INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE
E. Günsay, Vol.13, No.1, pp. 23-39
https://doi.org/10.55859/ijiss.1410041

Decentralized Anonymous IoT Data Sharing with Key-Private Proxy Re-Encryption

Esra Günsay[1] , Oğuz Yayla[1]

[1]Institute of Applied Mathematics, Middle East Technical University, Ankara, Turkey
Corresponding Author: esra.gunsay@metu.edu.tr

Research Paper Received: 26.12.2023 Revised: 13.02.2024 Accepted: 01.03.2024

Abstract—Secure and scalable data sharing is one of the main concerns of the Internet of Things (IoT) ecosystem. In this paper, we introduce a novel blockchain-based data-sharing construction designed to ensure full anonymity for both the users and the data. To share the encrypted IoT data stored on the cloud, users generate tokens, prove their ownership using zk-SNARKs, and target the destination address anonymously. To tackle the privacy concerns arising from uploading the data to the cloud, we use key-private re-encryption and share only the necessary information with the proxy. As the first time in the literature, we have integrated a token-based blockchain and a key private proxy re-encryption to achieve a fully anonymous data sharing scheme. Furthermore, we provide security proof of our proposed scheme is secure against existential forgery under chosen-plaintext attacks, under eDBDH assumption in the random oracle model.

Keywords—proxy re-encryption, blockchain, IoT data sharing, zero-knowledge proofs

1. Introduction

In the past few years, IoT technology has become essential in many constructions, such as smart homes [1], smart grids [2], autonomous vehicles [3], and smart healthcare [4] systems. With the development of 5G, the importance of this technology will significantly increase and be widely used. According to Global System for Mobile Communications Foundation (GSMA), 5G connections are expected to grow to 1.8 billion, and total IoT connections are expected to touch 25.2 billion by 2025 [5]. In such systems, a massive amount of data is collected and shared among stakeholders according to need or request. Management of the IoT data, i.e., storing and sharing it while preserving privacy and confidentiality, emerges as an essential problem. So that these systems have to supply some crucial requirements such as user identification and authentication, permission authorization, permission to access data, scaling data integrity, and others.

As an example, smart health systems are used to securely record, store and share sensitive data without allowing any malicious changes. These systems are of great importance for regular follow-up of the conditions of the patients. Since the data will be used for future clinical studies, keeping these data unchanged is essential for ensuring that these studies are reliable and trustworthy. Therewith, while the sensitive personal information of the patients is stored, providing the necessary access control to the relevant parties is of great importance in terms of providing solutions to the system needs of the user.

Considering the technical requirements of such data storage and sharing systems, the use of distributed ledger technologies (DLT) emerges as a solution [6]. The characteristics of the blockchain technology enable us to build constructions that provide non-tampering and anonymity in a decentralized way.

Besides privacy concerns, dealing with large-scale IoT data has essential issues such as limited computing and storage capacity. Storing the encrypted data itself on the blockchain will require extremely high resources. A common approach to deal with these restrictions is to keep the sensitive data on the cloud servers. However, one of the drawbacks of this approach is that the cloud servers are highly prone to malicious usage so that it is crucial to trust the cloud servers as little as possible.

1.1. Related Works

In the literature, many recent studies are focusing on the privacy concern of data storing and sharing. Some of them use a blockchain-assisted method together with a proxy re-encryption (PRE) [7], [8], [9]. The main drawback of these studies is that in many PRE schemes, the proxy can easily determine the participants of the communication from the re-encryption key.

Manzoor et al. [10] proposed a blockchain-based IoT data-sharing scheme that uses pairing-free proxy re-encryption. Their system uses dynamic smart contracts to eliminate untrusted third parties. To protect data privacy, they use the proxy re-encryption so that the data is only visible to the participants in the smart contract. By employing a smart contract they managed the financial transactions automatically so that they eliminated the manual verification steps and some predefined requirements. Though their construction efficiently eliminates a need for a trusted third-party, it has challenges adapting blockchain platforms, resulting in throughput and latency problems.

In 2021, Yang et al. [9] presented a blockchain-based data-sharing scheme that uses a proxy re-encryption technique based on identity together with certificateless encryption for medical institutions. Since the communication is constructed in between medical institutions, they do not need to fully anonymize the participants though the data is anonymous. Their construction is resistant to identity disguise and replay attacks.

Recently, Song et al. [8] adopted blockchain-based data traceability and sharing mechanism for the power material supply chain. They use proxy re-encryption to ensure security and privacy. For their use case, data needs to be traceable, which is a feature we avoid in our case to keep anonymity.

Zonda and Meddeb [11] focused on sharing data among organizations, particularly a use case of carpooling. Their scheme is integrated within smart contracts together with a proxy re-encryption technique. They kept the encrypted data on-chain, which may cause scalability problems. On the other hand, the identity of the data owner is not hidden from the proxy.

Feng et al. [7] proposed a blockchain privacy protection scheme based on the zero-knowledge proof for secure data sharing via smart contracts for industrial IoT. They keep the encrypted sensitive data in the cloud and share the hash and the digital signature in the blockchain. Using zk-SNARKs with a combination of a smart contract, they aim the data availability between the owner and requester. For their use case, complete traceability of the data has importance. On the other hand, for a fully-anonymous data-sharing scheme, data needs to be untraceable.

To protect the large-scale IoT health data,

Healtchain is introduced by Xu et al. [12]. They used two different blockchains for fine-grained access control; one chain is for users, while the other is for doctor's diagnoses. They used a content-addressable distributed file system to store the data and stored only the hash of the data on the blockchain.

FHIRChain [13] is another blockchain-based architecture to solve the data sharing problem for clinical decision-making. They used digital signatures for tamper-proofing and public key encryption to prevent unauthorized access and spoofing. They also proposed a DApp to analyze the benefits and limitations of their designed scheme.

Recently, Zhang et al. [14] proposed an identity-based broadcast encryption method for data sharing in vehicular ad hoc networks. Their scheme operates independently of the real identity of the vehicle and eliminates reliance on third parties for hash values. The decryption process is independent of the number of receivers, and the scheme boasts fixed ciphertext length and system public parameters, resulting in a lower overall cost compared to alternatives.

In 2023, Ge et al. [15] addressed the challenge of revoking users from the sharing set in cloud data sharing. Their solution involved introducing an attribute-based proxy re-encryption scheme featuring a direct revocation mechanism. The authors supported their proposal with experimental analysis to support their findings.

Keshta et al.[16] focused on the data access problem of blockchain data-sharing systems. They offered to utilize a hybrid attribute-based proxy re-encryption approach, allowing the proxy server to convert attribute-encrypted ciphertexts into identity-based encrypted ciphertexts to make the previously encrypted data accessible for users with limited resources.

In 2004, Ben-Sasson et al. [17] proposed Zerocash decentralized anonymous payment (DAP) scheme using zk-SNARKs. It enables users to pay each other privately, hiding the origin and destination of the payment, and transferred amount. That is why we take this study as a cornerstone of our proposed system.

In Table 1, we tabulated the comparison of previous data-sharing schemes with our proposal. For the anonymity, partial means, while the data is anonymous by some unauthorized parties; data source or data direction is not hidden from authorized parties i.e. medical researchers, proxy, data owner, etc. It can be seen that many previous schemes are traceable. This is because in many previous use cases (e.g., carpooling, medical diagnosis, vehicular communication systems), it is desired that the data be traceable according to the problem definition. However, after some proof of validity, we want the data to be untraceable to achieve complete anonymity. We also specified the PRE schemas used by other studies in the literature. Note that while previous studies were based on a smart contract (SC), our study differs from these studies in that it is a token-based architecture.

1.2. Our Contribution

In order to solve the problem of IoT data privacy, security, availability, and consistency, we propose a token-based system that allows the anonymous sharing of secret information. Here, we proposed a novel token-based decentralized construction integrated with key private proxy re-encryption to achieve fully anonymous data sharing for the first time. This novel approach not only ensures CPA security but also establishes a framework for achieving complete anonymity in data sharing, representing the first instance of such a comprehensive integration. We offer a novel scheme where only sensitive information

Table 1.
Comparisons of known constructions.

	Manzoor et al.	Yang et al.	Song et al.	Zonda and Meddeb	Feng et al.	Keshta et al.	Our Scheme
Anonymity	Partial	Partial	Partial	Partial	Partial	Partial	✓
Untraceability	✗	✗	✗	✗	✗	✗	✓
Proxy Re-Encryption	CB-PRE	ID-based PRE	keyword search PRE	Changeable	ID-based PRE	AB-PRE	KP-PRE
Blockchain	SC	SC	SC	SC	SC	SC+BC	Token
Cloud Server	✓	✓	✗	✗	✓	✓	✓

is shared with authorized users without revealing the identity of the recipients to both the proxy and the users in the system. The recipient of the data knows that it comes from a valid person thanks to certain zero-knowledge proofs, but the identity of the sender is not disclosed. The contributions of our scheme are as follows:

- We propose a scheme based on distributed ledger technology due to its wide range of usage areas that deploy the trusted central party. The main advantage of using blockchain is to keep the previous token transactions on the chain in an immutable way. Even though we achieve full anonymity keeping the transaction records is essential to prevent any malicious attempt. Instead of smart contracts, we design a token-based structure to provide both scalability and anonymity concerns. By revising the DAP construction in [17], we propose a novel token-based data-sharing construction.
- We use key private proxy re-encryption to encrypt the data securely before storing it on the cloud. Since this method allows two types of encryption, i.e., the first level (non-re-encryptable) and the second level (re-encryptable), we use the second level encryption to store data while using the first level for other required system information on transactions. For this encryption method, it is impossible to derive the participants' identities from the re-encryption key.
- We conduct a thorough examination of the security aspects of our proposed scheme, validate its correctness, and provide a comprehensive security proof under the CPA framework.

1.3. Organization of the paper

The remainder of the paper is organized as follows. Section 2 provides an overview of the preliminaries to the subject together with the underlying key-private proxy re-encryption scheme. Section 3 describes our proposed architecture by illustrating the pseudocode of the transactions. Section 4 analyses the security, i.e., gives the proof of correctness and anonymity. Section 5 presents concluding remarks and future work.

2. Preliminaries

We propose a token-based system that allows the anonymous sharing of secret information. Our data-sharing scheme comprises of 4 entities: data owner, requester, secure cloud, and blockchain network. These entities can be identified as follows:

1 *Data owner* is the party who owns the IoT devices. After the IoT data is encrypted and stored by the data owner, he/she also needs to generate a mint transaction to generate the corresponding token. Moreover, the data owner generates the re-encryption key and publishes the *share transaction*.

2 *Requester* is the user who searches for a token by checking the public ledger using his secret encryption key.
3 *Cloud server (Proxy)* is the place we store our encrypted IoT data. Proxy scans all the *share transactions* published by the users and executes the re-encryption process It also publishes a new type of *share transaction*, which is scannable and readable by the users.
4 *Blockchain network* is where we have the public ledger and share transactions by users and proxy. A snapshot of the ledger is available to all users whenever they want to access it.

Because of scalability and sensitivity problems of the many data sharing e.g., clinical data, we only add the access pointer of the encrypted data to the blockchain system and keep the sensitive information off-chain, i.e., on a secure cloud. An address access pointer is a reference that denotes the exact location of the encrypted data on the cloud, which also can be considered as the address of the encrypted data. In order to get a cost-effective designed system in terms of storage and transaction fees, access pointers related to a data set are used instead of adding encrypted data to a block.

The data addresses can be added to the blockchain by exposing secure access tokens to data. These secure tokens are published on the public ledger for decentralized access. For non-traceability, the data in the tokens also hold the hiding and binding properties. In addition to those tokens, an immutable transaction log of all events related to exchanging and actually consuming these tokens is maintained on the public ledger.

2.1. Cryptographic Primitives

We apply a revised approach of Zerocash to our problem and use similar cryptographic techniques to build our proposed scheme with anonymity.

We use a *collision-resistant hash function* (CRH) to compress the input string; and a *pseudorandom function* (PRF) to securely generate public address keys from a given secret address key as a seed. We use a *trapdoor commitment function* $\mathrm{comm}_r(x)$ for a given trapdoor r and an input x to statistically hide and computationally bind the input to the committed value. *Digital signatures* are used in this study to verify digital messages' authenticity. For a given security parameter λ, $\mathrm{KeyGen}_{\mathrm{Sign}}$ generates the signature key pair $pk_{\mathrm{sig}}, sk_{\mathrm{sig}}$. The message m is signed as $\sigma = \mathrm{Sign}(sk_{\mathrm{sig}}; m)$, and verified by checking the accuracy of $m = (pk_{\mathrm{sig}}; m, \sigma)$.

2.2. Proxy re-encryption (PRE)

The idea of PRE was proposed by Blaze et al. [18] in 1998. After its introduction, PRE has been used in a wide range of areas such as distributed file systems [19], access control [20], email forwarding [21], cloud [22], and others. It is of great importance which PRE scheme as the underlying re-encryption we will use to set up the scheme that serves our purpose. There are different types of PRE schemes with the their key features as: attribute-based setting [23], [24], [25], identity-based setting [26], [27], broadcast setting [28], [29], schemes using keyword search [30], and similar.

2.2.1 Key-private proxy re-encryption

Our aim is to reveal as little information as possible to the proxy. So that the address keys, encryption keys, and the content of the message are kept hidden from the proxy. To encrypt the measured data, we use *key-private proxy re-encryption*, which is a unidirectional, single-hop, CPA-secure PRE method with key privacy. A detailed explanation of the system is given in [31]. For convenience, we first

give the underlying key-private PRE scheme and then explain the overall architecture.

The scheme is based on pairing-based cryptography. Let q be a prime number and $\mathbf{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map, denoted by $(q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$, where $\mathbb{G}$ is an additive cyclic group of order q generated by g and $\mathbb{G}_T$ is another group of order q. There are five polynomial-time algorithms in the key-private PRE scheme: SetUp, KeyGen, Encrypt, ReEncrypt, and Decrypt.

Setup(1^k)**:** For a randomly chosen $h \in \mathbb{G}$, $Z = \mathbf{e}(g, h)$ is computed so that the public parameters of the system are (g, h, Z).

KeyGen: Choose $u_1, u_2 \xleftarrow{\$} \mathbb{Z}_q$. For each user in the system public encryption keys are (Z^{u_1}, g^{u_2}), with the corresponding secret key (u_1, u_2).

Encryption: User A with the secret key (a_1, a_2) encrypts his data m with the corresponding public key (Z^{a_1}, g^{a_2}) by first selecting a random $k \in \mathbb{Z}_q$, and computing

$$E = (g^k, h^k, mZ^{a_1 k}) = (\alpha, \beta, \gamma). \tag{1}$$

We refer to the result of this encryption as the second-level ciphertext. With the same public, the user can also generate a first-level ciphertext as:

$$\tilde{E} = (\mathbf{e}(g^{a_2}, h)^k, mZ^k) = (Z^{a_2 k}, mZ^k). \tag{2}$$

ReKeyGen: A re-encryption key is generated by selecting random elements $r, w \in \mathbb{Z}_q$ and computing

$$\begin{aligned} rk_{A \rightarrow B} &= ((g^{b_2})^{a_1 + r}, h^r, \mathbf{e}(g^{b_2}, h)^w, \mathbf{e}(g, h)^w), \\ &= (g^{b_2(a_1+r)}, h^r, Z^{b_2 w}, Z^w), \\ &= (R_1, R_2, R_3, R_4). \end{aligned} \tag{3}$$

Re-Encryption: Using $rk_{A \rightarrow B}$, the re-encrypt operation on the encrypted data (α, β, γ) is done as in the following steps.

1 Check that $\mathbf{e}(\alpha, h) = \mathbf{e}(g, \beta)$. If it holds, then there exist $k \in \mathbb{Z}_q$ and $m \in \mathbb{G}_T$ such that $\alpha = g^k$, $\beta = h^k$ and $\gamma = mZ^{a_1 k}$.
2 Compute:

$$\begin{aligned} t_1 &= \mathbf{e}(R_1, \beta) = \mathbf{e}(g^{b_2(a_1+r)}, h^k) = Z^{b_2 k(a_1+r)}. \\ t_2 &= \gamma \mathbf{e}(\alpha, R_2) = mZ^{a_1 k} \mathbf{e}(g^k, h^k) = mZ^{k(a_1+r)}. \end{aligned} \tag{4}$$

3 Choose a random $w' \in \mathbb{Z}_q$.
4 Re-randomize t_1 and t_2 into θ and δ respectively as:

$$\begin{aligned} \theta &= t_1 . R_3^{w'} = Z^{b_2 k(a_1+r)} . (Z^{w b_2})^{w'} = Z^{b_2(k(a_1+r)+ww')}. \\ \delta &= t_2 . R_4^{w'} = mZ^{k(a_1+r)} . (Z^w)^{w'} = mZ^{k(a_1+r)+ww'}. \end{aligned} \tag{5}$$

5 Publish the ciphertext $E' = (\theta, \delta)$, which is called as the second-level ciphertext.

Decryption: User B can decrypt the first-level ciphertext $\tilde{E}$ with his secret key (b_1, b_2) as follows:

$$m = \delta / \theta^{1/b_2}. \tag{6}$$

He can also decrypt the second-level ciphertext E' as:

$$m = \gamma / \mathbf{e}(\alpha, h)^{b_1}. \tag{7}$$

3. Proposed Scheme

The overall architecture for secure storing and anonymous sharing of the measured IoT data is demonstrated in Figure 1.

1 The data owner, user A, encrypts his measured IoT data using his *key-private* public key and stores it on the cloud server. Note that the result of this encryption is a second-level (i.e., re-encryptable) ciphertext.
2 User A generates a token including his public address key and information to reach out to data. He publishes a *mint transaction* to the ledger. At the same time, he sends the commitment of the token to the commitment list,

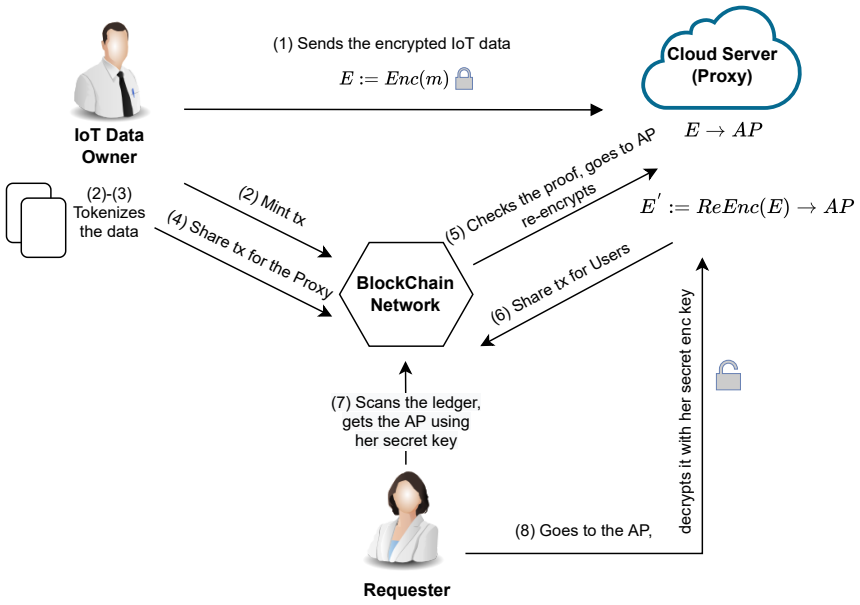

Figure 1. Workflow of our proposed scheme.

namely CMList. This token will be used to prove his ownership in a secret way.

3 When he wants to share his data with some other user B, he generates a new token, including the address public key of the B. Note that he also shares a *mint transaction* for the new token and sends the commitment of the token to the CMList. This process corresponds to the 'A. Mint' headline in the pseudocode in Figure 3.

4 He publishes a *share transaction* including:

- Merkle root of commitment list,
- commitment of the token related to requester,
- re-encryption key,
- digital signatures,
- a zk-SNARK proof that proves his ownership without revealing his address,
- encryption of trapdoors and access pointer as first-level ciphertext using the public encryption key of user B,
- encryption of trapdoors and access pointer as first-level ciphertext using the public encryption key of the proxy.

This process corresponds to the 'B. Share from users to Proxy' headline in the pseudocode in Figure 3.

5 As soon as the transaction is added to the ledger, the proxy reads the transaction and checks the accuracy of the zero-knowledge proof. If the proof is valid, it decrypts the related area with its secret encryption key and gets the AP, and then re-encrypts the value in AP with the corresponding re-encryption key.

6 Proxy publishes a new *share transaction*, which is quite similar to the *share transaction* the user A generates; it just eliminates the parts that are not related to user B so that the transaction includes:

- Merkle root of commitment list,
- commitment of the token related to requester,
- digital signatures,

- a zk-SNARK proof that proves his ownership without revealing his address,
- encryption of trapdoors and access pointer as first-level ciphertext using the public encryption key of user B.

This process corresponds to the 'C. Receive and share from Proxy to users' headline in the pseudocode in Figure 3.

7 User B scans the *share transactions* on the ledger; using her secret encryption key, she finds the related transaction and decrypts it. This process corresponds to the 'D. Verify transaction' headline in the pseudocode in Figure 3.

8 After learning the address access pointer AP shared with her, she decrypts the ciphertext on the cloud using her secret encryption key. This process corresponds to the 'E. Receive' headline in the pseudocode in Figure 3.

Note that the system has two types of *share transactions*. One type is generated by the users, and such transactions are only scanned by the proxy. The other type is generated by the Proxy and published to all the users in the system.

3.1. Architecture Description

We give the pseudocode of the system beginning from minting in Figure 3. In our construction, pp denotes the public parameters. defined by the trusted setup. Note that this setup only occurs at the very beginning of the system, afterwise there will be no need for any type of trusted party.

Each user has a pair of address keys (a_{pk}, a_{sk}), which will be used for hiding the origin of the transactions, and a pair of encryption keys (pk_{enc}, sk_{enc}) to encrypt the secret information. We will represent these keys as $\text{addr}_{pk} := (a_{pk}, pk_{enc})$, $\text{addr}_{sk} := (a_{sk}, sk_{enc})$. To be able to give users the flexibility to change their addresses; we use a pseudo-random function $\text{PRF}_{a_{sk}}()$ for address keys. After choosing a random secret address key a_{sk}, a user generates the corresponding address public key as $a_{pk} := PRF_{a_{sk}}(0)$. Note that encryption keys are $pk_{enc} = (Z^{a_1}, g^{a_2})$, $sk_{enc} = (a_1, a_2)$ as defined previously.

3.1.1 Storing the data on the cloud

Assume that (pk^A_{enc}, sk^A_{enc}) denotes the *key-private encryption* keys of the data owner. The data owner encrypts the measured data m with his public encryption key $pk^A_{enc} = (Z^{a_1}, g^{a_2})$, and gets the second-level ciphertext $E = \text{Enc}(pk^A_{enc}; m)$. He stores the encrypted data on a cloud storage server, where the access pointer AP denotes the exact location of the data on the server.

3.1.2 Tokenizing the data

After storing the measured data m as encrypted in the cloud, the data owner knows the exact location of the data. However, to send the data anonymously, he somehow needs to prove that he owns the data in a zero-knowledge way. To this end, for each encrypted data on the cloud, users generate a token t including the information of the ownership, i.e., the address key of the owner.

The tokens are generated to be able to exchange data. When a user wants to share his measured data, he sends the corresponding token to the other party, which is a certain way of sending the decryption rights of the data. The sensitive information in the tokens needs to be hidden to maintain anonymity. For this aim, a statistically hiding non-interactive commitment scheme is used. User A generates a token for the access pointer AP as follows:

$$\begin{aligned} k &:= \text{comm}_r(a^A_{pk}), \\ \text{cm}^A &:= \text{comm}_s(k \mid\mid AP), \end{aligned} \tag{8}$$

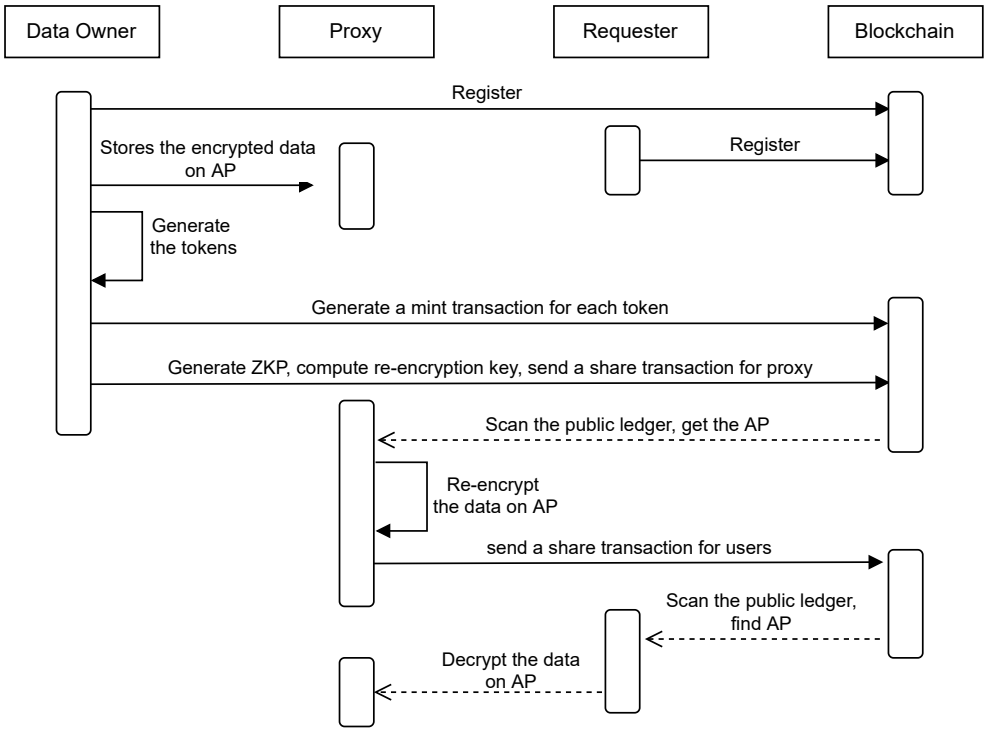

Figure 2. The timing diagram of our data sharing scheme.

The data owner chooses random trapdoors r and s, then commits his address public key to hide the origin of the token together with the access pointer. To do so, he would prove that given the access pointer, he owns the data on the location of AP indicates without revealing his address key. Similar to the DAP scheme of Zerocash, he sends cm^A to the CMList. To reduce the time and space complexity, the CMList is compressed as an efficiently updatable append-only CRH-based Merkle-tree structure whose root is denoted by rt.

He sets his token as $\mathbf{t}^A := (a^A_{pk}, AP, r, s, \mathrm{cm}^A)$. The token commitments are appended to the ledger after they are minted. Subsequently, he generates a mint transaction as:

$$\mathrm{tx}_{\mathrm{Mint}} = (k, s, \mathrm{cm}^A) \tag{9}$$

A mint transaction indicates that for a given location AP, there exists a token whose commitment cm^A is at the CMList.

3.1.3 Sending a transaction for proxy

If the data owner wants to share his data anonymously with some other user B, he needs to generate a *share transaction*. Using the address public key committed in $\mathbf{t}^A$, he is able to prove the origin in a zero-knowledge way. On the other hand, to prove the direction of the transaction anonymously, he generates another token that commits the address of the recipient.

First, the data owner generates a new token to indicate the direction of sharing; to this end includes the address public key of user B to the new token

as follows:

$$\begin{aligned} k' &:= \text{comm}_{r'}(a^B_{pk}), \\ \text{cm}^B &:= \text{comm}_{s'}(k' \mid\mid AP), \\ \text{tx}^B_{\text{Mint}} &= (AP, k', s', \text{cm}^B). \end{aligned} \tag{10}$$

The new token is set as $\mathbf{t}^B := (a^B_{pk}, AP, r', s', \text{cm}^B)$. User A mints this new token and sends the corresponding commitment cm^B to the CMList.

Second, user A computes a re-encryption key $rk_{A\to B}$ by using his own secret encryption key sk^A_{enc} and the public encryption key of the requester pk^B_{enc} as described in Eq.(3).

Third, to tackle the trace problems that might arise from sending AP disclosed, the user A sends it encrypted to the proxy. Aside from a little trust in the proxy, the reason for this encryption is to hide AP from other users scanning the ledger. Even if the proxy acts maliciously, the leaked information about AP does not violate the anonymity. The leaked information is just a random access pointer for an outside user. Hence, user A encrypts the AP with the public encryption key of the proxy:

$$PC := \text{Enc}(pk^{Proxy}_{enc};\ AP \mid\mid \text{nonce}). \tag{11}$$

He also needs to send trapdoors r' and s' in a secret way to let the user B open up the commitments. So that he encrypts the trapdoors using the public encryption key of user B. Since there is no need to re-encrypt these ciphertexts, he uses first-level encryption in this step. Let UC denotes the encryption of $\{r', s'\}$ under pk^B_{enc}:

$$UC := \text{Enc}(pk^B_{enc};\ AP, r', s'). \tag{12}$$

Third, to prove his ownership of the data located on AP, he generates a zk-SNARK proof π_{share} containing:

Given Merkle root rt, access pointer AP, and commitment cm^B, I know $\mathbf{t}^A$ and $\mathbf{t}^B$ s.t.:

- *The tokens $\mathbf{t}^A$ and $\mathbf{t}^B$ are well-formed.*
- *Address secret key matches with the address public key: $a^A_{pk} = PRF_{a^A_{sk}}(0)$.*
- *The token commitment cm^A appears as a leaf of a Merkle tree with root rt.*

Lastly, the data owner samples a signature key $(pk_{\text{sig}}, sk_{\text{sig}})$ to prevent the malleability attacks on the transaction he will share. He computes;

$$\begin{aligned} h_{\text{sig}} &:= \text{CRH}(pk_{\text{sig}}), \\ h_1 &:= \text{CRH}(h_{\text{sig}}). \end{aligned} \tag{13}$$

Later, generates two signatures; σ_1 for the proxy, and the σ_2 for the requester.

$$\begin{aligned} \sigma_1 &:= \text{Sign}(sk_{\text{sig}}, (rt, \text{cm}^B, h_{\text{sig}}, h_1, \pi_{\text{share}}, PC)) \\ \sigma_2 &:= \text{Sign}(sk_{\text{sig}}, (rt, \text{cm}^B, h_{\text{sig}}, h_1, \pi_{\text{share}}, UC)) \end{aligned} \tag{14}$$

Then adds the π_{share} to prove that these two signatures are well formed, i.e., computed correctly, and appends these signatures to the *share transactions*. Remember that in the overall system, there are two types of *share transactions*: one is generated by the users while the proxy generates the other. Now he publishes the *share transaction* for the proxy:

$$\begin{aligned} \text{tx}^U_{\text{share}} := (rt, \text{cm}^B, rk_{A\to B}, pk_{\text{sig}}, h_1, \pi_{\text{share}}, \\ PC, UC, \sigma_1, \sigma_2) \end{aligned} \tag{15}$$

3.1.4 Proxy cloud operations

As soon as a user publishes a transaction proxy is notified and operates on it. The proxy first checks the accuracy of the π_{share}, and σ_1. Then it decrypts the PC using its secret encryption key and gets the access pointer AP. After that, using $rk_{A\to B}$, he re-encrypts the data on the AP. At the end of this re-encryption, it generates a new *share transaction* for the users:

$$\text{tx}^P_{\text{share}} := (rt, \text{cm}^B, pk_{\text{sign}}, h_1, \pi_{\text{share}}, UC, \sigma_2). \tag{16}$$

A. Mint

- INPUTS:
  - public parameters pp
  - access pointer AP
  - corresponding address public key addr_{pk}
- OUTPUTS: a token t and mint transaction tx_{Mint}

1 Parse addr_{pk} as (a_{pk}, pk_{enc}).
2 Randomly sample two trapdoors r, s.
3 Compute $k := \text{comm}_r(a_{pk})$.
4 Compute $\text{cm} := \text{comm}_s(k||AP)$.
5 Set $\mathbf{t} := (a_{pk}, AP, r, s, \text{cm})$.
6 Set $\text{tx}_{\text{Mint}} = (AP, \text{cm}, *)$ where $* := (k, s)$.
7 Output $\mathbf{t}$ and tx_{Mint}.

B. Share from users to Proxy

- INPUTS:
  - public parameters pp
  - Merkle root rt
  - sender's token $\mathbf{t}^A$
  - sender's secret key a_{sk}
  - path **path** from commitment $\text{cm}(\mathbf{t}^A)$ to root rt
  - new address public key addr^B_{pk}
  - transaction string info
- OUTPUTS: token $\mathbf{t}^B$ and share transaction $\text{tx}^U_{\text{share}}$

1 Parse $\mathbf{t}^A$ as $(a^A_{pk}, AP, r, s, \text{cm}^A)$.
2 Parse addr^A_{sk} as (a^A_{sk}, sk^A_{enc})
3 Parse addr^B_{pk} as (a^B_{pk}, pk^B_{enc})
4 Randomly sample two new trapdoors $r^{'}, s^{'}$.
5 Compute $k^{'} := \text{comm}_{r^{'}}(a^B_{pk})$.
6 Compute $\text{cm}^B := \text{comm}_{s^{'}}(k^{'}||AP)$.
7 Set $\mathbf{t}^B := (a^B_{pk}, AP, r^{'}, s^{'}, \text{cm}^B)$.
8 Set $UC := \text{Enc}(pk^B_{enc}; AP, r^{'}, s^{'})$.
9 Set $PC := \text{Enc}(pk^{\text{Proxy}}_{enc}; AP||\text{nonce})$.
10 Generate $(pk_{\text{sig}}, sk_{\text{sig}}) := \text{KeyGen}_{\text{Sign}}$.
11 Compute $h_{\text{sig}} := \text{CRH}(pk_{\text{sig}})$.
12 Compute $h_1 := \text{PRF}_{a^A_{sk}}(h_{\text{sig}})$.
13 Compute $rk_{A\to B}$.
14 Set $\vec{x} := (rt, \text{cm}^B, h_{\text{sig}}, h_1)$.
15 Set $\vec{a} := (\textbf{path}, \mathbf{t}^A, a^A_{sk}, \mathbf{t}^B)$.
16 Compute $\pi_{\text{share}} := \text{Prove}(pk_{\text{share}}, \vec{x}, \vec{a})$.
17 Set $m_1 := (\vec{x}, \pi_{\text{share}}, PC)$.
18 Set $m_2 := (\vec{x}, \pi_{\text{share}}, UC)$.
19 Compute $\sigma_1 := \text{Sign}(sk_{\text{sig}}, m_1)$.
20 Compute $\sigma_2 := \text{Sign}(sk_{\text{sig}}, m_2)$.
21 Set $\text{tx}^U_{\text{share}} := (rt, \text{cm}^B, rk_{A\to B}, *)$, where $* := (pk_{\text{sig}}, h_1, \pi_{\text{share}}, UC, PC, \sigma_1, \sigma_2)$.
22 Output $\mathbf{t}^B$ and tx_{share}.

C. Receive and share from Proxy to users

- INPUTS:
  - transaction $\text{tx}^U_{\text{share}}$
- OUTPUTS: a share transaction $\text{tx}^P_{\text{share}}$

1 Parse $\text{tx}^U_{\text{share}}$ as $(rt, \text{cm}^B, rk_{A\to B}, *)$, where $* := (pk_{\text{sig}}, h_1, \pi_{\text{share}}, UC, PC, \sigma_1, \sigma_2)$.
2 Compute $AP = \text{Dec}(sk^{\text{Proxy}}_{enc}; PC)$.
3 Compute $E^{'} = \text{ReEnc}(rk_{A\to B}; E)$.
4 Set $\text{tx}^P_{\text{share}} := (rt, \text{cm}^B, *)$, where $* := (pk_{\text{sig}}, h_1, \pi_{\text{share}}, UC, \sigma_2)$.
5 Output $\text{tx}^P_{\text{share}}$.

D. Verify Transaction

- INPUTS:
  - public parameters pp
  - a (mint or share) transaction
  - the current ledger L
- OUTPUTS: a bit b

1 If $tx = \text{tx}_{\text{Mint}}$:
  a) Parse tx_{Mint} as $(\text{cm}, AP, *)$, and $*$ as (k, s).
  b) Set $\text{cm}^{'} := \text{comm}_s(AP||k)$.
  c) If $\text{cm} = \text{cm}^{'}$ output $b = 1$ else output $b = 0$.
2 If $\text{tx}=\text{tx}^P_{\text{share}}$:
  a) Parse $\text{tx}^U_{\text{share}}$ as $(rt, \text{cm}^B, rk_{A\to B}, *)$, where $* := (pk_{\text{sig}}, h_1, \pi_{\text{share}}, UC, PC, \sigma_1, \sigma_2)$.
  b) If the Merkle root rt does not appear on L outp $b = 0$.
  c) Compute $h_{\text{sig}} := \text{CRH}(pk_{\text{sign}})$.
  d) Set $x := (rt, \text{cm}^B, h_{\text{sig}}, h_1)$ where $* := (k, s)$.
  e) Set $m := (x, \pi_{\text{share}}, PC)$.
  f) Compute $b := V_{\text{sig}}(pl_{\text{sig}}, m, \sigma_1)$.
  g) Compute $b^{'} := \text{Verify}.(vk_{\text{share},x,\pi_{\text{share}}})$, output $b \wedge b^{'}$
3 If $tx = \text{tx}^U_{\text{share}}$
  a) Parse $\text{tx}^U_{\text{share}} := (rt, \text{cm}^B, rk_{A\to B}, *)$, where $* := (pk_{\text{sign}}, h_1, \pi_{\text{share}}, UC, \sigma_1, \sigma_2)$.
  b) If the Merkle root rt does not appear on L outp $b = 0$.
  c) Compute $h_{\text{sig}} := \text{CRH}(pk_{\text{sign}})$.
  d) Set $x := (rt, \text{cm}^B, h_{\text{sig}}, h_1)$ where $* := (k, s)$.
  e) Set $m := (x, \pi_{\text{share}}, UC)$.
  f) Compute $b := V_{\text{sig}}(pl_{\text{sig}}, m, \sigma_2)$.
  g) Compute $b^{'} := \text{Verify}(vk_{\text{share},x,\pi_{\text{share}}})$, output $b \wedge b^{'}$.

E. Receive

- INPUTS:
  - public parameters pp
  - recipient's address key pair (addr_{pk}, addr_{sk})
  - the current ledger L
- OUTPUTS: a received token

1 Parse addr_{sk} as (a_{sk}, sk_{enc}).
2 Parse addr_{pk} as (a_{pk}, pk_{enc}).
3 For each share transaction on the ledger:
  a) Parse $\text{tx}^P_{\text{share}}$ as $(rt, \text{cm}^B, *)$, where $* := (pk_{\text{sign}}, h_1, \pi_{\text{share}}, UC, \sigma_2)$.
  b) Compute $(AP, r^{'}, s^{'}) = \text{Dec}(sk^B_{enc}; UC)$.
  c) If the output of the previous step is not $\perp$, verif that:
    - $\text{cm}^B \stackrel{?}{=} \text{comm}_{s^{'}}(AP||\text{comm}_{r^{'}}(a^B_{pk}))$.
4 If it is valid go to AP on the cloud, compute m $\text{Dec}(sk_{enc}; E^{'})$.

Figure 3. Algorithm description of our proposed data sharing scheme.

Note that the Proxy does not compute any instances; it simply copies the related information from the *share transaction* generated by user A and appends it to the ledger, which is public to all users.

3.1.5 Decrypting the message

Using his secret encryption key sk^B_{enc}, the user B can find and decrypt the message by scanning the pour transactions. To be able to find $\text{tx}^P_{\text{share}} = (rt, \text{cm}^B, \pi_{\text{share}}, RP, UC)$, he computes:

$$(AP, r', s') = \text{Dec}(sk^B_{enc}; UC) \tag{17}$$

If the output of the decryption is not $\perp$, he verifies that

$$\text{cm}^B \stackrel{?}{=} \text{comm}_{s'}(AP \mid\mid \text{comm}_{r'}(a^B_{pk})). \tag{18}$$

If these equations hold, this is a valid transaction for sending data to the user B.

4. Security Analysis

In this section, we analyze the security of our proposed architecture.

4.1. Correctness Proof

For the correctness of our proposed scheme, we need to consider the transaction shared by the Proxy. It is easy to see that the requester, user B, can decrypt the UC using his secret encryption key (b_1, b_2), as follows:

$$(AP, r', s') = \delta/\theta^{1/b_2}. \tag{19}$$

Re-encrypted ciphertext on the AP can be decrypted as:

$$m = \gamma/\mathbf{e}(\alpha, h)^{b_1}. \tag{20}$$

Thus, the correctness holds as the correct execution of the each previous step.

4.2. Security Model

The definition of the underlying hard problem of our scheme in this paper, i.e., extended-Decisional Bilinear Diffie-Hellman (eDBDH), is given below.

Definition 1 *(Extended Decisional Bilinear Diffie-Hellman (eDBDH) Problem [32]): Let $(q, g, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ be a map generated with a security parameter 1^k. We define Adv^{IND}_{eDBDH} of an probabilistic polynomial time adversary $\mathcal{A}$,*

$$Pr[a, b, c \leftarrow \mathbb{Z}_q;\ \ x_1 \leftarrow \boldsymbol{e}(g, g)^{abc};\ \ z \leftarrow \{0, 1\};$$
$$z' \leftarrow \mathcal{A}(g, g^a, g^b, g^c, \boldsymbol{e}(g, g)^{bc^2}, x_z) :\ \ z = z'].$$

where the probability is taken over the random choices of $\mathcal{A}$ and the random selection of a, b, c. The adversary breaks the eDBDH problem, of there exists a negligible function ϵ s.t. $\geq \frac{1}{2} + \epsilon(k)$. eDBDH is hard if no such adversary exists.

The security is based on the indistinguishability of secret address keys and chosen-plaintext attack (CPA), $IND - CPA$. To this aim, we define the security game which has 5 stages. These games are run between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$.

1 *Setup phase:* The challenger takes the security parameter λ and sets up the system parameters, then the challenger provides $\mathcal{A}$ with the oracle access to public parameters, the secret parameters are kept hidden from $\mathcal{A}$. Also, chooses a random coin $c \in \{0, 1\}$, and keeps c secret.

2 *Phase 1:* The adversary makes the following queries to obtain $sk^* \in \mathbb{Z}_q$ and $(m_1, m_2 \in \mathcal{M})$, where $\mathcal{M}$ is the message space, as a result of this phase.

Key generation query $Q_{KeyGen} = (pp, i, m)$ $\mathcal{A}$ chooses an identity index i of a target receiver and a message m from message space. Sends these values to $\mathcal{C}$. Upon receiving values, $\mathcal{C}$ retrieves and returns the corresponding pk^i_{enc}.

Otherwise, forces $\mathcal{A}$ to choose a random coin $\hat{c} \leftarrow \{0,1\}$.

Re-encryption key generation query $Q_{ReKeyGen} = (pp, i, pk^j_{enc})$ $\mathcal{A}$ chooses an identity index i and pk^j_{enc} of a target receiver, j'th user and a message m from message space. Sends these values to $\mathcal{C}$. Upon receiving values, $\mathcal{C}$ first retrieves sk^i_{enc} and returns $rk_{A\rightarrow B}$. Otherwise, forces $\mathcal{A}$ to choose a random coin $\hat{c} \leftarrow \{0,1\}$. Note that $i \neq j$ to prevent trivial computation. To be able to prevent trivial results following conditions also be made:

a) When $\mathcal{A}$ queries (Dec, pk, E), return $\perp$.
b) When $\mathcal{A}$ queries $(Dec, pk, \tilde{E})$, return $\perp$.
c) When $\mathcal{A}$ queries $(ReEnc, pk^A, pk^B, E')$, return $\perp$.

3 *Challenge phase:* The following encryption query is made:

Encryption query $Q_{Enc} = (pp, sk^*_{enc}, m_0, m_1)$ $\mathcal{A}$ sends Q_{Enc}, then $\mathcal{C}$ flips a random coin b, and computes $E^* = (pp, sk^*_{enc}, m_b)$ to $\mathcal{A}$.

4 *Phase 2:* The adversary $\mathcal{A}$ makes polynomial number of queries to the oracle.

5 *Guess phase:* The adversary outputs a guess $b^* \in \{0,1\}$ of b. $\mathcal{A}$ wins the game if $b^* = b$. With the security in the random oracle, the adversary's advantage in the game is defined as:

$$Adv^{IND-CPA}_{eDBDH}(\lambda) = \left| Pr[b^* = b] - \frac{1}{2} \right| \quad (21)$$

The security of the proposed scheme against the attack is achieved if for all p.p.t. algorithms $\mathcal{A}$, the $Adv^{IND-CPA}_{eDBDH}(\lambda)$ is negligible.

4.3. Security Proof

Theorem 1 *The proposed system is IND-CPA secure in the random oracle model under the eDBDH assumption.*

Proof: $\mathcal{A}$ is considered as p.p.t algorithm with non-negligible advantage ϵ in $e^{IND-CPA}$. $\mathcal{A}$ is engaged to define another algorithm $\mathcal{C}$ having a non-negligible advantage in solving eDBDH. $\mathcal{C}$'s input $\langle \mathbb{G} = \langle g \rangle, g^a, g^b, g^c, J \rangle \in \mathbb{G} \times \mathbb{G}_T$, the output will be 1 if $J = \mathbf{e}(g,g)^{bc^2}$. the random oracle CRH is simulated as after address or signing public keys received, a random number θ is selected, and a random coin c is flipped with probability distribution $\mathcal{X}$. If $c = 0$; $\text{cm}^A \leftarrow \text{comm}_s(\theta \,||k\,||\, AP)$.

We show the interaction between $\mathcal{A}$ and $\mathcal{C}$ as follows.

1 *Setup phase:* $\mathcal{A}$ is given $pp = (q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, CRH)$ by $\mathcal{C}$. Concurrently, $\mathcal{C}$ generates a List_{pk}. Chooses a random coin $c \in \{0,1\}$, and keeps it secret.

2 *Find phase:* $\mathcal{A}$ sends the key generation query $Q_{KeyGen} = (pp, i)$. After receiving Q_{KeyGen}, $\mathcal{C}$ randomly selects $(u_1, u_2) \xleftarrow{\$} \mathbb{Z}_q$ as sk^i_{enc}, then evaluates and sends pk^i_{enc}. Later, $\mathcal{A}$ sends the query $Q_{ReKeyGen} = (pp, i, pk^j_{enc})$.

3 *Challenge phase:* $\mathcal{A}$ outputs $pk^{i^*}_{enc}, m_0, m_1$, where the identity of $pk^{i^*}_{enc}$ is non-trivial. $\mathcal{C}$ selects $b \leftarrow (0,1)$. Then computes the encryption $E^* = (pp, sk^{i^*}_{enc}, m_b)$.

4 *Guess phase:* Upon receiving E^*, $\mathcal{A}$ outputs its guess $b^* \in \{0,1\}$.

Note that in this game, it is infeasible to distinguish randomly generated E since it has the same distribution as the real-world E. Thus, at the end of the polynomial number of queries, the advantage of $\mathcal{A}$ to win this game where b^* is a random coin selected by $\mathcal{A}$ can be written as:

$$Adv^{A}_{IND-CPA}(\lambda) = |Pb[b = b^*] - 1/2| = \epsilon. \quad (22)$$

□

5. Properties and performance analysis

5.1. Informal analysis of security properties

Our scheme has the following properties:

1 **Anonymity:** Without any pre-assumptions, our proposed scheme satisfies user anonymity, i.e., it is difficult to reveal the identity of the data sharer and the receiver. The sensitive data is stored in the cloud in the form of ciphertext. At this point, we assume that the computing power of the adversaries is limited so that the secret key of the participants' cannot be obtained by the adversaries. Therefore the secret keys are secure. Note that only the access pointer of the encrypted data is transmitted as tokens. Since address public keys are kept hidden using a statistically hiding commitment scheme, the tokens leak no information about the transaction's origin or direction so that user anonymity is achieved. The data owner proves his ownership of the stored data in a zero-knowledge way.
Our scheme achieves high anonymity as a result of the following properties.
   - The access pointers leak no information about the tokens.
   - The commitments in the CMList are not directly related to the $\mathbf{t}^A$.
   - $\mathbf{t}^A$ leaks no information about its owner, i.e., the address public key of the new token targeted.
   - The participants' keys and the re-encryption keys are infeasible to be related.

2 **Non-traceability:** For non-traceability, the data in the tokens holds the hiding and binding properties. We use a commitment scheme and key-private encryption to hide data in the tokens. Although an immutable transaction log of all events related to exchanging and consuming these tokens is maintained on the public ledger, these logs reveal nothing to trace access tokens or the data itself.

3 **Access controllable:** Transactions for related tokens are published on the public ledger for decentralized access. At the very beginning, we create the tokens with the address public keys of the relevant persons, i.e., the data owner or the requester. In case of an attempt of malicious access, it will fail, as it is impossible to decrypt the ciphertext without a corresponding secret encryption key.

4 **Authentication:** Authentication means users prove their identity as a prerequisite to allowing access to resources in an information system [33]. In our system, each user has a unique secret address key, and when a user wants to share data with another user a token is generated with the public key of the related secret address key. Only the secret key of the requester is able to decrypt the encrypted data. Since we have fixed the address secret keys, we guarantee the link between the identity and the public key.

5 **Immutable:** Immutability means that the data can only be written, not modified or deleted [34]. Since the encrypted measured data is stored on a cloud server; the data owner could decrypt it using his secret encryption key to check integrity. At the same time, the requester could verify the equations above, and the integrity was ensured.

5.2. Performance analysis

Our proposed scheme is based on the bilinear map operations on $(\mathbf{e}, \mathbb{G}, \mathbb{G}_T)$. Therefore, the complexity of our scheme is dominated by the operations of exponentiation, pairing, signature, commitment, and CRH. Hence, we give the number of these operations in Table 2 to evaluate the computation com-

Table 2.
Complexity and security comparisons of known constructions.

	Our Scheme	Manzoor et al.[10]	Agyekum et al.[35]	Keshta et al.[16]
Enc + ReEnc	$2t_e + 9t_{eT} + 7t_p + 2t_h + 2t_c + 2t_s$	$5t_h + 4t_m + 3t_a$	$t_e + t_G + t_p + 2t_s$	$3t_m + 4t_h + 2t_a$
Blockchain	$t_{eT} + 2t_v + t_h$	-	-	-
Dec + Dec2	$t_c + 2t_{eT} + t_p$	$5t_h + 4t_m + 2t_a$	$3t_G + 2t_s$	$t_m + 2t_h + t_a$
Security	IND-CPA	Informal analysis	IND-CPA	IND-CCA

plexity of our scheme. Exponentiation on the group $\mathbb{G}$ and $\mathbb{G}_T$ are denoted by t_e and t_{eT} respectively. The cost of the pairing operation is denoted by t_p. In addition, we denote the cost of signature generation, signature verification, CRH and commitment as t_s, t_v, t_h, and t_c, respectively.

As the first step, the encryption of the measured IoT data uses second-level encryption ($2t_e + t_{eT}$), and the ciphertext size is given by $2|\mathbb{G}| + |\mathbb{G}_T|$. The tokenization step includes four commitments for two tokens (t_c). To construct the share transaction, the data owner computes two first-level encryptions ($2(2t_{eT} + t_p)$), two CRH ($2t_h$), two commitments ($2t_c$), the re-encryption key ($2t_e + 2t_{eT} + t_p$) and two signatures ($2t_s$). After that, the proxy uses a first-level decryption (t_{eT}) and a re-encryption ($4t_p + 2t_{eT}$). Since the proxy will share the same transaction by discarding some parts, there will be no computational cost in re-sharing process. Consequently, the requester scans the ledger and decrypts the first-level ciphertext (t_{eT}), after finding the correct transaction, checks the commitment (t_c) and decrypts the second-level ciphertext ($t_p + t_{eT}$).

Please note that since here as the first time in the literature, we have integrated a token-based blockchain and a key private proxy re-encryption to achieve a fully anonymous data sharing scheme. Performance comparison of the proposed scheme with others would not be fair. In addition, we proposed the cryptographic components in our scheme as a proof of concept. Hence, it would be a good future direction to implement our architecture to get the communication cost together with the transaction cost and latency analysis in blockchain measurements.

6. Concluding Remarks

In this paper, we have proposed a decentralized data-sharing architecture with the combination of a key-private proxy re-encryption scheme to ensure anonymity for both the data owner and the requester. The underlying encryption method we used is CPA-secure under eDBDH assumption. To recapitulate, our scheme stores the encrypted IoT data in the cloud to ensure the efficiency. For each data, a token including the address public key is generated. When a user wants to share his/her data, he simply generates another token including the requester's address public key, and generates a transaction with the related zero-knowledge proof of the ownership. Proxy re-encrypts the corresponding data without knowing the owner or the requester. The proxy publishes a new transaction by simply eliminating some parts that are not necessary for the requester.

References

[1] D. Fogli, R. Lanzilotti, and A. Piccinno, "End-user development tools for the smart home: A systematic literature review," in

Distributed, Ambient and Pervasive Interactions, N. Streitz and P. Markopoulos, Eds. Cham: Springer International Publishing, 2016, pp. 69–79.

[2] D. Zheng, K. Deng, Y. Zhang, J. Zhao, X. Zheng, and X. Ma, "Smart grid power trading based on consortium blockchain in internet of things," in *Algorithms and Architectures for Parallel Processing*, J. Vaidya and J. Li, Eds. Cham: Springer International Publishing, 2018, pp. 453–459.

[3] B. V. Philip, T. Alpcan, J. Jin, and M. Palaniswami, "Distributed real-time iot for autonomous vehicles," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 1131–1140, 2019.

[4] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of things for smart healthcare: Technologies, challenges, and opportunities," *IEEE Access*, vol. 5, pp. 26 521–26 544, 2017.

[5] GSMA, "The Internet of Things by 2025," Accessed Mar. 28, 2024. [Online]. Available: https://www.gsma.com/iot/wp-content/uploads/2018/08/GSMA-IoT-Infographic-2019.pdf

[6] G. Leeming, J. Cunningham, and J. Ainsworth, "A ledger of me: personalizing healthcare using blockchain technology," *Frontiers in medicine*, vol. 6, p. 171, 2019.

[7] T. Feng, P. Yang, C. Liu, F. Junli, and R. Ma, "Blockchain data privacy protection and sharing scheme based on zero-knowledge proof," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–11, 2022.

[8] J. Song, Y. Yang, J. Mei, G. Zhou, W. Qiu, Y. Wang, L. Xu, Y. Liu, J. Jiang, Z. Chu, W. Tan, and Z. Lin, "Proxy re-encryption-based traceability and sharing mechanism of the power material data in blockchain environment," *Energies*, vol. 15, no. 7, p. 2570, 2022.

[9] X. Yang, X. Li, A. Chen, and W. Xi, "Blockchain-based searchable proxy re-encryption scheme for ehr security storage and sharing," *Journal of Physics: Conference Series*, vol. 1828, p. 012120, 2021.

[10] A. Manzoor, A. Braeken, S. S. Kanhere, M. Ylianttila, and M. Liyanage, "Proxy re-encryption enabled secure and anonymous iot data sharing platform based on blockchain," *Journal of Network and Computer Applications*, vol. 176, p. 102917, 2021.

[11] D. Zonda and M. Meddeb, "Proxy re-encryption for privacy enhancement in blockchain: Carpooling use case," in *2020 IEEE International Conference on Blockchain (Blockchain)*, 2020, pp. 482–489.

[12] J. Xu, K. Xue, S. Li, H. Tian, H. Jianan, P. Hong, and N. Yu, "Healthchain: A blockchain-based privacy preserving scheme for large-scale health data," *IEEE Internet of Things Journal*, vol. 6, pp. 8770–8781, 2019.

[13] P. Zhang, J. White, D. C. Schmidt, G. Lenz, and S. T. Rosenbloom, "Fhirchain: applying blockchain to securely and scalably share clinical data," *Computational and structural biotechnology journal*, vol. 16, pp. 267–278, 2018.

[14] J. Zhang, S. Su, H. Zhong, J. Cui, and D. He, "Identity-based broadcast proxy re-encryption for flexible data sharing in vanets," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4830–4842, 2023.

[15] C. Ge, W. Susilo, Z. Liu, J. Baek, X. Luo, and L. Fang, "Attribute-based proxy re-encryption with direct revocation mechanism for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 949–960, 2024.

[16] I. Keshta, Y. Aoudni, M. Sandhu, A. Singh, P. A. Xalikovich, A. Rizwan, M. Soni, and S. Lalar, "Blockchain aware proxy re-encryption algorithm-based data sharing scheme," *Physical Communication*, vol. 58, p. 102048, 2023.

[17] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE symposium on security and privacy*. IEEE, 2014, pp. 459–474.

[18] M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *Advances in Cryptology — EUROCRYPT'98. Lecture Notes in Computer Science*, K. Nyberg, Ed., vol. 1403. Springer Berlin Heidelberg, 1998, pp. 127–144.

[19] E. Kirshanova, "Proxy re-encryption from lattices," in *Public-Key Cryptography – PKC 2014*, H. Krawczyk, Ed. Springer Berlin Heidelberg, 2014, pp. 77–94.

[20] G. Pareek and B. Purushothama, "Proxy re-encryption for fine-grained access control: Its applicability, security under stronger notions and performance," *Journal of Information Security and Applications*, vol. 54, p. 102543, 2020.

[21] L. Ibraimi, Q. Tang, P. Hartel, and W. Jonker, "A type-and-identity-based proxy re-encryption scheme and its application in healthcare," vol. 5159, 2008, pp. 185–198.

[22] D. Nuñez, I. Agudo, and J. Lopez, "Proxy re-encryption: Analysis of constructions and its application to secure access delegation," *Journal of Network and Computer Applications*, vol. 87, pp. 193–209, 2017.

[23] H. Deng, Z. Qin, Q. Wu, Z. Guan, and Y. Zhou, "Flexible attribute-based proxy re-encryption for efficient data sharing," *Information Sciences*, vol. 511, pp. 94–113, 2020.

[24] K. Liang, M. H. Au, J. K. Liu, W. Susilo, D. S. Wong, G. Yang, Y. Yu, and A. Yang, "A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing," *Future Generation Computer Systems*, vol. 52, pp. 95–108, 2015.

[25] C. Ge, W. Susilo, J. Baek, Z. Liu, J. Xia, and L. Fang, "A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 2907–2919, 2022.

[26] X. A. Wang, F. Xhafa, Z. Zheng, and J. Nie, "Identity based proxy re-encryption scheme (ibpre+) for secure cloud data sharing," in *2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2016, pp. 44–48.

[27] P. Dutta, W. Susilo, D. H. Duong, and P. S. Roy, "Collusion-resistant identity-based proxy re-encryption: lattice-based con-

structions in standard model," *Theoretical Computer Science*, vol. 871, pp. 16–29, 2021.

[28] Q. Zhang, J. Cui, H. Zhong, and L. Liu, "Toward data transmission security based on proxy broadcast re-encryption in edge collaboration," *ACM Transactions on Sensor Networks (TOSN)*, vol. 18, no. 3, pp. 1–27, 2022.

[29] Y. Liu, Y. Ren, C. Ge, J. Xia, and Q. Wang, "A cca-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system," *Journal of Information Security and Applications*, vol. 47, pp. 125–131, 2019.

[30] J. Shao, Z. Cao, X. Liang, and H. Lin, "Proxy re-encryption with keyword search," *Information Sciences*, vol. 180, no. 13, pp. 2576–2587, 2010.

[31] G. Ateniese, K. Benson, and S. Hohenberger, "Key-private proxy re-encryption," in *Topics in Cryptology – CT-RSA 2009*, M. Fischlin, Ed. Springer Berlin Heidelberg, 2009, pp. 279–294.

[32] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Privacy and Security*, vol. 9, no. 1, pp. 1–30, 2006.

[33] Z. Song, Z. Li, and W. Dou, "Different approaches for the formal definition of authentication property," in *9th Asia-Pacific Conference on Communications (IEEE Cat. No. 03EX732)*, vol. 2. IEEE, 2003, pp. 854–858.

[34] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," *ArXiv*, vol. abs/1906.11078, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:69842399

[35] K. O.-B. O. Agyekum, Q. Xia, E. B. Sifah, C. N. A. Cobblah, H. Xia, and J. Gao, "A proxy re-encryption approach to secure data sharing in the internet of things based on blockchain," *IEEE Systems Journal*, vol. 16, no. 1, pp. 1685–1696, 2021.