Blockchain-Oriented Data Exchange Protocol With Traceability and Revocation for Smart Grid

Prince Silas Kwesi Oberko[1], Tianang Yao[1], Hu Xiong[1,2,3,4], Saru Kumari[5], Sachin Kumar[6]*

[1] Network and Data Security Key Laboratory of Sichuan Province, University of Electronic Science and Technology of China
[2] The State Key Laboratory of Networking and Switching Technology,
Beijing University of Posts and Telecommunications, China
[3] Advanced Cryptography and System Security Key Laboratory of Sichuan Province
[4] Intelligent Terminal Key Laboratory of Sichuan Province, China
[5] Department of Mathematics, Chaudhary Charan Singh University, India
[6] Department of Computer Science and Engineering, Ajay Kumar Garg Engineering College, India
princesilasoberko@hotmail.com, tianangyao@gmail.com, xionghu.uestc@gmail.com,
saryusiirohi@gmail.com, imsachingupta@rediffmail.com

Abstract

The smart energy system (SES) encourages data administration and information services developments, particularly smart grids. Presently, numerous SESs cloud environments are accessible to smart grids. Nonetheless, because of the semi-credible character of the SES cloud environments, achieving secured access, information storage, updates, deletion, tracing, and revocation of ill-disposed clients is a genuine concern. In this publication, an Ethereum blockchain-oriented secured access regulation design upholding traceability and revocability is offered for smart grids to resolve these problems. The blockchain implements unified identity verification and saves all public-keys, users' attribute sets, and revocable lists. The system administrator creates system parameters and sends private-keys to users. The domain administrator prepares the domain's security and privacy-preservation policies and executes encryption procedures. If the attributes correspond with the access policy and the user's ID is unrevoked, they could acquire interim-decryption capabilities from the edge/cloud servers. Tracking malevolent users for revocation is applicable throughout all stages, ensuring the system is secured under Decisional-Bilinear-Diffie-Hellman (DBDH) complex theory and can withstand multi-attacks. Analysis revealed the size of the public/private keys to be shorter, contrary to relevant schemes. The overhead duration is less for generating the public-key, data encryption, and decryption phases.

Keywords: Blockchain, Smart grid, Privacy-preserving, Data exchange, Revocation

1 Introduction

Intelligent sensor networks [1] render diverse opportunities for smart grid applications, comprising power monitoring, demand-end energy administration, distributed storage harmonization, and renewable energy generators integration. A broad spectrum of sensors embedded in every grid domain and application is utilized to gather data within the smart grid ecosystem. This promotes the functionality and utility of big data in grids, termed big data in smart grids (BDSG) [2]. BDSG has been broadly investigated owing to its colossal applicable benefit to workers, administrators, and policymakers, notably, how BDSG can be shared securely while protecting clients' privacy on the IIoT/IoT cloud platforms [3-5].

Nevertheless, the cloud-assisted [6] platforms have significant security risks when managing grid-generated information as a semi-trusted entity. Administrative rights could be exploited, and private information leaks from properties inherent to the edge/cloud computing architecture, such as heterogeneity, mobility, geo-dispensation, and location information. Several data encryption techniques have been employed to secure cloud data outsourcing to resolve data confidentiality and integrity dilemmas [7-11]. Attribute-based encryption is an example of access regulation technologies for untrustworthy cloud storage services [12-13]. Related cryptographic systems have been suggested previously to accomplish confidentiality and protect privacy [14-18]. However, various concerns exist, expressly concerning malevolent user tracking and revocation.

With its initial release in 2009, crypto-currency, which depends on a public distributed ledger called blockchain, arose, and thereafter, applying blockchain technology to cloud-assisted IoT/IIoT became achievable [19-20]. Blockchain is a chronological, dispersed, verifiable, and tamper-proof ledger. Its nodes exchange data, collectively sustain the ledger, and utilize consensus mechanisms to guarantee data constancy [21]. However, blockchain is still not profoundly incorporated with smart grids' generating, managing, operating, and selling as an emerging security technology. In smart grids, data must be exchanged across the overall production-transmission chain, like consumption data, power costs, monitor sudden power behavior, etc. These data are chronologically timed and need to be tracked during the entire process. Concurrently, it demands to be tamper-proof, inimitable, and trackable. Blockchain possesses the proper qualities to match

*Corresponding Author: Prince Silas Kwesi Oberko; E-mail: princesilasoberko@hotmail.com
DOI: 10.53106/160792642023032402026

the demands of a smart grid [19-20, 22].

This article proposes a blockchain-oriented secured access regulation scheme that supports a smart grid's traceability and revocability. This paper's main contributions are as follows. (1) A ciphertext-policy attribute-based access regulation design for the smart grid that sustains traceability and revocation is presented and proved secure utilizing the DBDH assumption. Rivaled with other systems, the public/private keys size is smaller, and the duration overhead is more negligible in the public key generation, data encryption, and decryption phases. (2) Blockchain technology is applied to execute unified identity authentication and store all public keys, user attribute sets, and revoked user lists. Each user needs to be registered onto it. The blockchain in this proffered design is an alliance chain.

1.1 System Architecture

The framework of the proffered secure privacy-preserving BDSG Data Exchange Protocol is depicted in Figure 1. Five entities are entailed in the information flow: Grid Operator (GO), Edge Server, Distribution Grid Domains (DGD), Cloud Services Provider, and Blockchain.

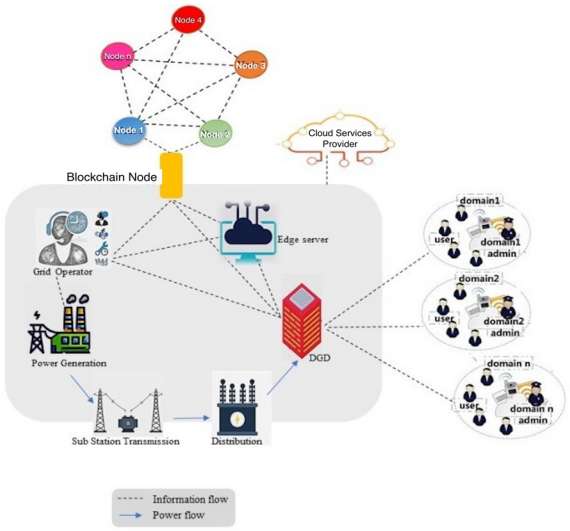

Figure 1. System architecture

1.2 Related Work

Access regulatory schemes built on attribute-based encryption are exhibited [7-10]. Ostrovsky [8] proposed a "one-to-multiple" outsourcing encryption system supporting a monotone access structure. An ABE system with proxy re-encryption to resolve the lack of tracing and revocation in cloud environments is proposed by [7]. Yan and Meng [10] offered an access regulation design with direct revocation, where the revocation list is inserted in the ciphertext during encryption. Nonetheless, scheme [10], together with [9], cannot handle the updating enigma of the access policy. Within the designs of [7-10], both the encryptor and decryptor bear hefty computational overheads.

A combination of blockchain technology and smart grids are evolving. Mengelkamp et al. [22] combined classical IIoT architecture and blockchain engineering to construct a smart grid disseminated network. Gao et al. [23] developed a sovereign-blockchain technology to alleviate smart grid consumer-data tampering. Agung et al. [24] devised an intelligent grid with secured immutable transactions, utilizing blockchain as a tool. Gai et al. [25] proposed a consortium-blockchain-oriented technique to solve the situation of privacy leakage within smart grid. Wang et al. [26] integrated blockchain, Elliptic Curve Cryptography (ECC), dynamic Join-and-Exit mechanism, and batch verification to propose an authentication protocol for smart meters and utility centers.

2 Design: Overview and Construction

2.1 Design Overview

Table 1 summarizes notations used in this work. The implementing processes of the smart grid design depicted in Figure 2, are as follows:

Table 1. Notations used

Notation	Meaning
λ	Security parameter
PK_s, MK_s	System Public and Master key
W	Registration List
L_{id}, Uid	Attribute Set and Identification of i
T, T^*	Access Structure
RvL RvL^*	List of Revoked Users
U	User Set
$\rho, \mathcal{M}$	Access Policy
CT	Ciphertext
SK_{id}	Private-Key
m	Plaintext
Υ_{id}	User's Certificate

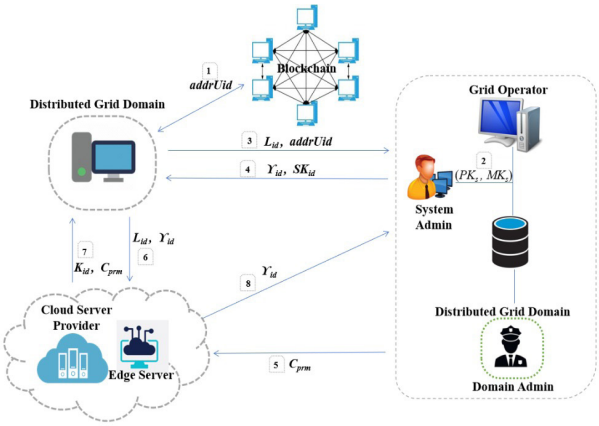

Figure 2. Process framework of the smart grid system

① Users register to blockchain and apply for account address as an Identification termed *addrUid*.

② *GO* picks λ arbitrarily and utilizes *Setup*(λ) to obtain PK_s, and MK_s according to the system's attributes universe L_s,

then transmit PK_s and L_s to the blockchain.

③ The Users send L_{id} and *Uid* to the *GO* for registration.

④ *GO* uses *KeyGen*() to generate (Υ_{id}, SK_{id}), assign it to the user per his L_{id}, and transfer Υ_{id} to blockchain.

⑤ The grid domain admin picks PK_s, and *RvL* from blockchain, forms a matching access structure ($\rho, \mathcal{M}$), formulate an attribute matrix $\mathcal{M}$, and invokes *Encrypt*() to return the ciphertext *CT*. Where the data is temporary, it saves to *ES*; else, convey to *CSP*, and convey ($\rho, \mathcal{M}$) to the blockchain.

⑥ The user inquires into the grid domain's data and sends Υ_{id} and L_{id} to *ES* or *CSP*.

⑦ *ES* or *CSP* acquires PK_s from blockchain and performs a policy comparison; if it matches, it delivers the decryption parameter to the user, who derives the plaintext utilizing algorithm *Decrypt*().

⑧ *CSP* conveys Υ_{id} to *GO*, and *GO* performs *Trace*() to investigate malevolent users. When a malevolent user is traced by the system, the user's *addrID* is included on the revocation list, then forwarded to the blockchain. *ES*/*CSP* then re-encrypts the data.

2.2 Design Construction

A scheme composed of nine stages is presented: initialization, system-setups, key-generation, data encryptions, decryptions, malevolent-user tracing, data re-encryptions, user updates private-keys, and *CSP*/*ES* updates saved private-keys of the users.

Stage 1. Initialization:

This is the blockchain registration stage. The user enrolls on blockchain by setting the user account password, *pswd* to receive the account public and private key combination (*pubK*, *prvK*). Then, users acquire the account address (*addrUid*), formed according to *pubK*, and transmit *addrUid* to the blockchain, where *addrUid* ← *IDGen*(*password*) and *IDGen*() generates an account address from the user's account password.

Stage 2. System-setup:

Setup: The algorithm formulates MK_s and PK_s. Select two bilinear groups ($\mathbb{G}$, $\mathbb{G}_T$) of order p (prime) and establish a bilinear map $\hat{e}: \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$, where g is the generator. Set $U = \{addrUid | 1 \le i \le n\}$ as the user set. The query list W with the initial state ϕ comprises two items (*randomparameter* and *addrID*). Select a hash function $H:\{0,1\} \to \mathbb{G}$, $H': \mathbb{G}_T \to \mathbb{Z}_p^*$, pick two arbitrary numbers $\alpha, \beta \in \mathbb{Z}_p$. Correspondingly, the system outlets MK_s and PK_s, for $MK_s = \{\alpha, \beta\}$ and $PK_s = \{\mathbb{G}, \mathbb{G}_T, g, \hat{e}(g,g)^{\alpha}, g^{\beta}, H(), H'()\}$. The Grid Operator secretly keeps MK_s. PK_s is transferred to blockchain by running *Save_Transaction*().

GridDomainSetup: This algorithm generates a public-key for each grid domain administrator. It arbitrarily chooses $g_l \in \mathbb{G}$ for every domain, the initial version number $d_{l,vrn} \in \mathbb{Z}_p^*$ of the *DMK* is secretly kept with the domain administrator DO_l then computes $\hat{e}(g,g)^{d_{l,vrn}}$, and $g^{d_{l,vrn}}$. The system issues *DPK* and *DMK*, for $DPK = \{\forall i \in [1...l]\ \{version, DO_l, g_l, \hat{e}(g,g)^{d_{l,vrn}}, g^{d_{l,vrn}}\}\}$ and $DMK = d_{l,vrn}$. The domain administrator keeps *DMK*, and *DPK* transferred to blockchain.

Stage 3. Key generation:

KeyGen: After users' registration, the system selects a user's L_{id} and *addrUid* as its input. It issues SK_{id} and Υ_{id}. The algorithm arbitrarily picks $r, r_j, r_d \in \mathbb{Z}_p^*$.

$$\begin{aligned} SK_{id} &= \{version, Usk = \{K = g^{\frac{\alpha+r}{\beta}}\ \forall j \in L_{id}: \\ &\quad D_{j1} = g^{r_j}, D_{j2} = H(Uid)^{rd_{l,vrn}} H(attr_j)^{r_j}\} \\ Usk_{D_{id}} &= \{D^1_{D_{id}} = g^{\frac{d_{l,vrn}}{\beta}} \beta \cdot H(Uid)^{\frac{rd_{l,vrn}}{\beta}} \\ &\quad D^2_{D_{id}} = g^{r} \cdot g_l^{r_d}, D^3_{D_{id}} = g^{r_d}\}\}. \end{aligned} \quad (1)$$

Each user chooses an identifier $g_{id} \in \mathbb{G}$, $\Upsilon_{id} = H(Uid)^{\beta^2 d_{l,vrn}} g_{id}^{d_{l,vrn}} g_{id}$, SK_{id} is transferred to user securely. Υ_{id} is publicized and transferred to blockchain. To diminish hefty decryption computations, the Grid Operator delegates SK_{id} to *CSP*/*ES*, then random picks component $z \in \mathbb{Z}_p^*$, and *TFid*, $TFsk'_{D_{id}}$ sent to the user securely.

$$\begin{aligned} TFid &= \{TFsk'_{id} = \{K' = g^{\frac{z(\alpha+r)}{\beta}}\ \forall j \in L_{id}: D'_{j1} = g^{zr_j} \\ &\quad D'_{j2} = H(Uid)^{zrd_{l,vrn}} H(attr_j)^{zr_j}\} \\ TFsk'_{D_{id}} &= \{T^1_{D_{id}} = g^{\frac{zd_{l,vrn}}{\beta}} \cdot H(Uid)^{\frac{zr \cdot d_{l,vrn}}{\beta}} \\ &\quad T^2_{D_{id}} = g^{zr} \cdot g_l^{zr_d} \cdot T^3_{D_{id}} = g^{zr_d}\}\}. \end{aligned} \quad (2)$$

Stage 4. Data encryption:

Encrypt: According to the access structure for the grid domain, an $l \times n$ weighted access matrix $\mathcal{M}$ is constructed, for l signifies the number of attributes. The algorithm selects with randomness vector $\vec{v} = s_1, s_2, s_3, ..., vrn \in \mathbb{Z}_p^*$ for the distribution of secret value s. Then computes $\lambda_i = \vec{v} \cdot \mathcal{M}_i$. $\mathcal{M}_i$ represents the row vector matching the attribute related matrix:

$$\begin{aligned} CT_{prm} &= \{version = vrn, \\ &\quad M, C = m \cdot \hat{e}(g,g)^{\alpha s_2}\ \hat{e}(g,g)^{d_{l,vrn} s_1}: \\ &\quad C_1 = g^{\beta s_1}, C_2 = g^{\beta s_2}, C' = g^{s_1}, C'_{D_{id}} = g^{s_2}, C_{D_{id}} = g_l^{s_2} \\ &\quad V = g^{H'(\hat{e}(g,g)^{\alpha s_2}\ \hat{e}(g,g)^{d_{l,vrn} s_1})} \\ &\quad \forall y \in l: C_i = g^{\lambda_i}, C'_i = H(attr_i)^{\lambda_i}\}. \end{aligned} \quad (3)$$

The Grid Operator forwards CT_{prm} to the *ES* if the CT_{prm} is temporary or lately utilized; else, sends to the *CSP*.

Stage 5. Data decryption:

With this algorithm, *ES*/*CSP* handles hefty and complex computational resources with *TFid*. Firstly, it decrypts the attribute item of the ciphertext:

$$A_1 = \prod_{i\in I}\left(\frac{\hat{e}(C_i, D'_{j2})}{\hat{e}(C'_i, D'_{j1})}\right)^{\omega_i}$$
$$= \prod_{i\in I}\frac{\hat{e}(H(attr_i), g)^{zr_j\lambda_i\omega_i}\ \hat{e}(H(Uid), g)^{zrd_{l,vrn}\lambda_i\omega_i}}{\hat{e}(H(attr_i), g)^{zr_j\lambda_i\omega_i}}$$
$$= \hat{e}\left(H\left(Uid\right), g\right)^{zrd_{l,vrn}s_1}. \quad (4)$$

Then, proceed to decrypt the domain item of ciphertext, for multiple domains key, selects the related domain key, to get A_2, then get A_5 :

$$A_2 = \frac{\hat{e}(T^2_{D_{id}}, C'_{D_{id}})}{\hat{e}(C_{D_{id}}, T^3_{D_{id}})} = \frac{\hat{e}(g, g_l)^{zts_2}\,\hat{e}(g, g)^{zrs_2}}{\hat{e}(g, g_l)^{zts_2}} = \hat{e}(g, g)^{zrs_2}$$

$$A_3 = \hat{e}(C_2, K) = \hat{e}(g^{\beta s_2}, g^{\frac{(\alpha+r)}{\beta}}) = \hat{e}(g, g)^{z(\alpha+r)s_2}$$
$$A_4 = \hat{e}(C_1, T^1_{D_{id}}) = \hat{e}(g, g)^{zd_{l,vrn}s_1}\ \hat{e}(H(Uid), g)^{zrd_{l,vrn}s_1}$$
$$A_5 = \frac{A_3\cdot A_4}{A_1\cdot A_2} = \hat{e}(g, g)^{z\alpha s_2}\ \hat{e}(g, g)^{zd_{l,vrn}s_1} \quad (5)$$

If $V = g^{A_5^{(1/z)}}$, then cloud decryption is a success; if contrary, output $\perp$

$$\frac{C}{A_5^{(1/z)}} = \frac{m\cdot\hat{e}(g, g)^{\alpha s_2}, \hat{e}(g, g)^{d_{l,vrn}s_1}}{(\hat{e}(g, g)^{z\alpha s_2}\,\hat{e}(g, g)^{zd_{l,vrn}s_1})^{1/z}} = m. \quad (6)$$

Here, the decryption is outsourced. The algorithm uses the user's Transformed SK (*TFsk*) saved on the cloud to partly decrypt the ciphertext then convey it to the user. The user solely executes uncomplicated computations to finalize the ultimate decryption.

Stage 6. Malevolent users trace:

Supposed a malevolent user leaks data, that user would be traced and included in the revocable list *RvL*. When the user calls for data, *CSP*/*ES* compares policies and sends Υ_{id} to the *GO*. The *GO* verifies the legitimacy of Υ_{id}. If Υ_{id} is legitimate, the related *addrUid* would be in list *W*. Else; the user possesses no authorization to decrypt the message; hence it is unnecessary to invoke tracing.

Verifyingphase: *GO* checks the authenticity of Υ_{id}. *GO* inputs parameters that include L_{id}, PK_s and Υ_{id}. Then tests for the equation

$\hat{e}(H(addrUid)^{\beta d_{l,vrn}}\, g_{id}^{\beta d_{l,vrn}}, g) = \hat{e}(H(addrUid)g_{id}, g^{\beta d_{l,vrn}})$, if it matches, the user with *addrUid* did leak the private-key.

Stage 7. Data re-encryption:

In the event of data exposure, the malevolent data could be queried utilizing the data header, and the revocation list will then be revised with the *addrUid*, the data would be re-encrypted. In the event of leakage, the user's private-key, and identification, is first uncovered. Then, the domain's ciphertext is re-encrypted. The grid operator chooses $d_{l,vrn+1} \in \mathbb{Z}_p^*$ with randomness, and calculates $g^{d_{l,vrn+1}}, \hat{e}\left(g, g\right)^{d_{l,vrn+1}}$, then makes $DO_l, \hat{e}\left(g, g\right)^{d_{l,vrn+1}}, g^{d_{l,vrn+1}}$ public. It generates the re-encryption key $RenK = g^{\frac{(d_{l,vrn+1}-d_{l,vrn})}{\beta}}$ and sends to the *CSP/ES* to update the ciphertext.

$$CT = \{\ version = vrn,$$
$$M,\ C = m\cdot\ \hat{e}(g, g)^{\alpha s_2}\ \ \hat{e}(g, g)^{d_{l,vrn}s_1}:$$
$$C_1 = g^{\beta s_1}, C_2 = g^{\beta s_2}, C' = g^{s_1}, C'_{D_{id}} = g^{s_2}, C_{D_{id}} = g_l^{s_2}$$
$$\forall y\in l: C_i = g^{\lambda_i}, C'_i = H(attr_i)^{\lambda_i}\}. \quad (7)$$

$$TransCT = \hat{e}(RenK, C_1) = \hat{e}(g, g)^{s_1(d_{l,vrn+1}-d_{l,vrn})}$$
$$C_{vrn+1} = C\cdot TransCT = m\cdot\hat{e}(g, g)^{\alpha s_2}\,\hat{e}(g, g)^{d_{l,vrn+1}s_1}. \quad (8)$$

Stage 8. Update user private-key:

The Grid Operator generates for each user, *TFsk*, and then updated user private-key *Usk* and $Usk_{D_{id}}$

$$TFsk = \begin{cases} d_1 = H(Uid)^{r(d_{l,vrn+1}-d_{l,vrn})} \\ d_2 = g^{\frac{d_{l,vrn}}{\beta}}\cdot H(Uid)^{\frac{r(d_{l,vrn+1}-d_{l,vrn})}{\beta}}. \end{cases} \quad (9)$$

$$Usk = \{K = g^{\frac{(\alpha+r)}{\beta}}, L = g^t\forall j\in L_{id}:$$
$$D_{j1} = g^{r_j}, D_{j2} = H(Uid)^{d_{l,vrn+1}}\,H(attr_j)^{r_j}\}$$
$$Usk_{D_{id}} = D^1_{D_{id}} = (g\cdot H(Uid))^{\frac{d_{l,vrn+1}}{\beta}}$$
$$D^2_{D_{id}} = g^r\cdot g_l^{r_d}, D^3_{D_{id}} = g^{r_d}\}. \quad (10)$$

Stage 9. ES or CSP update private-key saved for user:

After receiving *TransCloudSK*, *ES/CSP* updates the stored private-key of users:

$$TransCloudSK = \begin{cases} d_1 = H(Uid)^{zr(d_{l,vrn+1}-d_{l,vrn})} \\ d_2 = (g\cdot H(Uid))^{\frac{z(d_{l,vrn+1}-d_{l,vrn})}{\beta}} \end{cases}$$

$$D'_{j2} = H(Uid)^{zrd_{l,vrn}}\,H(attr_j)^{zr_j}\,d_l$$

$$= H(Uid)^{zrd_{l,vrn+1}} H(attr_j)^{zr_j}$$

$$T^1_{D_{id}} = g^{\frac{zd_{l,vrn}}{\beta}} \cdot H(Uid)^{\frac{zr \cdot d_{l,vrn}}{\beta}} d_2$$

$$= g^{\frac{zd_{l,vrn}}{\beta}} \cdot H(Uid)^{\frac{zr \cdot d_{l,vrn+1}}{\beta}}. \quad (11)$$

3 Analysis

3.1 Security Analysis

The proffered scheme is built to endure the collusion attack among the revoked and prevailing users. This is achieved by embedding the user's certificate from DGD into each user's private-key.

Definition 1 (Decisional bilinear Diffie-Hellman (DBDH) Assumption): Considering g, g^x, g^y, g^z, R where $x, y, z \in Z_p^*, R \in G_2$ it is hard to determine if $\hat{e}(g,g)^{xyz} = R$.

Theorem 1. *The proffered construction's security is proved on the DBDH assumption and secured under the Ciphertext indistinguishability (IND-CPA) security model. Supposed the DBDH assumption withstands; thus, no polynomial adversary could selectively break the proffered design. If there exists an adversary* $\boldsymbol{A}$ *to break this scheme, a polynomial-time algorithm* $\mathfrak{B}$ *could be built to break the DBDH assumption.*

Proof. The challenger generates parameters $\{\mathbb{G}, \mathbb{G}_T, g, g^a, g^b, q, \hat{e}\}$ then sends to $\mathfrak{B}$ (note: variables a, and b are unknown to $\mathfrak{B}$). $\mathfrak{B}$ aims to successfully deliver $g^{a/b}$ with the following interaction with the adversary $\boldsymbol{A}$.

Init: $\boldsymbol{A}$ discloses the access matrix $(\mathcal{M}^*, \rho^*)$, Domain (D_{id}), and the corresponding version number $d^*_{l,vrn}$ to be challenged.

SetUp: $\mathfrak{B}$ generate MK_s and PK_s, for $MK_s = \{\alpha, \beta\}$, $PK_s = \{\mathbb{G}, \mathbb{G}_T, g, \hat{e}(g,g)^\alpha, g^\beta, H(), H'()\}$. $\mathfrak{B}$ picks with randomness $d_{l,1}, d_{l,2}, \ldots, d_{l,vrn^*} \in \mathbb{Z}_p^*$, then computes $g^{d_{l,vrn}}, \hat{e}(g,g)^{d_{l,vrn}}$

$$DPK = \left\{version, DO_l, g_l, \hat{e}(g,g)^{d_{l,vrn}}, g^{d_{l,vrn}}\right\},$$

$$DPK_{vrn^*} = \left\{version^*, DO_l, g_l, \hat{e}\left(g, g^b\right), g^b\right\}. \quad (12)$$

Phase 1

1. Random Oracle H Query. A forwards queries to the hash function H for user identity, and attributes as follows:

- **Random Oracle query for user identity.** $\mathfrak{B}$ runs a user identity list $LT_{addrUid}$; if Uid_{addr} already in $LT_{addrUid}$, i.e., <Uid, H_i, l_i, c_i> existed, then returns $H(addrUid) = H_i$ to $\boldsymbol{A}$.
- **Random Oracle-query for user attributes**. Algorithm $\mathfrak{B}$ runs user attribute list LT_{attr}, to establish if $attr_j$ already exist in a tuple <$attr_j$, H_{attr_j}, a_j>. Then H_{attr_j} is returned as response to the $attr_j$ query. If not, $a_j \in \mathbb{Z}_p^*, H_{attr_j} = g^{attr_j}$ are chosen with randomness then, a tuple <$attr_j$, H_{attr_j}, a_j> is inserted into the list, and H_{attr_j} returned to $\boldsymbol{A}$.

The merit of algorithm $\boldsymbol{A}$ in the above game is represented as $\Pr\left[\hat{b}' = \hat{b}\right] - \frac{1}{2}$.

2. Type-I Adversarial-Query. (KeyGen Query()): $\mathfrak{B}$ ascertains the user private-key list LT^{Type-I}_{SK}, if private-key of Uid already present in $LT^{Type-II}_{SK}$ it printout $False$ then abolish the program. Else, execute: $KeyGen(MK_s, L_{id}, addrUid)$ algorithm to generate: (Υ_{id}, SK_{id}). The universe of attribute L_{id} satisfies the access matrix $(\mathcal{M}^*, \rho^*)$, The $addrUid$ current version number $version < version^*$

$$SK^I_{id} = Usk^I = \{K = g^{(a+r)/\beta}, L = g^t$$
$$\forall j \in L_{id} : D_j = ((g^a)^{l_i})^{d_{l,vrn}} H^t_{attr_j}\}. \quad (13)$$

$$Usk^I_{D_{id}} = \left\{D^1_{D_{id}} = \left(g \cdot (g^a)^{l_i}\right)^{(d_{l,vrn})/\beta}, D^2_{D_{id}} = g^r \cdot g_l^t\right\}. \quad (14)$$

write $addrUid, SK^I_{id}$ to LT^{Type-I}_{SK} then return SK_{id} to $\boldsymbol{A}$.

3. Type-I Adversarial-Query. (KeyGen Query()). If private-key of $addrUid$ already exist as <$addrUid, SK^I_{id}$> in $LT^{Type-II}_{SK}$ it printout $False$ then, abolish the program. However, if private-key of $addrUid$ appears as $addrUid, SK^{II}_{id}$ in $LT^{Type-II}_{SK}$ then send SK^{II}_{id} to $\boldsymbol{A}$. Else $\mathfrak{B}$ executes $KeyGen(MK_s, L_{id}, addrUid)$ to printout (Υ_{id}, SK_{id}). In case the attributes universe L_{id} fails to fulfill the access matrix $(\mathcal{M}^*, \rho^*)$, the current version number of user Uid becomes $version = version^*$

$$SK^{II}_{id} = Usk^{II} = \{K = g^{(a+r)/\beta}, L = g^t$$
$$\forall j \in L_{id} : D_{j1} = g^{r_j}, D_{j2} = (g^b)^{rl_i} H^t_{attr_j}\}. \quad (15)$$

$$Usk^{II}_{D_{id}} = \left\{\begin{array}{l} D^1_{D_{id}} = \left(g \cdot (g^b)^{l_i}\right)^{1/\beta} \\ D^2_{D_{id}} = g^r \cdot g_l^{r_d}, D^3_{D_{id}} = g^{r_d} \end{array}\right\}, \quad (16)$$

write <$addrUid, SK^{II}_{id}$> to $LT^{Type-II}_{SK}$ then return SK_{id} to $\boldsymbol{A}$.

4. Re-Encrypt Query. If the version-stamp of CT_{vrn} is lower than the current-version, proxy re-encryption is performed. To begin, $\mathfrak{B}$ formulates the re-encryption key $REnc = g^{(d_{l,vrn+1}-d_{l,vrn})/\beta}$ then updates the ciphertext to CT_{vrn^*}. $\mathfrak{B}$ generates and sends the $TransSK$ = { $d_1 = ((g^a)^{l_i})^{r(d_{l,vrn+1}-d_{l,vrn})}$,

$d_2 = g^{(d_{l,vrn+1}-d_{l,vrn})/\beta} \cdot ((g^a)^{l_i})^{r(d_{l,vrn+1}-d_{l,vrn})}\}$ to $\boldsymbol{A}$ to update the private-key.

Challenge: $\boldsymbol{A}$ tenders m_1, m_2 with equal lengths, and $\mathfrak{B}$ selects $\hat{b} \in \{0,1\}$ to encrypt, such that:

$$CT = \{version^*, M, C = m \cdot \hat{e}(g,g)^{\alpha s_2}\ \hat{e}(g,g)^{d_{l,vrn}s_1},$$
$$C_1 = g^{\beta s_1}, C_2 = g^{\beta s_2}, C' = g^{s_1}, C'_{D_{id}} = g^{s_2},$$
$$C_{D_{id}} = g_l^{s_2},\ V = g^{H'(\hat{e}(g,g)^{\alpha s_2}\hat{e}(g,g)^{d_{l,vrn}s_1})}$$
$$\forall y \in l; C_i = g^{\lambda_i}, C'_i = H(attr_i)^{\lambda_i}\}. \quad (17)$$

Phase 2: Emulates Phase1

Guess: Ultimately, $\mathfrak{B}$ disregards $\boldsymbol{A}$'s result and arbitrarily picks a $Usk_{I,i}$ (as *Type − I query*'s reply) and a $Usk_{II,i}$ (as *Type − II query*'s reply). Hypothetically, $\boldsymbol{A}$ could decrypt the challenge ciphertext $CT_{\hat{b}}$ if and only if it had $T^1_{D_{id}}$ of $SK_{II,i}$ matching D_j of $SK_{I,i} \cdot <Uid'_j, H'_i, l'_i, coin'_i = 1>$ is denoted to correspond to $SK_{I,i}$ and $<Uid'_j, H'_i, l'_i, coin'_i = 0>$ to correspond to $SK_{II,i}$. To formulate the aforementioned theoretical evidence, $(g^b)^{l'_i} = (H_i)^{rd_{l,vrn}}$, meaning, $g^{b \cdot l'_i r \cdot d_{l,vrn}} = g^{a \cdot l_i r \cdot d_{l,vrn}}$. Therefore, the output $g^{a/b} = g^{l'_i/l_i}$ represents the answer of $\mathfrak{B}$.

Should $\mathfrak{B}$ not respond $\perp$ in *phase*1, it means $\boldsymbol{A}$'s opinion is indistinguishable from its opinion in the real-world attack. Then, for simulation, it computes $\mathfrak{B}$'s possibility of success to complete the proof. Presuppose $\boldsymbol{A}$ makes q_I *Type − Iquery* and q_{II} *Type − IIquery* Then, the probability that $\mathfrak{B}$'s response in *phase*1 is not $\perp$ is $\Upsilon^{q_I} \cdot (1-\Upsilon)^{q_{II}}$ (in simple terms, it is presupposed $q_I = q_{II}$). This value is maximized when $\Upsilon = 1/2$. Consequently, $\mathfrak{B}$'s probability of success during the simulation is $1/2(q_I + q_{II})$. $\mathfrak{B}$'s probability of selecting the right $SK_{I,i}$ and $SK_{II,i}$ is $1/q_I \cdot q_{II}$, indicating that the advantage of $\mathfrak{B}$ is $\varepsilon/2^{(q_I + q_{II})} \cdot q_I \cdot q_{II}$ at its highest.

3.2 Security Properties Comparisons

This section compares the security characteristics of the proffered scheme with other relevant designs. For this comparison, recently introduced attribute-based encryption data exchange frameworks [7-10] have been earmarked as a bench-mark. Precise outcomes are epitomized in Table 2.

Table 2. Comparison of security properties

Scheme	Indirect user revocation	Direct user revocation	Policy hiding	Policy comparison	Traceability	IND-CPA based security
[7]	Yes	No	Yes	AND gate	No	Yes
[8]	Yes	No	No	LSSS	No	Yes
[9]	No	Yes	No	LSSS	No	Yes
[10]	No	Yes	No	Access Tree	No	Yes
Proffered scheme	No	Yes	No	Access Tree	Yes	Yes

4 Simulated Experimentation

This section provides performance evaluation. Linear pairing and exponentiation operations are performed 100 times utilizing a machine equipped with a Windows 10 operating system (64-bit, Intel Core i5 CPU, @2.4 GHz, and 16 G RAM) for a high level of simulation accuracy.

4.1 Communication Overhead

The expenses of communication are appraised with a comparison between the proffered design and schemes [7-10] concerning the trajectories of encryption/decryption duration and public/private key generation duration against the number of attributes. Figure 3. exhibits the statistics. Figure 3(a) reveals that where parameter $r = 1$, the encryption duration for the five constructions grows linearly as the number of attributes grows from 1 to 10. Amongst them, [7] rises the quickest, claiming most encryption duration. The remaining four constructions rise nearly simultaneously, but the proffered scheme enjoys the least encryption duration. Figure 3(d) describes the trajectories of the encryption duration for the five constructions as the parameter $r = 3$.

Figure 3(d) is similar to Figure 3(a). The uniqueness is the encryption duration of the proffered scheme being somewhat elevated than [8-10]. Next, the trajectories of the decryption duration for the five constructions are plotted. In Figure 3(b), where $r = 1$, the duration for decryption for [7, 9-10] progresses moderately as the number of attributes grows from 1 to 10, although [8] and the proffered design stay relatively constant. Figure 3(e) portrays the trajectories of the decryption duration for the five constructions, as parameter $r = 3$. Figure 3(e) is similar to Figure 3(b). The uniqueness is that the decryption rate of the proffered design is slightly elevated than [8].

Lastly, the public/private key generation duration trajectories are plotted for the five constructions. Figure 3(c) presents the public-key generation duration trajectory. As displayed in Figure 3(c), while the number of attributes grows from 1 to 10, the public-key generation duration for [7-10] rises linearly. Specifically, [7, 9] rises the quickest; then [8, 10]. Further, the public-key generation duration of the proffered scheme stays virtually unchanged. Figure 3(f) reveals the private-key generation duration trajectory. The private-key generation duration for the five systems grows linearly by the number of attributes. Specifically, [7] rises the quickest, next is [8], and the other three systems progress the slowest maintaining relatively the same pace.

4.2 Storage Overhead

The appraisal for storage overheads compares the proffered design and schemes [7-10] regarding the trajectories of public/private key lengths against the number of attributes and the ciphertext length against the number of users within the revocation universe. Figure 4 demonstrates the details. As exhibited in Figure 4(a), as the number of attributes rises from 1 to 10, the storage overhead of the public-key in [7-10] rises linearly, but the proffered scheme retains constant. Figure 4(b) reveals that growth in the number of attributes linearly gives rise to the private-keys' storage overhead in

all five schemes. However, the proffered scheme enjoys the minimum rise.

In addition, Figures 4(c) and Figure 4(d) report that while the number of attributes rises, the ciphertext length of the proffered design is nearly equivalent to [8-10] although significantly less than [7].

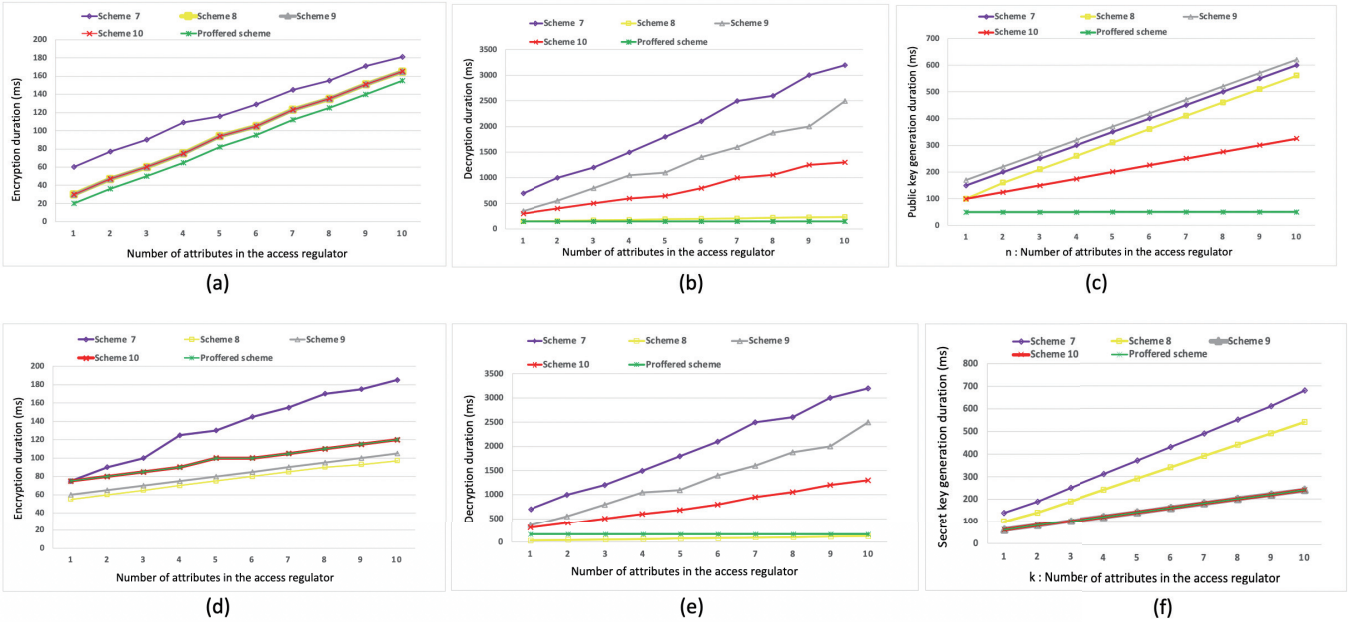

Figure 3: Duration overheads for encryption/decryption, and public/secret key generation duration. r represents the number of attributes in the access regulator. 3(a) Encryption duration where $r = 1$. 3(b) Decryption duration where $r = 1$. 3(c) Public key generation duration. 3(d) Encryption duration where $r = 3$. 3(e) Decryption duration where $r = 3$. 3(f) Secret key generation duration.

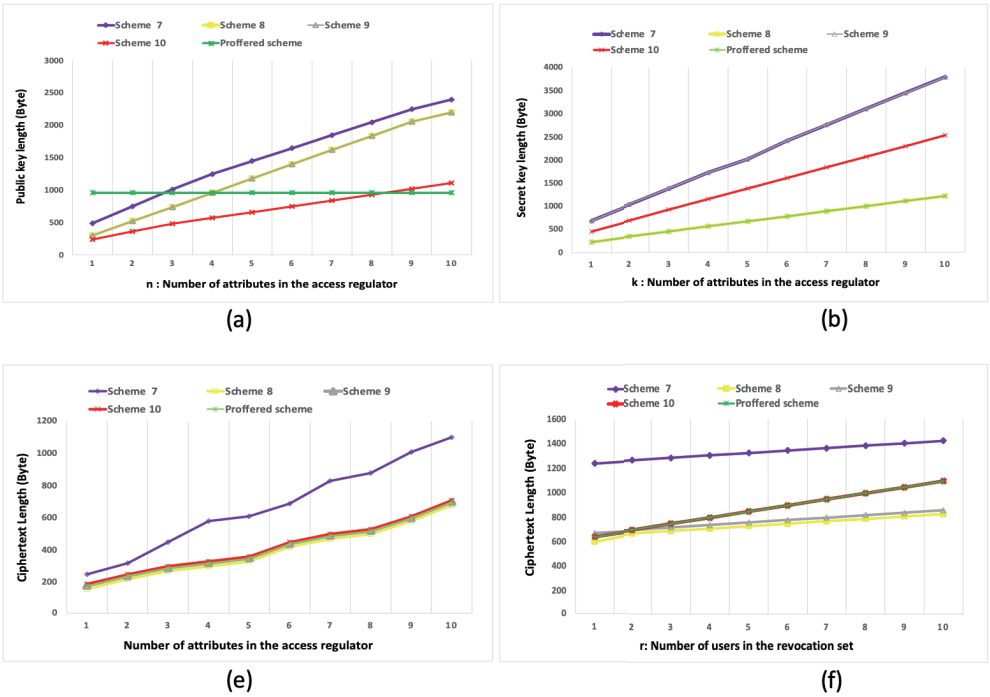

Figure 4: Storage overheads of public/private key and ciphertext against the number of attributes. 4(a) Storage overheads for public-key. 4(b) Storage overheads for private-key. 4(c) Storage overheads for ciphertext. 4(d) Storage overheads for ciphertext against the number of users in the revocation universe.

4.3 Throughput of Blockchain

The private chain platform: Ethereum is utilized for the blockchain in the construction procedure. The simulation machine operates on Ubuntu 18.04.6 LTS (Bionic Beaver) with Intel Core i7-8565u quad-core 1.80 GHz processor and 8GB RAM. The reputed blockchain throughput represents the number of transactions (accesses) handled per second. This experiment simulates two circumstances of access to the blockchain in the smart grid. On the one hand, many smart grid applications *read* data from the blockchain; on the other hand, numerous smart grid applications *write* transaction data to the blockchain. The amount of transactions per second (tps) utilized in the simulation is from 100 to 1000, and the test outcome is the average number of which 1 represents 10 runs. Figure 5 displays the performance of throughput along with delay for blockchain *READ* and *WRITE* transactions. During *READ* operations, the proffered design provides high throughput and low latency. Moreover, the proffered design works well within Ethereum private chain, while the throughput and delay operations are satisfactory when executing *WRITE* operations in blockchain.

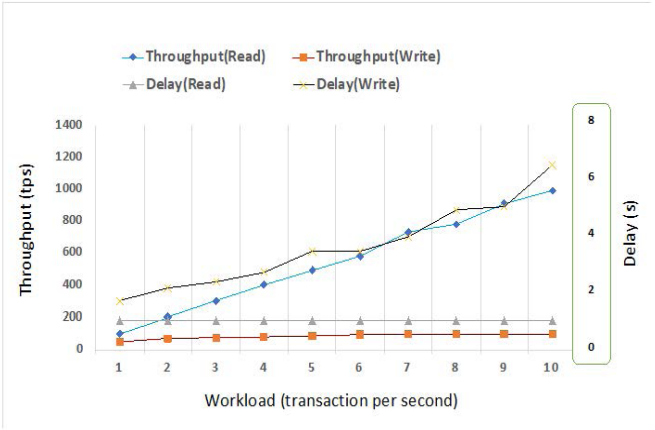

Figure 5. Blockchain operations: READ and WRITE throughput and delay

5 Conclusion

The cloud secure storage and data exchange for smart grid data has been challenging for several systems and users. This study offered a refined ciphertext-policy secured access regulation built on blockchain engineering that upholds user identification-based tracing and revocation. A distinctive identification parameter is generated and implanted during the private-key formation stage to detect and thwart collusion attacks with this design. Any user with attributes fulfilling the access policy and are not on the revocable list can successfully compute the decryption key. Malevolent clients would then be traced per the tracing record and revoked directly.

In future research, we consider developing more generic constructions which do not require the DBDH assumption. It may also be interesting to design a blockchain data exchange protocol for public-healthcare systems with decryption-test that support flexible authorization.

Acknowledgment

This work was supported in part by the Open Fund of Advanced Cryptography and System Security Key Laboratory of Sichuan Province under Grant SKLACSS-202102, in part by the Intelligent Terminal Key Laboratory of Sichuan Province under Grant SCITLAB-1019, in part by the Sichuan Science and Technology under Grant 2021JDRC0072 and 2021YFG0164, and in part by the Open Foundation of State Key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) under Grant SKLNST-2019-2-13.

References

[1] T.-Y. Wu, L. Yang, Z. Lee, S.-C. Chu, S. Kumari, S. Kumar, A Provably Secure Three-Factor Authentication Protocol for Wireless Sensor Networks, *Wireless Communications and Mobile Computing*, Vol. 2021, Article No. 5537018, April, 2021.

[2] M. Jaradat, M. Jarrah, A. Bousselham, Y. Jararweh, M. Al-Ayyoub, The Internet of Energy: Smart Sensor Networks and Big Data Management for Smart Grid, *Procedia Computer Science*, Vol. 56, pp. 592-597, 2015.

[3] C. Tu, X. He, Z. Shuai, F. Jiang, Big Data Issues in Smart Grid–A Review, *Renewable and Sustainable Energy Reviews*, Vol. 79, pp. 1099-1107, November, 2017.

[4] C.-M. Chen, X. Deng, S. Kumar, S. Kumari, S. K. Islam, Blockchain-based medical data sharing schedule guaranteeing security of individual entities, *Journal of Ambient Intelligence and Humanized Computing*, pp. 1-10, August, 2021.

[5] H. Xiong, Y. Hou, X. Huang, Y. Zhao, C.-M. Chen, Heterogeneous Signcryption Scheme from IBC to PKI With Equality Test for WBANs, *IEEE Systems Journal*, pp. 1-10, February, 2021.

[6] X. Huang, H. Xiong, J. Chen, M. Yang, Efficient Revocable Storage Attribute-based Encryption with Arithmetic Span Programs in Cloud-assisted Internet of Things, *IEEE Transactions on Cloud Computing*, November, 2021.

[7] J. Li, Y. Shi, Y. Zhang, Searchable Ciphertext-Policy Attribute-Based Encryption with Revocation in Cloud Storage, *International Journal of Communication Systems*, Vol. 30, No. 1, Article No. e2942, January, 2017.

[8] R. Ostrovsky, A. Sahai, B. Waters, Attribute-Based Encryption with Non-Monotonic Access Structures, *Proceedings of the 14th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, 2007, pp. 195-203.

[9] C. Bai, Y. Zhang, H. Ma, Z. Liu, Expressive Ciphertext-Policy Attribute-Based Encryption with Direct User Revocation, *International Journal of Embedded Systems,* Vol. 9, No. 6, pp. 495-504, November, 2017.

[10] X.-X. Yan, M. Hui, Ciphertext Policy Attribute-Based Encryption Scheme Supporting Direct Revocation,

Journal on Communications, Vol. 37, No. 5, pp. 44-50, May, 2016.
[11] H. Xiong, L. Wang, Z. Zhou, Z. Zhao, X. Huang, S. Kumari, Burn after Reading: Adaptively Secure Puncturable Identity-Based Proxy Re-Encryption Scheme for Securing Group Message, *IEEE Internet of Things Journal*, pp. 1-12, November, 2021.
[12] H. Xiong, Z. Zhou, L. Wang, Z. Zhao, X. Huang, H. Zhang, An Anonymous Authentication Protocol with Delegation and Revocation for Content Delivery Networks, *IEEE Systems Journal*, pp. 1-12, October, 2021.
[13] C. M. Chen, Y. Huang, K. H. Wang, S. Kumari, M. E. Wu, A Secure Authenticated and Key Exchange Scheme for Fog Computing, *Enterprise Information Systems*, Vol. 15, No. 9, pp. 1200-1215, 2021.
[14] P. S. K. Oberko, V.-H. K. S. Obeng, H. Xiong, A Survey on Multi-Authority and Decentralized Attribute-Based Encryption, *Journal of Ambient Intelligence and Humanized Computing*, Vol. 13, No. 1, pp. 515-633, January, 2022.
[15] H. Xiong, J. Chen, Q. Mei, Y. Zhao, Conditional Privacy-Preserving Authentication Protocol with Dynamic Membership Updating for VANETs, *IEEE Transactions on Dependable and Secure Computing*, No. 1, pp. 1-1, December, 2020.
[16] J. Wang, W. Chen, Y. Ren, O. Alfarraj, L. Wang, Blockchain Based Data Storage Mechanism in Cyber Physical System, *Journal of Internet Technology*, Vol. 21, No. 6, pp. 1681-1689, November, 2020.
[17] P. S. K. Oberko, V.-H. K. S. Obeng, H. Xiong, S. Kumari, A survey on Attribute-Based Signatures, *Journal of Systems Architecture*, Vol. 124, Article No. 102396, March, 2022.
[18] R. Chaudhary, G. S. Aujla, S. Garg, N. Kumar, J. J. Rodrigues, SDN-Enabled Multi-Attribute-Based Secure Communication for Smart Grid in IIoT Environment, *IEEE Transactions on Industrial Informatics*, Vol. 14, No. 6, pp. 2629- 2640, June, 2018.
[19] C.-M. Chen, X. Deng, W. Gan, J. Chen, S. K. Islam, A secure blockchain-based group key agreement protocol for IoT, *The Journal of Supercomputing*, Vol. 77, No. 8, pp. 9046- 9068, August, 2021.
[20] S. Shamshad, Minahil, K. Mahmood, S. Kumari, C.-M. Chen, A secure blockchain-based e-health records storage and sharing scheme, *Journal of Information Security and Applications*, Vol. 55, Article No. 102590, December, 2020.
[21] J. Davis, *The crypto-currency*, The New Yorker 87, October, 2011.
[22] E. Mengelkamp, B. Notheisen, C. Beer, D. Dauer, C. Weinhardt, A Blockchain-Based Smart Grid: Towards Sustainable Local Energy Markets, *Computer Science-Research and Development*, Vol. 33, No. 1-2, pp. 207-214, February, 2018.
[23] J. Gao, K. O. Asamoah, E. B. Sifah, A. Smahi, Q. Xia, H. Xia, X. Zhang, G. Dong, Gridmonitoring: Secured sovereign blockchain based monitoring on smart grid, *IEEE Access*, Vol. 6, pp. 9917-9925, February, 2018.
[24] A. A. G. Agung, R. Handayani, Blockchain for smart grid, *Journal of King Saud University-Computer and Information Sciences*, Vol. 34, No. 3, pp. 666-675, March, 2022.
[25] K. Gai, Y. Wu, L. Zhu, M. Qiu, M. Shen, Privacy-preserving energy trading using consortium blockchain in smart grid, *IEEE Transactions on Industrial Informatics*, Vol. 15, No. 6, pp. 3548-3558, June, 2019.
[26] W. Wang, H. Huang, L. Zhang, C. Su, Secure and efficient mutual authentication protocol for smart grid under blockchain, *Peer-to-Peer Networking and Applications*, Vol. 14, No. 5, pp. 2681-2693, September, 2021.

Biographies

Prince Silas Kwesi Oberko received the Ph.D. degree in Software Engineering from the University of Electronic Science and Technology China (UESTC). He received the Master of Information Technology from the Open University of Malaysia (OUM), in 2013. His research interest includes data security and cryptography..

Tianang Yao is currently pursuing his M.S. degree from the School of Information and Software Engineering, University of Electronic Science and Technology of China. He received his B.S. degree from University of Electronic Science and Technology of China in 2019. His research interests include public-key cryptography and network security.

Hu Xiong is currently a professor in the School of Information and Software Engineering, UESTC. He received the Ph.D. degree in the School of Computer Science and Engineering from the University of Electronic Science and Technology of China (UESTC) in 2009. His research interests include cryptographic protocols and network security.

Saru Kumari is currently an Assistant Professor with the Department of Mathematics, Chaudhary Charan Singh University (CCSU), Meerut, Uttar Pradesh, India. She received her Ph.D. degree in Mathematics in 2012 from CCSU, Meerut, UP, India. Her current research interests include information security and applied cryptography.

Sachin Kumar received the Ph.D. degree in computer science from Chaudhary Charan Singh University, Meerut, India, in 2007. He is working as a Professor of computer science and engineering with Ajay Kumar Garg Engineering College, Ghaziabad, India. His research interests include computer networks and applied cryptography.