although the Kafka consensus algorithm is only fault tolerant to crashes and not to malicious nodes, and its security depends on the authentication of the system itself, we believe that the Kafka mechanism is optional considering the practical scenario of power.

IV. PERFORMANCE ANALYSIS AND COMPARISON

A. Theoretical Comparison

We compare the features of our scheme and [30], [31], and [32]. As shown in Table I, the DABS is the only one that satisfies all of the following properties: decentralized, privacy, unforgeable, the resistance to collusion, and update verifiability. The update verifiability refers to once the user's attributes are changed, whether the signature generated with the previous attributes can still be correctly verified. In [30] and [31] use centralized AA, which does not have the characteristics of decentralization. Due to the lack of decentralization, the authors in [30] and [31] did not mention the characteristics of resistance to collusion attacks. Since the authors in [32] use a cloud server that cannot be fully trusted, it faces a single point of failure.

TABLE II
COMPARISON OF STORAGE OVERHEADS

	The size of signing key	The size of signature
DABS	$d\|G\| + d\|Z_p\|$	$(1+l)\|G\|$
[19]	$(1+2d)\|G\|$	$(6+l)\|G\|$
[20]	$2d\|G\|$	$(1+2l)\|G\|$
[30]	$(1+2d)\|G\|$	$(2+2l)\|G\|$
[31]	$(2d+2)\|G\|$	$(2+2l)\|G\|$
[32]	$(2d+1)\|G\|$	$(4+2d)\|G\|$

TABLE III
COMPARISON OF COMPUTATIONAL COST

	Sign	Verify
DABS	$(1+4l)T_e$	$(2l+1)T_p + lT_e$
[19]	dT_p	$(2d+1)T_p + T_e$
[20]	$(4l+1)T_p + (l+1)T_e$	$2lT_p + (3l+1)T_e$
[30]	$(4l+4)T_e$	$(5+2l)T_p + 2T_e$
[31]	$(2d+2)T_e$	$(2d+2)T_p + 2lT_e$
[32]	$(l+3d+3)T_e$	$(d+2)T_p + T_e$

We first compare the DABS scheme with [19], [20], [30], [31], and [32] in terms of storage and computational cost. The comparison results are displayed in Tables II and III, respectively. In the aforementioned tables, d represents the number of attribute sets, l represents the number of attributes related, and $|G|$ and $|Z_p|$ represent the bit length of the elements in G (or G_T) and Z_p. It is not difficult to see that the DABS has a lower storage overhead than the comparison schemes. After analysis, it is found that if $l < \frac{1}{2}d$, the DABS requires less computation than other schemes in terms of computational overhead. If $l > \frac{1}{2}d$, the DABS is only weaker than SDABS [31] in the verify phase. As can be seen from Table III, the execution times of *SignGen* and *SignVer* are a linear function of the number of attributes. Therefore, the time complexity of *SignGen* and *SignVer* algorithms are $O(l)$, where l represents the number of attributes involved in algorithm.

Then, we compare the DABS and the advanced scheme with respect to the signing key update phase. Suppose that before the update, the number of users with attributes is n. The number of users who have this attribute after the update is $n-r$. In the basic scheme, AAs redistribute secret keys to all users, and the complexity of this process is $O(n)$. In the extension scheme, we use the KUNodes algorithm to calculate the minimum set of users to be updated. This means that AAs do not need to distribute secret keys to all users. The KUNodes algorithm is constructed based on a binary tree, and the output is set Y, which is the smallest set to ensure update security. According to its algorithm property [33], the complexity of key update is $O(\log n)$.

B. Experimental Results

We evaluate the performance of the basic scheme DABS and the transaction processing efficiency of the federated blockchain through simulation experiments. First, for different stages of

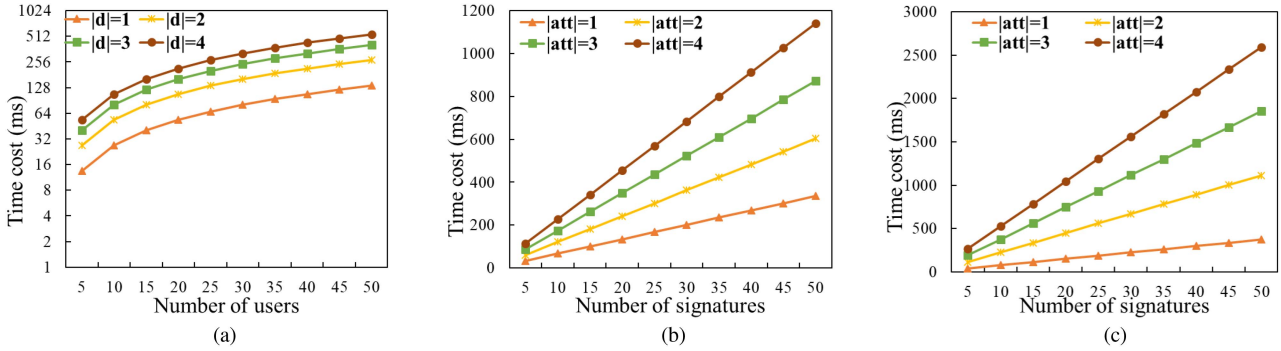

Fig. 3. Time cost of different phases. (a) Time cost of *AKeyGen*. (b) Time cost of *SignGen*. (c) Time cost of *SignVer*.

algorithms in the DABS, the main influencing factors are considered separately to obtain the average time of operation. Second, the electricity transactions are simulated based on the Hyperledger Fabric, focusing on the CPU usage at different transaction speeds. All the aforementioned simulations were run 100 times and their average values were used as the simulation result. The simulation experiments are run on a laptop computer with Intel Core i7-8750H CPU at 2.20 GHz and 8.00-GB RAM hardware configuration running on 64-bit Ubuntu 14.04 running on a virtual machine VMware Workstation Player (with initial settings of 1-GB RAM). The simulations use hypersingular curves $y^2 = x^3 + x$ over a 512-bit finite field and A-curves over a 160-bit elliptic curve set. We used Hyperledger Fabric as the application platform. Specifically, we built the fabric blockchain platform on an Ubuntu system where we simulated the implementation of a distributed network environment using multiple Docker containers. In this simulation, we set up multiple participants, five peer nodes, and four order nodes. We make the nodes start by modifying the configuration file.

First, we set $|\text{att}| = 1, 2, 3, 4$, and analyze the computational cost of *AKeyGen*, *SignGen*, *SignVer* in the DABS. As shown in Fig. 3(a), when the number of attributes is a fixed value, the time cost of the *AKeyGen* algorithm increases as the number of users increases. When the number of users is a fixed value, the time cost of the *AKeyGen* algorithm increases with the increase in the number of attributes. Then, we evaluated the computational time of *SignGen*. As shown in Fig. 3(b), we can see that: as the relevant attributes increase, the calculation time of the *SignGen* algorithm also increases. When the number of attributes is constant, the more the number of signatures, the longer the time cost required by the algorithm. It can be seen from Fig. 3(c) that with the increase of related attributes, the computational time of the *SignVer* algorithm also increases. When the number of attributes is constant, the more signatures that need to be verified, the longer the time overhead required by the algorithm.

Second, the number of attributes held by the signer is set to 3, and the number of users is set to 6. From Fig. 4, it can be seen that when the number of users and the number of attributes are fixed, the cost of our scheme is less than that of the other three schemes. Specifically, in the DABS, the time consumed in the three phases is 48.4, 32.2, and 228.4 ms, respectively. In [19], the time consumed is 96.6, 178.7, and 769.7 ms, respectively. In [20], the time consumed is 48.3, 423.2, and 233.7 ms, respectively. In [30], the time consumed is 54.4, 72.6, and 266.2 ms, respectively. In [31], the time consumed is 120.9, 48.6, and 322.5 ms, respectively. In [32], the time consumed is 72.5, 129.4, 228.4 ms, respectively.

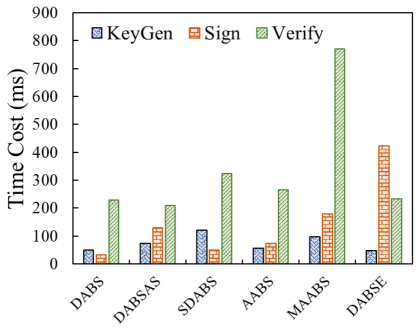

Fig. 4. Comparison of different phases.

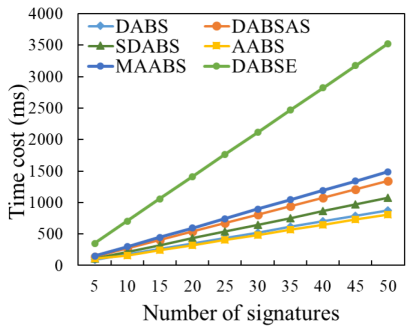

Fig. 5. Comparison of *SignGen*.

We can conclude that DABS has advantages in the *AKeyGen*, *SignGen*, and *SignVer* phases compared to the other three schemes.

After that, we use the number of signatures as a variable to simulate *SignGen* and *SignVer*. The number of attributes involved is set to 3, the number of signatures varies from 5 to 50, and the results are shown in Figs. 5 and 6. For *SignGen*, the time cost of the DABS changes from 87.4 to 875.1 ms, while in [19], [20], [30], [31], and [32], the time overhead are from 148.9 to 152.1 ms, 352.1 to 356.1 ms, 104.7 to 806.5 ms, 107.7 to

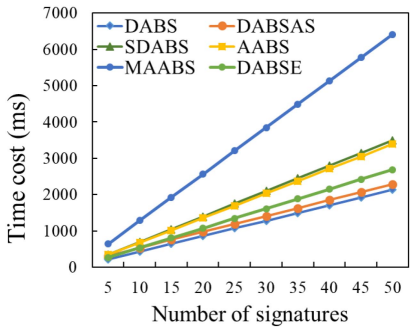

Fig. 6. Comparison of *SignVer*.

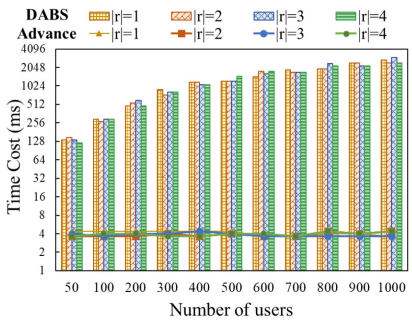

Fig. 7. Time cost of signing key update.

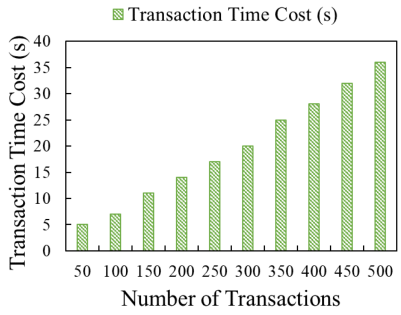

Fig. 8. Transaction time cost in blockchain.

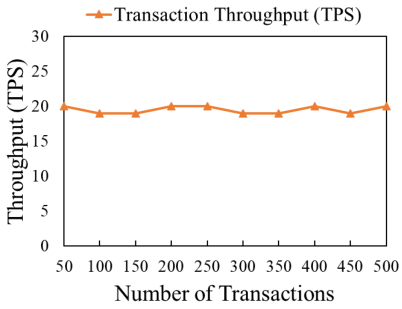

Fig. 9. Transaction throughput in blockchain.

1084.8 ms, and 134.2 to 1327.0 ms, respectively. For *SignVer*, the time cost of the DABS changes from 192.3 to 2013.5 ms, while in [19], [20], [30], [31], and [32], the time overhead are from 641.2 to 6415.2 ms, 268.62 to 2618.2 ms, 333.2 to 2453.2 ms, 348.4 to 3145.2 ms, and 318.8 to 2356.2 ms, respectively. The time cost of SigVer is positively correlated with the number of signatures, but our scheme has a significantly slower growth rate in terms of time cost.

In order to show the comparison between the extended scheme and the basic scheme in terms of key update, we performed an experimental simulation of the signing key update, and the experimental results are shown in Fig. 7. Due to the use of the KUNode idea [28] in the extension algorithm, the minimum set of all users who have not been revoked can be obtained. We assume that the number of system users is n and the number of users whose some attribute is revoked is r. According to the execution process of the KUNode algorithm [19], [31], it is known that the set of users to be updated by the extended algorithm is maximum when $r = n/2$ and these users are adjacent leaf nodes in the KUnodes algorithm. In the worst condition, the complexity of the UPKeyGen algorithm is $O(\log n)$. In the basic DABS scheme, the algorithmic complexity required for the update is $O(n)$. Therefore, the designed extension scheme can reduce the complexity of the update operation from $O(n)$ to $O(\log n)$. In the experiment, we set the number of nonrevoked users to vary from 50 to 1000. Afterward, simulations were carried out for the cases where the number of revoked users $|r| = 1, 2, 3, 4$, respectively. We can see that the advanced scheme requires much less computational overhead than the basic scheme when performing signing key updates. This is due to the fact that the computational overhead of the update phase in the basic scheme is linearly related to the un-revoked users, while the computational overhead of the update phase in the advanced scheme is independent of the un-revoked users.

In blockchain simulation, we mainly focus on the performance parameters of throughput and time cost. Throughput mainly refers to the number of transactions per second that successfully reach the blockchain system. The main factors that affect throughput include server performance, number of nodes, consensus algorithm, etc. The time cost mainly includes the network waiting time for initiating request and receiving response, the execution time for encryption, decryption, and signature, the time for reaching consensus between nodes, and the time for sorting and verification. We set the number of users to 5 and the transaction submit speeds to 20 transactions per second. After that, performance tests are conducted for different transaction volumes. As shown in Fig. 8, the time cost basically increases with the number of upload transactions, and its maximum value is about 35–40 s. In addition, the test result for throughput is shown in Fig. 9. When the number of users is 5 and the transaction submit speeds is 20 transactions per second, the throughput is close to 20. This means that the transaction upload request to the blockchain can basically be processed in time. Finally, we simulated a distributed electricity trading system based on Hyperledger Fabric. In this simulation, we focus on the impact of the number of transactions on the CPU usage. We monitored the CPU usage with transaction submit speeds of 15, 20, and 25, respectively. From Fig. 10, we can see that the CPU is almost fully used when the total number of

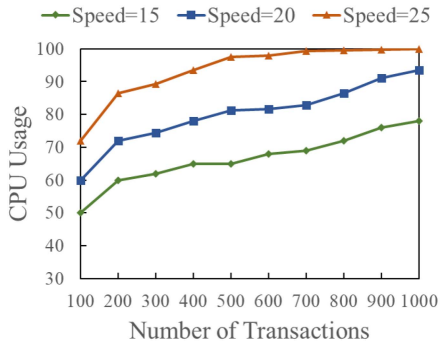

Fig. 10. CPU usage.

transactions reaches 1000. This indicates that the power trading system running on the blockchain requires the high performance of the participants' CPUs, a compromise made to achieve a secure and free exchange.

In the DABS, a transaction is valid only if the block containing the transaction is confirmed. Therefore, different consensus mechanisms can affect the speed of transactions. Moreover, as our experimental results show, if the user is both the trader and the maintainer of the blockchain, the CPU usage is almost full when the transaction is faster, which indicates that the scheme requires more hardware for the user. However, considering the actual situation, a more efficient consensus mechanism can be selected in the consortium blockchain, thereby improving the consensus efficiency and increasing transaction throughput. In addition, due to the high reliability of the blockchain, users can participate as verifiers, which avoids consuming a lot of resources for consensus and avoids high requirements for hardware. Therefore, we believe that the proposed scheme strikes a balance between the performance and security requirements, and the cost incurred is tolerable.

In future work, we hope to investigate the performance of consensus protocols as well as transaction validation as a way to improve the efficiency of blockchain operation. On the other hand, since the simulation experiments are deployed on a single server, there are inevitably hardware performance bottlenecks. In practical applications, as the performance and number of devices develop, the performance of the federated blockchain will also improve greatly, which is beneficial to the development of distributed electricity trading.

V. Conclusion

In this article, we proposed a DABS scheme that supports users to freely select trading partners in a way that protects identity privacy when trading electricity in smart grid. In DBAS, a new definition was made for the update cycle, which was combined with the timestamp field of the blockchain. With the help of the openness and nontamperability of the blockchain, this method realized the forward verifiability of the signature and balances the function and efficiency. Although the validity of the transaction was related to whether the block was confirmed or not, and the scheme had certain hardware requirements. However, considering the practicalities, users can still get a secure and efficient power trading service with low overhead. In order to further improve the update efficiency, an advanced DABS was proposed using the KUNodes algorithm. Finally, experimental analysis showed that the DABS achieved the expected efficiency. In future work, we hope to further reduce the amount of signature and verification calculations required by users, and at the same time, research more efficient blockchain consensus algorithms to ensure the safety and efficiency of power transactions.

References

[1] L. D. Xu, W. He, and S. Li, "Internet of Things in industries: A survey," *IEEE Trans. Ind. Inform.*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014.
[2] H. Zhang, J. Yu, C. Tian, G. Xu, P. Gao, and J. Lin, "Practical and secure outsourcing algorithms for solving quadratic congruences in Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2968–2981, Apr. 2020.
[3] G. Dileep, "A survey on smart grid technologies and applications," *Renewable Energy*, vol. 146, pp. 2589–2625, 2020.
[4] X. Fang, S. Misra, G. Xue, and D. Yang, "Smart grid–the new and improved power grid: A survey," *IEEE Commun. Surv. Tuts.*, vol. 14, no. 4, pp. 944–980, Oct.–Dec. 2012.
[5] G. S. Aujla, N. Kumar, M. Singh, and A. Y. Zomaya, "Energy trading with dynamic pricing for electric vehicles in a smart city environment," *J. Parallel Distrib. Comput.*, vol. 127, pp. 169–183, 2019.
[6] Z. Li, J. Kang, R. Yu, D. Ye, Q. Deng, and Y. Zhang, "Consortium blockchain for secure energy trading in industrial Internet of Things," *IEEE Trans. Ind. Inform.*, vol. 14, no. 8, pp. 3690–3700, Aug. 2018.
[7] D. Liu, J. Xiao, X. Yuan, Y. Zheng, and H. Cao, "A dynamic energy trading and management algorithm for the elastic end-user in smart grids," in *Proc. IEEE Globecom Workshops*, 2020, pp. 1–6.
[8] P. Tasatanattakool and C. Techapanupreeda, "Blockchain: Challenges and applications," in *Proc. IEEE Int. Conf. Inf. Netw.*, Jan. 2018, vol. 1, pp. 473–475.
[9] H. Zhang, P. Gao, J. Yu, J. Lin, and N. N. Xiong, "Machine learning on cloud with blockchain: A secure, verifiable and fair approach to outsource the linear regression," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 6, pp. 3956–3967, Nov.–Dec. 2022.
[10] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in IoV-assisted smart city," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1373–1385, Jul.–Sep. 2021.
[11] Y. Cao, X. Ren, C. Qiu, X. Wang, H. Yao, and F. R. Yu, "A multi-agent reinforcement learning approach for blockchain-based electricity trading system," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.
[12] F. Luo, Z. Y. Dong, G. Liang, J. Murata, and Z. Xu, "A distributed electricity trading system in active distribution networks based on multi-agent coalition and blockchain," *IEEE Trans. Power Syst.*, vol. 34, no. 5, pp. 4097–4108, Sep. 2019.
[13] J. Guo, X. Ding, and W. Wu, "A blockchain-enabled ecosystem for distributed electricity trading in smart city," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 2040–2050, Feb. 2021.
[14] T. Okamoto and K. Takashima, "Decentralized attribute-based signatures," in *Proc. Int. Workshop Public Key Cryptogr.*, 2013, pp. 125–142.
[15] S. Ruj, S. Milos, and N. Amiya, "Decentralized access control with anonymous authentication of data stored in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 384–394, Feb. 2014.
[16] H. Cui, R. H. Deng, J. K. Liu, X. Yi, and Y. Li, "Server-aided attribute-based signature with revocation for resource-constrained industrial-Internet-of-Things devices," *IEEE Trans. Ind. Inform.*, vol. 14, no. 8, pp. 3724–3732, Aug. 2018.
[17] H. K. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures: Achieving attribute-privacy and collusion-resistance," *Cryptol. ePrint Arch.*, vol. 2008, 2008, Art. no. 328.
[18] S. F. Shahandashti and R. Safavi-Naini, "Threshold attribute-based signatures and their application to anonymous credential systems," in *Proc. Int. Conf. Cryptol. Afr.*, 2009, pp. 198–216.
[19] R. Guo, H. Shi, Q. Zhao, and D. Zheng, "Secure attribute-based signature scheme with multiple authorities for blockchain in electronic health records systems," *IEEE Access*, vol. 6, pp. 11676–11686, 2018.
[20] Y. Sun, R. Zhang, X. Wang, K. Gao, and L. Liu, "A decentralizing attribute-based signature for healthcare blockchain," in *Proc. IEEE 27th Int. Conf. Comput. Commun. Netw.*, 2018, pp. 1–9.

[21] Q. Su, R. Zhang, R. Xue, and P. Li, "Revocable attribute-based signature for blockchain-based healthcare system," *IEEE Access*, vol. 8, pp. 127884–127896, 2020.
[22] H. Zhong, Y. Zhou, Q. Zhang, Y. Xu, and J. Cui, "An efficient and outsourcing-supported attribute-based access control scheme for edge-enabled smart healthcare," *Future Gener. Comput. Syst.*, vol. 115, pp. 486–496, 2021.
[23] J. Ma, T. Li, J. Cui, Z. Ying, and J. Cheng, "Attribute-based secure announcement sharing among vehicles using blockchain," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10873–10883, 2021.
[24] H. Zhong, W. Zhu, Y. Xu, and J. Cui, "Multi-authority attribute-based encryption access control scheme with policy hidden for cloud storage," *Soft Comput.*, vol. 22, no. 1, pp. 243–251, 2018.
[25] J. Cui, H. Zhou, H. Zhong, and Y. Xu, "AKSER: Attribute-based keyword search with efficient revocation in cloud computing," *Inf. Sci.*, vol. 423, pp. 343–352, 2018.
[26] H. Cui and R. Deng, "Revocable and decentralized attribute-based encryption," *Comput. J.*, vol. 59, pp. 1220–1235, Feb. 2016.
[27] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Commun. Surv. Tuts.*, vol. 21, no. 4, pp. 3796–3838, Oct.–Dec. 2019.
[28] H. Cui, R. H. Deng, and G. Wang, "An attribute-based framework for secure communications in vehicular ad hoc networks," *IEEE/ACM Trans. Netw.*, vol. 27, no. 2, pp. 721–733, Apr. 2019.
[29] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL, USA:CRC Press, 2018.
[30] J. Zhang, T. Li, M. S. Obaidat, C. Lin, and J. Ma, "Enabling efficient data sharing with auditable user revocation for IoV systems," *IEEE Syst. J.*, vol. 16, no. 1, pp. 1355–1366, Mar. 2022.
[31] Y. Li, X. Chen, Y. Yin, J. Wan, and Z. Dong, "SDABS: A flexible and efficient multi-authority hybrid attribute-based signature scheme in edge environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1892–1906, Mar. 2021.
[32] J. Li, Y. Chen, J. Han, C. Liu, Y. Zhang, and H. Wang, "Decentralized attribute-based server-aid signature in the Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4573–4583, Mar. 2022.
[33] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based encryption with efficient revocation," in *Proc. 15th ACM Conf. Comput. Commun. Secur.*, 2008, pp. 417–426.

Qianqian Su received the Ph.D. degree in cyberspace security from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2021 .

She is currently working with the School of Computer Science and Technology, Qingdao University, Qingdao, China. Her research interests include cloud computing security, blockchain, and data security.

Rui Zhang was born in 1981. She received the Ph.D. degree in information security from Beijing Jiaotong University, Beijing, China, in 2011.

She is currently an Assistant Researcher with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing. Her research interests include cloud data security, privacy preservation, and security protocols.

Rui Xue was born in 1963. He received the Ph.D. degree in mathematics from the Beijing Normal University, Beijing, China, in 1999.

He is currently a Full Research Professor and Vice Director with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing. He has authored and co-authored more than a hundred papers. His research interests include information security and privacy in data and information systems, with a focus on public-key encryption and cryptographic protocols.

Dr. Xue is a Member of the ACM. He serves also as the Vice Director Member of security protocols association in Cryptographic Association in China.

You Sun received the B.S. degree in cyberspace security from the Software College, Northeastern University, Shenyang, China, in 2017, and received the Ph.D. degree in cyberspace security from the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China, in 2022.

She works with the Information Science Academy, China Electronics Technology Group Corporation, Beijing. Her research interests include blockchain technology.

Sheng Gao received the B.S. degree in information and computation science from the Xian University of Posts and Telecommunications, Xi'an, China, in 2009, and the Ph.D. degree in computer science and technology from Xidian University, Xi'an, in 2014.

He is an Assistant Professor with the School of Information, Central University of Finance and Economics, Beijing, China. His current research interests include finance information security and privacy computing.