A Framework for Blockchain Supported Carbon Emission Data Sharing with Flexible Access Control and Enhanced Data Privacy

Shichao Huang[1,a], Zehao Wang[1], Jiayan Wang[1], Youfei Lu[1], Xueqing Liang[1], Luhao Liu[1], Yi Zhou[1], Zetao Chen[1], Rui Zhang[2,3,4], Haoxin Wang[2,3,4,*]

[1]Guangzhou Power Supply Bureau, Guangdong Power Grid Co. Ltd., 510000, Guangzhou, China
[2]Institute of Information Engineering, Chinese Academy of Sciences, 100085, Beijing, China
[3]School of Cyberspace Security, University of Chinese Academy of Sciences, 100049, Beijing, China
[4]Key Laboratory of Cyberspace Security Defense, 100085, Beijing, China

[a]huangsc@guangzhou.csg.cn
*Corresponding author's e-mail: wanghaoxin@iie.ac.cn

Abstract. In this paper, the problem of carbon emission data sharing using blockchains is studied. To avoid data leakage, the data is usually encrypted before uploading to the blockchain. To enable flexible searching, a symmetric key searchable encryption is applied, where the blockchain can transform any index ciphertext into a searchable one. We give a rigorous system model and related security definitions. We also analyze and evaluate the performance of our scheme. The results of our experiments show our scheme is secure and efficient.

Keywords: blockchain, Carbon Emission Data, access control, data privacy.

1 Introduction

Carbon emission related data from the grid has become a crucial data to quantify carbon emission [1]. However, these data are closely related to an enterprise's actual production, hence it is essential to ensure boarder of the how information is used and shared.

On the other hand, blockchain is decentralized ledger, and widely used for data sharing [2], but the naïve uploading data on blockchain will certainly leak information. A common method is to encrypt the data before storing it on the blockchain. But then, data usability is a problem since traditional encryption prevents data usage [3]. Encrypted data loses its original semantic characteristics, making it difficult for users to perform data retrieval and meet fine-grained access control requirements.

To solve these problems, we consider to use searchable encryption and attribute-based encryption. Searchable encryption [4] is a special encryption technology that allows users to perform search operations on encrypted data without decrypting it. The known schemes can support different types of search functions. For the large volume

W. He et al. (eds.), *Proceedings of the 2024 10th International Conference on Advances in Energy Resources and Environment Engineering (ICAESEE 2024)*, Atlantis Highlights in Engineering 36,
https://doi.org/10.2991/978-94-6463-708-3_44

data, numerous users, and varying retrieval permissions, symmetric searchable encryption scheme [5,6] is desirable, which allows control over the searchable range for different types of users, while supporting both single-keyword and multi-keyword searches. Attribute-based encryption is an advanced encryption technology that allows access control of data based on user attributes or attribute sets. Compared to traditional encryption technologies, attribute-based encryption provides finer-grained access control, enabling encryption access permissions to be customized based on the user's identity, role, or other attributes. Attribute-based encryption technology is divided into Key Policy Attribute-Based Encryption (KP-ABE) and Ciphertext Policy Attribute-Based Encryption (CP-ABE). CP-ABE, by embedding the access structure in the ciphertext and the attribute set in the key, ensures that only users meeting specific attribute conditions can decrypt the data, thus providing flexible data access control. Here we choose CP-ABE, achieving mandatory access control and addressing the timely revocation of user permissions, ensuring both the security and flexibility of data sharing.

Our results. We introduce a framework for sharing power grid data based on attribute-based searchable encryption, which enjoys retrievability and fine-grained access control for encrypted power grid data. In particular, by subdividing user attributes into attribute names and attribute values, a linear secret-sharing mechanism is used to partially conceal access policies. In addition, the scheme constructs a binary tree structure to distribute attribute group keys, optimizing key management and implementing dynamic revocation of user attributes. Furthermore, the scheme integrates multi-keyword search functionality, significantly improving the efficiency of encrypted data retrieval, making data access more flexible and efficient.

2 Preliminary

2.1 KEK Binary Tree

A KEK binary tree is special binary tree, where each node v_j in the tree stores an independent and random symmetric key denoted by KEK_j. The path from the leaf node to the root node is defined as the path key (cf. Figure 1).

2.2 Attributed Based Encryption

Sahai and Waters [7] introduced the concept of attribute-based encryption (ABE), which features fine-grained access control on data. Later more efficient ABE scheme was proposed. To date the most efficient one is [8,9,10].

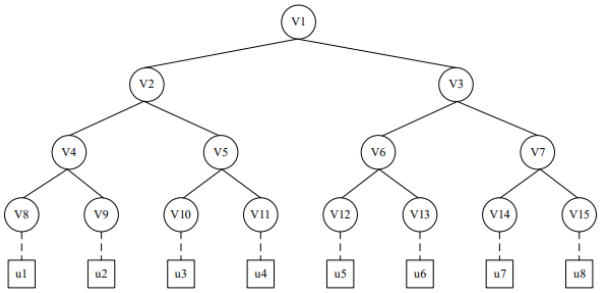


Fig. 1. KEM binary tree

3 The Model

3.1 System Model

3.1.1 Network.

We assume the users are connected via secure channels, which mimics the current Internet communication with TLS or IPSec. The network is synchronous in the sense that data will arrive the target user within a fixed time epoch.

3.1.2 Players and Workflow.

The system model consists of four entities: power grid users, data users, the blockchain, and the key generation center (KGC). The workflow is depicted in Figure 2.

The scheme consists of seven stages: system initialization, key generation, data encryption, re-encryption, data retrieval, data decryption, and ciphertext update, involving the following ten polynomial-time algorithms:

1. $\mathrm{Setup}(\lambda) \rightarrow (\mathrm{PK}, \mathrm{masterKey})$, run by KGC, takes security parameter λ as input, and outputs a public parameter PK and a master key $\mathrm{masterKey}$.
2. $\mathrm{ABEKeyGen}(\mathrm{PK}, \mathrm{masterKey}, \mathrm{S}) \rightarrow \mathrm{sk}$, run by KGC, takes PK, $\mathrm{masterKey}$ and data user u a set of attributes S as input, and outputs a secret key sk for data user u.
3. $\mathrm{KEKeyGen}(\mathrm{u_t}) \rightarrow \mathrm{PK_t}$, run by the blockchain, takes data user's ID $\mathrm{u_t}$ as input, and outputs a path secret key $\mathrm{PK_t}$.
4. $\mathrm{Encrypt}(\mathrm{PK}, \mathrm{W}, \mathrm{m}, \mathcal{F}) \rightarrow (\mathrm{CT}, \mathrm{I_w})$, run by power grid user, takes public parameter PK, keyword set W, data m, and access policy $\mathcal{F}$, and outputs ciphertext CT and index ciphertext $\mathrm{I_w}$ for the corresponding keywords.
5. $\mathrm{ReEncrypt}(\mathrm{CT}, \mathrm{G}) \rightarrow (\mathrm{CT}')$, run by the blockchain, takes ciphertext CT and a set of attributes G as input, and outputs re-encrypted ciphertext CT'.
6. $\mathrm{Trapdoor}(\mathrm{PK}, \mathrm{sk}, \mathrm{W}') \rightarrow (\mathrm{T_q})$, run by power grid user, takes public parameter PK, secret keys for his attributes sk and set of keywords W' as input, and outputssearch trapdoor $\mathrm{T_q}$.

7. $\mathrm{Search}(\mathrm{PK}, \mathrm{T_q}, \mathrm{I_W}) \rightarrow (\mathrm{CT}', \mathrm{Hdr})$, run by the blockchain, takes public parameter PK, search trapdoor $\mathrm{T_q}$, index ciphertext $\mathrm{I_W}$ for the corresponding keywords as input, and outputs re-encryption ciphertext CT' and ciphertext header Hdr or $\perp$.
8. $\mathrm{Transform}(\mathrm{PK}, \mathrm{sk}', \mathrm{K_\tau}) \rightarrow (\mathrm{K_I})$, run by data user, takes public parameterPK, transform key for his attributes sk' and transformed key $\mathrm{K_\tau}$ as input, and outputs $(\mathrm{CT}', \mathrm{Hdr})$ or $\perp$.
9. $\mathrm{PartDecrypt}(\mathrm{PK}, \mathrm{K_I}, \mathrm{CT}') \rightarrow \mathrm{PCT}$, run by the blockchain, takes public parameter PK, transform key $\mathrm{K_I}$, re-encryption ciphertext CT' as input, and outputspartial decrypted ciphertext PCT.
10. $\mathrm{Decrypt}(\mathrm{sk}', \mathrm{PCT}, \mathrm{K_\tau}) \rightarrow \mathrm{m}$, run by data user, takes the transformed secret keys for his attributes sk', partial decrypted ciphertext PCT , transform key $\mathrm{K_\tau}$ as input, and outputs power grid data m.

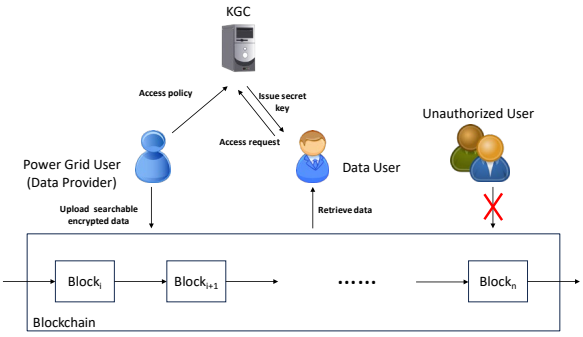


Fig. 2. System Workflow

3.2 Security Definitions

We consider data privacy and search correctness.

Definition 1 (Confidentiality) We require that a set of users other than the enabled user sets with the prespecified attributes should not have any information of the target ciphertext.

Definition 2 (Search Correctness) We require that a user with correct search token, can correctly find a data using search trapdoors.

4 The Proposed Scheme

The scheme consists of seven stages: system initialization, key generation, data encryption, re-encryption, data retrieval, data decryption, and ciphertext update, involving ten polynomial-time algorithms. The following scheme is a natural combination of an attribute-based encryption scheme and a searchable encryption scheme of multiple users and multiple keywords.

- $\mathbf{Setup}(\lambda) \rightarrow (\mathbf{PK}, \mathbf{masterKey})$

The key generation center initializes with a security parameter λ. It selects two cyclic groups of prime order$p$, $G_1$and G_2 with g as the generator of $g_1 \in G_1$. Given a bilinear pairing $e: G_1 \times G_1 \to G_2$ and two hash functions $H_1: \{0,1\}^* \to G_1$ and $H_2: \{0,1\}^* \to Z_{p^*}$, compute each attribute $T_i = H_1(\text{att(i)})Ti = H_1(\text{att(I)})$ in the attribute set. The key generation center randomly selects α, β, and μ from $Z_p{}^*$, and computes $e(g,g)^{\alpha}$, g^{β} and g^{μ}. The center saves $\text{masterKey} = \{\alpha, \beta\}$ and publishes public parameters $PK = \{G_1, G_2, p, e, e(g,g)^{\alpha}, g^{\beta}, g^{\mu}, H_1, H_2, T_i\}$.

- $\textbf{ABEKeyGen}(\textbf{PK}, \textbf{masterKey}, \textbf{S}) \to \textbf{sk}$

1. The key generation center selects a random number $x \in Z_p^*$, computes $K = g^{x\beta}$, $K_1 = g^x$, $K_2 = \beta/x$ and $K_3 = g^{\alpha}$. Additionally, the center randomly selects $y \in Z_p^*$ and computes $K_4 = (g^{\alpha} g^{\mu y})^{\beta/x}$, $K_5 = g^{xy/\beta}$, and $K_i = (T_i^{y \cdot s_i})^{x/\beta}$. The center sends the attribute key $k = \{S, K, K_1, K_2, K_3, K_4, K_5, K_I\}$ to the data user.
2. The center sends each attribute name i and its corresponding attribute group G_I to the blockchain.

- $\textbf{KEKeyGen}(\mathbf{u_t}) \to \mathbf{PK_t}$:

The blockchain defines a KEK binary tree for all users, assigning each user a unique leaf node. Each node v_j in the tree stores an independent and random symmetric key denoted by KEK_j. The path from the leaf node to the root node is defined as the path key. The blockchain assigns each user a path key PK_t from the KEK tree.

- $\textbf{Encrypt}(\textbf{PK}, \textbf{W}, \textbf{m}, \textbf{F}) \to (\textbf{CT}, \mathbf{I_w})$:

1. The power grid user randomly selects a symmetric key k and encrypts the power grid data mm to obtain the ciphertext $C = \text{Enc}_k(m)$. The ciphertext C is uploaded to the blockchain for storage, and the system returns the storage address R.
   The power grid user selects a multi-keyword set W from the data, which contains d keywords. For each $w_j \in W$, randomly select ϕ and ϕ_1 from Z_p^*, and compute $C_{w_j} = g^{(H_2(wj)\beta\phi)}$, $CK_{W_1} = g^{\phi_1}$, $CK_{W_2} = g^{\phi_1 \beta}$, $CK_{W_3} = g^{\mu\phi}$. The multi-keyword index ciphertext is $I_w = \{Cwj, CK_{W_1}, CK_{W_2}, CK_{W3}\}$.
2. The power grid user encrypts the symmetric key k and file address R with the access policy $F = (M, \rho, T)$, where MM is an $l \times n$ matrix, ρ is a function mapping the matrix M to attribute names, and $T = \{t\rho(i)\}i \in [1,l]$ is the attribute values associated with (M, ρ). The user randomly selects a vector $v = (s, v_2, \ldots, v_n)$ with $v_2, \ldots, v_n \in Z_p^*$ to share the secret value $s \in Z_p^*$. For each $i \in [1,l]$, compute $\lambda_i = M_i \cdot v$, where M_I represents the i-th row of the matrix. Additionally, for each $i \in [1,l]$, randomly select $r_i \in Z_p^*$, and compute $C_k = ke(g,g)^{\alpha s}$, $C_R = Re(g,g)^{\alpha s}$, $C_\eta = g^s, \{B_i = g^{\mu\lambda_I}, C_i = T_{\rho(i)}^{-r_i \cdot t_{\rho(I)}}, D_i = g^{r_i}\}_{i \in [1,l]}$. The ciphertext is $CT = \{\bar{\mathcal{F}}, C_k, C_R, C_\eta, \{B_I, C_I, D_i\}_{i \in [1,l]}$, where $\bar{\mathcal{F}} = (M, \rho)$ is the remaining access policy without attribute values.
3. The power grid user uploads (CT, I_w) to the blockchain for storage.

- $\textbf{ReEncrypt}(\textbf{CT}, \textbf{G}) \to (\textbf{CT}')$:

1. For each $G_i \in G$, the blockchain randomly selects $\theta_i \in Z_p^*$, where G represents the attribute groups corresponding to all attribute names in the ciphertext access policy. The re-encrypted ciphertext is $CT' = \{\bar{\mathcal{F}}, C_k, C_R, C_\eta, \{B_i = g^{\mu\lambda_I}, C_i = \left(T_{\rho(i)}^{-r_i \cdot t_{\rho(i)}}\right)^{\theta_I}, D_i = g^{r_i}\}_{i\in[1,l]}\}$.
2. For each $G_i \in G$, the blockchain selects a root node in the KEK tree that covers all users in G_i. This is denoted by $KEK(G_i)$. This set covers all users in G_i, and users not in G_i cannot access any information in $KEK(G_i)$. The blockchain uses the symmetric keys from the $KEK(G_i)$ set to encrypt the attribute group key θ_i, resulting in the ciphertext header information Hdr, where $Hdr = \left(i \in [1,l]: \{Enc_k(\theta_i)\}_{K\in KEK(G_i)}\right)$.
3. The blockchain saves (CT', I_W, Hdr). After storing the related data, the deployed smart contract generates the hash value, timestamp, and transaction initiator information of (CT', I_W, Hdr) and uploads it to the blockchain. After consensus is reached among the blockchain nodes, the transaction ID is returned.

- $\mathbf{Trapdoor(PK, sk, W')} \rightarrow \mathbf{(T_q)}$:

1. The data user inputs the attribute key sk and the multi-keyword set W' to generate the search trapdoor. For each $w_i \in W'$, the user randomly selects $q \in Z_p^*$ and calculates $Y_1 = K^q, Y_2 = K_1{}^q, Y_3 = \sum_{l'=1}^{d'} g^{H_2(w_{l'})\beta}, Y_4 = g^{\mu}$. The search trapdoor is $T_q = \{Y_1, Y_2, Y_3, Y_4\}$.
2. The data user uploads the search trapdoor T_q to the blockchain, invoking the search smart contract deployed on the blockchain.

- $\mathbf{Search(PK, T_q, I_W)} \rightarrow \mathbf{(CT', Hdr)}$:

The blockchain matches the search trapdoor T_q with the multi-keyword index ciphertext I_W, verifying whether the following equation holds: $e(Y_2, C_{KW_2})e\left(\prod_{j=1}^{d'} C_{W_j}, Y_4\right) = e(Y_1, C_{KW_1})e(Y_3, C_{KW_3})$. If the verification passes, the blockchain returns (CT', Hdr) to the data user, and the smart contract updates the search status to $Sch = 1$. Otherwise, the search fails, and the blockchain returns $\perp$, with the smart contract updating the search status to $Sch = 0$. The smart contract records and uploads related transaction information such as search status, transaction initiator, timestamp, and other details to the blockchain, and after consensus is reached among the blockchain nodes, returns the transaction ID.

- $\mathbf{Transform(PK, sk', K_\tau)} \rightarrow \mathbf{(K_I)}$:

1. When the data user receives (CT', Hdr), they obtain all the attribute group keys θ_I corresponding to the attributes they own from Hdr. Then, the data user uses the attribute group keys θ_I to update the attribute key to get sk', where $sk' = \left\{S, K, K_1, K_2, K_4, K_5, K_i' = \left[(T_i^{y\cdot s_i})^{\frac{x}{\beta}}\right]^{\theta_i} : \forall i \in I_S\right\}$.

2. The data user randomly selects $\tau \in Z_p^*$, calculates $K_\tau = g^\tau, K_6 = K_4 \cdot K_\tau = (g^\alpha g^{\mu y})^{\frac{x}{\beta}} \cdot g^\tau$. The data user sends the transformation key K_I to the blockchain, where $K_I = \{K_5 = (g^y)^{\frac{x}{\beta}}, K_6 = (g^\alpha g^{\mu y})^{\frac{x}{\beta}}, K_i' = \left[(T_i^{y \cdot s_i})^{\frac{x}{\beta}}\right]^{\theta_i} : \forall i \in I_S\}$.

- **PartDecrypt$(PK, K_I, CT') \rightarrow$ PCT**:

When the blockchain receives the transformation key K_I from the data user, it decides whether to perform the partial decryption algorithm based on the ciphertext's access policy $\bar{\mathcal{F}} = (M, \rho)$. If the data user's attributes do not satisfy the access policy $\bar{\mathcal{F}} = (M, \rho)$, it returns $\perp$; otherwise, for $I = \{i: \rho(i) \in I_S\} \subset \{1,2,\dots,l\}$, the blockchain needs to find a set of coefficients $f_i \in Z_p^*$ such that $\sum_{i \in I} f_i M_i = (1,0,\dots,0)$, hence $\sum_{i \in I} f_i \lambda_i = s$. The blockchain returns the partial decryption ciphertext PCT to the data user, where $PCT = \frac{e(C_\eta, K_6)}{(\prod_{i \in I}(e(B_I, K_5) e(C_i', K_5) e(K'_{\rho(i)}, D_i))^{f_i})} = e(g,g)^{\frac{\alpha s x}{\beta}} e(g,g)^{s\tau}$.

- **Decrypt$(sk', PCT, K_\tau) \rightarrow$ m**:

1. After receiving the partial decryption ciphertext PCTPCT from the blockchain, the data user decrypts CR and Ck in CT′ to obtain the ciphertext file address R and symmetric key k, respectively. The decryption is as follows: $Dec(C_R, sk', PCT, K_\tau) = \frac{C_R \cdot e(C_\eta, K_\tau)^{K_2}}{PCT^{K_2}}$, $Dec(C_k, sk', PCT, K_\tau) = \frac{C_k \cdot e(C_\eta, K_\tau)^{K_2}}{PCT^{K_2}} = k$.
2. The data user retrieves the required data ciphertext from the blockchain using the file address R, and then decrypts the ciphertext using the symmetric key kk to obtain the plaintext grid data mm.
3. When the blockchain receives a membership change notification from the key generation center, it updates the attribute group keys θ_i for attribute i affected by the membership change. Suppose a membership change occurs in attribute group G_j, the blockchain randomly selects $s' \in Z_p^*$ and a new attribute group key θ_j' to update the ciphertext CT', while the attribute group key does not need updating. The updated ciphertext is as follows: $CT' = \{\bar{\mathcal{F}}, C_k = ke(g,g)^{\alpha(s+s')}, C_R = Re(g,g)^{\alpha(s+s')}, C_\eta = g^{s+s'}, B_j = g^{\mu(\lambda_j + s')}, C_j' = \left(T_{\rho(j)}^{-r_j \cdot t_{\rho(j)}}\right)^{\theta_j'}, \{B_i = g^{\mu(\lambda_i + s')}, C_i = \left(T_{\rho(i)}^{-r_i \cdot t_{\rho(i)}}\right)^{\theta_i}, D_i = g^{r_i}\}\}_{i \in [1,l] \setminus \rho(j)}$
4. Due to the update of the membership list for attribute j, the blockchain selects a new minimum cover set for G_j and uses the updated $KEK(G_j)$ to form a new header for the ciphertext CT', where $Hdr = (\{Enc_K(\theta_j')\}_{K \in KEK(G_j)}, i \in [1,l] \setminus \{\rho(j)\}: \{Enc_k(\theta_i)\}_{K \in KEK(G_i)})$.

5 Security Analysis

Theorem 1 (Data privacy) If the ABE scheme and the searchable encryption are semantically secure against chosen plaintext attack, our scheme achieves data privacy as Definition 1.

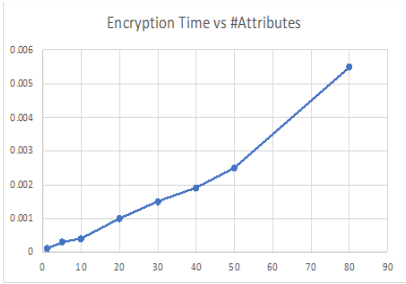
Proof. We only give the proof sketch. Since all the grid data is encrypted using ABE, no PPT adversary can gain essential knowledge as long as the secret key it controlled cannot meet the prespecified attributes. On the other hand, the index ciphertext is independent from the data. Hence, we can conclude our scheme is semantically secure.

Theorem 2 (Search correctness) If the searchable encryption has correctness, our scheme provides search correctness.

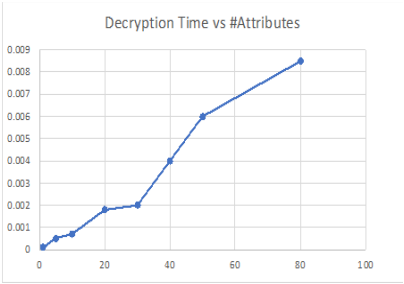
Proof. We only give the proof sketch. Note that ABE ciphertexts do not affect search correctness. It is easy to see that if the underlying searchable encryption scheme is correct, so it is with the whole scheme.

6 Performance Evaluation

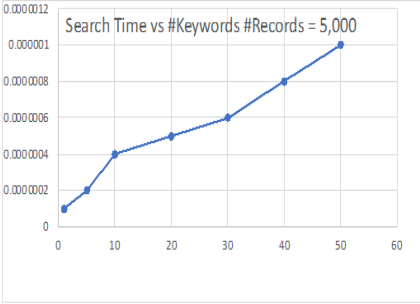
We implement our scheme using NTL library and Charm on a server with 5th Gen Intel Xeon CPU with 32 cores runs at 4.1GHz with 1TB memory. The operating system is CentOS Stream 9.



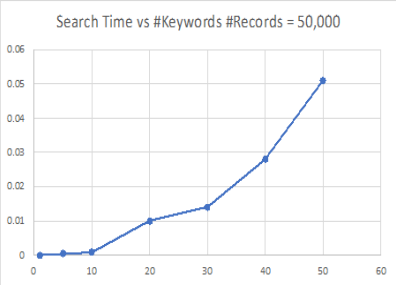
(a) Encryption Time (1MB data)



(b) Decryption Time (1MB data)



(c) Searching Performances (5000 records)



(d) Searching Performances (50,000 records)

Fig. 3. Encryption/ Decryption Time and Searching Performances

As shown in Fig. 3, we measured the encryption and decryption time for 1MB of data. For 80 attributes, both encryption and decryption were completed in less than 0.01 seconds. Additionally, we evaluated the query processing time. For 50 keywords over 50,000 records, the search time was at most 0.005 seconds. These results demonstrate the practicality of our scheme.

7 Conclusion

We proposed a scheme for power grid data sharing using blockchains using attribute-based encryption and symmetric searchable encryption. The results of our experiments show our scheme is secure and efficient. We remark that future research should further improve the throughput of the online search phase.

Acknowledgement

The authors would like to thank the anonymous reviewers for valuable comments. This work was partially supported by Guangzhou Power Supply Bureau, Guangdong Power Grid Co. Ltd. (No. 0301002023030301XX00149), National Natural Science Foundation of China (No.62172411, No.U2336205) and Beijing Municipal Science and Technology Project (No. Z231100005923047).

Reference

1. PCC (2007) Climate Change 2007: The Fourth Assessment Report of the Inter-Governmental Panel on Climate Change. Cambridge University Press, Cambridge.
2. Ju C, Shen Z, Bao F, et al. A novel credible carbon footprint traceability system for low carbon economy using blockchain technology[J]. International journal of environmental research and public health, 2022, 19(16): 10316.
3. Xu Y, Tao X, Das M, et al. A blockchain-based framework for carbon management towards construction material and product certification[J]. Advanced Engineering Informatics, 2024, 61: 102242.
4. Bellare M, Boldyreva A, O'Neill A. Deterministic and efficiently searchable encryption[C]//Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007. Proceedings 27. Springer Berlin Heidelberg, 2007: 535-552.
5. Chai Q, Gong G. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers[C]//2012 IEEE international conference on communications (ICC). IEEE, 2012: 917-922.
6. Zhang, H., Zeng, S., Yang, J.: Backward private dynamic searchable encryption with update pattern. Inf. Sci. 624, 1–19 (2023).
7. A. Sahai, B.Waters. Fuzzy Identity-Based Encryption. EUROCRYPT 2005. Lecture Notes in Computer Science, vol 3494. pp 457–473, Springer.
8. Y. Rouselakis, B. Waters. Practical Constructions and New Proof Methods for Large Universe Attribute-Based Encryption. CCS 2013: 463-474.
9. B. Qin, R. H. Deng, S. Liu and S. Ma, Attribute-Based Encryption with Efficient Verifiable Outsourced Decryption," IEEE Transactions on Information Forensics and Security, vol. 10, no. 7, pp. 1384-1393, 2015.
10. S. Lin, R. Zhang, H. Ma and M. Wang, "Revisiting Attribute-Based Encryption with Verifiable Outsourced Decryption," in IEEE Transactions on Information Forensics and Security, vol. 10, no. 10, pp. 2119-2130, 2015.