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

Esra Günsay$^{1[0000-0001-6672-4253]}$ and Oğuz Yayla$^{1[0000-0001-8945-2780]}$

Institute of Applied Mathematics,
Middle East Technical University, Ankara, Turkey
{gunsay,oguz}@metu.edu.tr

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 anonymously target the destination address. To tackle the privacy concerns arising from uploading the data to the cloud, we use key-private re-encryption and share as little information as possible with the proxy. Furthermore, we provide security proof of our construction.

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 home [10]s, smart grids [32]s, autonomous vehicles [21], 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 [1]. 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 [14]. 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) [9,24,29]. 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 [17] 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. [29] 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. [24] 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 [33] 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. [9] 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. [27]. 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 [30] 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.

In 2004, Ben-Sasson et al. [22] 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.

Table 1: Comparisons of known constructions

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

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. We offer a novel scheme where only sensitive information 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 [22], 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 analyze the security of our proposed scheme, confirm its correctness, and do its anonymity proof.

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. [5] in 1998. After its introduction, PRE has been used in a wide range of areas such as distributed file systems [13], access control [19], email forwarding [12], cloud [18], 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 [7,15,11], identity-based setting [26,8], broadcast setting [31,16], schemes using keyword search [23], and similar.

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 [2]. 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 $\mathbf{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 $u_1, u_2 \overset{\$}{\leftarrow} \mathbb{Z}_q$. 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

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

Re-Encryption: Using $rk_{A \to B}$, 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:

$$\begin{aligned} \theta &= t_1.R_3^{w'} = Z^{b_2 k(a_1+r)}.(Z^{wb_2})^{w'} = Z^{b_2(k(a_1+r)+ww')}. \\ \delta &= t_2.R_4^{w'} = mZ^{k(a_1+r)}.(Z^{w})^{w'} = mZ^{k(a_1+r)+ww'}. \end{aligned} \tag{5}$$

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.

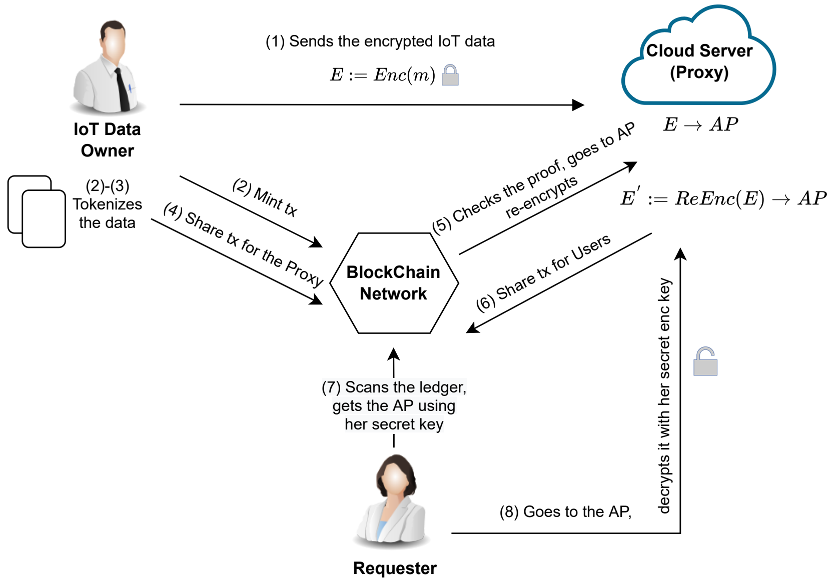
(1) Sends the encrypted IoT data
$E := Enc(m)$
Cloud Server (Proxy)
IoT Data Owner
$E \to AP$
(2)-(3) Tokenizes the data
(2) Mint tx
(4) Share tx for the Proxy
(5) Checks the proof, goes to AP re-encrypts
$E' := ReEnc(E) \to AP$
BlockChain Network
(6) Share tx for Users
(7) Scans the ledger, gets the AP using her secret key
(8) Goes to the AP,
decrypts it with her secret enc key
Requester

Fig. 1: Workflow of our proposed scheme.

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, 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.
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.
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.
7. User B scans the *share transactions* on the ledger; using her secret encryption key, she finds the related transaction and decrypts it.
8. After learning the address access pointer AP shared with her, she decrypts the ciphertext on the cloud using her secret encryption key.

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.

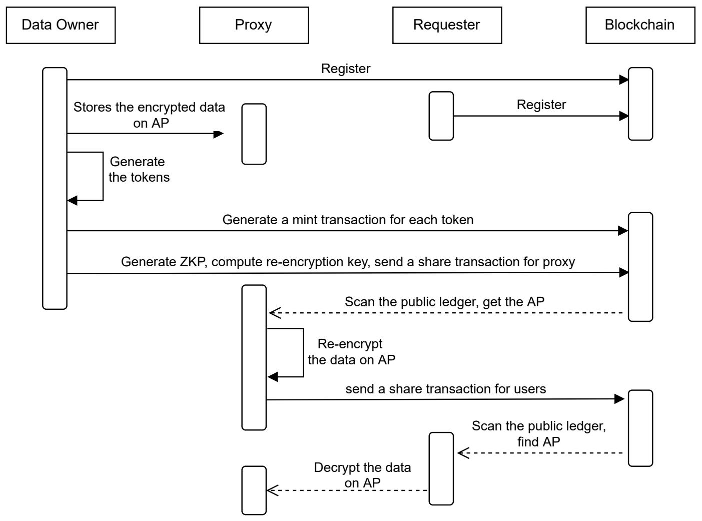
Data Owner
Proxy
Requester
Blockchain
Register
Stores the encrypted data on AP
Register
Generate the tokens
Generate a mint transaction for each token
Generate ZKP, compute re-encryption key, send a share transaction for proxy
Scan the public ledger, get the AP
Re-encrypt the data on AP
send a share transaction for users
Scan the public ledger, find AP
Decrypt the data on AP

Fig. 2: The timing diagram of our data sharing scheme.

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.

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.

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. We also need to keep the sensitive information in the tokens hidden to maintain anonymity. For this aim, we use a statistically hiding non-interactive commitment scheme. User A generates a token for the access pointer AP as follows:

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

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, we compress the CMList as an efficiently updatable append-only CRH-based Merkle-tree structure whose root is denoted by rt.

He sets his token as $\mathbf{t}^A := (a_{pk}^A, AP, r, s, \mathrm{cm}^A)$. 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.

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:

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

The new token is set as $\mathbf{t}^B := (a_{pk}^B, AP, r', s', \mathrm{cm}^B)$. 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_{enc}^A and the public encryption key of the requester pk_{enc}^B 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 our 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:

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

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_{enc}^{B}:

$$UC := \mathrm{Enc}(pk_{enc}^{B};\ AP, r', s'). \tag{12}$$

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* $\boldsymbol{t}^A$ *and* $\boldsymbol{t}^B$ *s.t.:*

- *The tokens* $\boldsymbol{t}^A$ *and* $\boldsymbol{t}^B$ *are well-formed.*
- *Address secret key matches with the address public key:* $a_{pk}^{A} = PRF_{a_{sk}^{A}}(0)$.
- *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;

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

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

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

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, we have 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:

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

Mint

```
• INPUTS:
    - public parameters pp
    - access pointer $AP$
    - corresponding address public key $\mathrm{addr}_{pk}$
• OUTPUTS: a token t and mint transaction $\mathrm{tx}_{\mathrm{Mint}}$
   1 Parse $\mathrm{addr}_{pk}$ as $(a_{pk}, pk_{enc})$.
   2 Randomly sample two trapdoors $r, s$.
   3 Compute $k := \mathrm{comm}_r(a_{pk})$.
   4 Compute $\mathrm{cm} := \mathrm{comm}_s(k||AP)$.
   5 Set $\mathbf{t} := (a_{pk}, AP, r, s, \mathrm{cm})$.
   6 Set $\mathrm{tx}_{\mathrm{Mint}} = (AP, \mathrm{cm}, *)$ where $* := (k, s)$.
   7 Output $\mathbf{t}$ and $\mathrm{tx}_{\mathrm{Mint}}$.
```

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

Receive and share from Proxy to users

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

Verify Transaction

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

Receive

```
• INPUTS:
    - public parameters pp
    - recipient's address key pair ($\mathrm{addr}_{pk}$, $\mathrm{addr}_{sk}$)
    - the current ledger $L$
• OUTPUTS: a received token
   1 Parse $\mathrm{addr}_{sk}$ as $(a_{sk}, sk_{enc})$.
   2 Parse $\mathrm{addr}_{pk}$ as $(a_{pk}, pk_{enc})$.
   3 For each share transaction on the ledger:
     a) Parse $\mathrm{tx}^P_{\mathrm{share}}$ as $(rt, \mathrm{cm}^B, *)$, where $* :=$
        $(pk_{\mathrm{sign}}, h_1, \pi_{\mathrm{share}}, UC, \sigma_2)$.
     b) Compute $(AP, r^{'}, s^{'}) = \mathrm{Dec}(sk^B_{enc}; UC)$.
     c) If the output of the previous step is not $\perp$, verify
        that:
          - $\mathrm{cm}^B \overset{?}{=} \mathrm{comm}_{s'}(AP||\mathrm{comm}_{r'}(a^B_{pk}))$.
   4 If it is valid go to $AP$ on the cloud, compute $m =$
     $\mathrm{Dec}(sk_{enc}; E^{'})$.
```

Fig. 3: Algorithm description of our proposed data sharing scheme.

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:

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

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.

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

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

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 Proof

We start by giving the definition of two well-known assumptions.

Definition 1 (Extended Decisional Bilinear Diffie-Hellman (eDBDH) Assumption [3]). *Let $(q, g, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ be a map generated with a security parameter 1^k. For all probabilistic polynomial time adversary $\mathcal{A}$, there exists a negligible function ϵ such that:*

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

Definition 2 (Decision Linear Assumption [6]). *Let $(q, g, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ be a map generated with a security parameter 1^k. Given that h, f are random generators in $\mathbb{G}$, for all probabilistic polynomial time adversary $\mathcal{A}$, there exists a negligible function ϵ such that:*

$$Pr[x, y, r \leftarrow \mathbb{Z}_q;\ \ q_1 \leftarrow f^{x+y};\ \ q_0 \leftarrow f^r;\ \ z \leftarrow \{0, 1\};$$
$$z^{'} \leftarrow \mathcal{A}(g, h, f, g^x, h^y, q_z, x_z) :\ \ z = z^{'}] \leq \frac{1}{2} + \epsilon(k).$$

Confidentiality. To prove confidentiality, two conditions need to be satisfied. First, the ciphertexts (we will assume a commitment is also ciphertext) leak no information about the messages. Second, the trapdoors reveal no information about the message. In order to prove these two clauses, we define two security games. These games are run between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ interacts with the oracles of hash functions and commitments.

1. Setup: $\mathcal{C}$ first takes a security parameter λ and sets up the system parameters, then sends the public parameters to adversary $\mathcal{A}$, and the secret parameters are kept hidden from $\mathcal{A}$.
2. Phase 1: For the different public address keys of the system, $\mathcal{A}$ adaptively issue the queries to the following oracle that is controlled by the $\mathcal{C}$:
   Commitment oracle $\mathcal{O}_c$: Given a message m, the oracle runs the *comm* algorithm and generates the corresponding commitment cm with related trapdoor r.
3. Challenge: $\mathcal{A}$ chooses two messages (m_0, m_1) to be challenged. $\mathcal{C}$ flips a random coin b, and computes the comm, then sends the commitment cm^* to $\mathcal{A}$.
4. Phase 2: The adversary $\mathcal{A}$ adaptively issues queries.
5. The adversary outputs a guess $b^{'} \in \{0, 1\}$ of b. $\mathcal{A}$ wins the game if $b^{'} = b$.

We define the $\mathcal{A}$'s advantage as:

$$Adv_A = \left| \Pr[b^{'} = b] - \frac{1}{2} \right|$$

Lemma 1. *If the eDBDH assumption holds and the hash function is collision-resistant, then our scheme satisfies the confidentiality in the random oracle model.*

We prove through the following games, which are constructed as defined above. Our aim is to show the advantages of $\mathcal{A}$ to win these games are negligible.

Game 0: Algorithm $\mathcal{C}$ simulates the challenger and interacts with $\mathcal{A}$. The challenger $\mathcal{C}$ first generates the public parameters $pp = (q, g, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$, and two address key pairs (a_{pk_i}, a_{sk_i}) and (a_{pk_j}, a_{sk_j}). Then, $\mathcal{C}$ publishes the public information pp, a_{pk_i}, a_{pk_j} and in response to commitment query, $\mathcal{C}$ generates two tokens:

$$\begin{aligned} k_1 &= \mathrm{comm}_{r_1}(a_{pk_i}), & k_2 &= \mathrm{comm}_{r_2}(a_{pk_j}), \\ \mathrm{cm}_1 &= \mathrm{comm}_s(k \mid\mid AP_1), & \mathrm{cm}_2 &= \mathrm{comm}_s(k \mid\mid AP_2), \end{aligned} \tag{21}$$

where r_1, r_2, s_1, s_2 chosen randomly. To response to trapdoor oracle query, $\mathcal{C}$ computes $k_* = \mathrm{comm}_{r_1 r_2}(a_{pk_i})$. Hence, $\mathcal{A}$'s ability to acquire the trapdoors is simulated.

By repeating the above queries, the adversary guesses the challenge commitment based on the information it has. We indicate the advantage of $\mathcal{A}$ to win this game as:

$$Adv_A^{Game\ 0}(\lambda) = Adv_A^{IND}(\lambda)$$

Game 1: The setup of this game is the same as Game 0. Note that CRH, PRF, and comm functions are constructed via the SHA256 hash function, which maps an arbitrary size input to 256-bit output. Challenger $\mathcal{C}$ selects new trapdoor values r_*, s_* and computes the new challenge commitment cm_*. Adversary $\mathcal{A}$ distinguishes cm_* from cm_1, so that $\mathcal{A}$ can find a path that reveals the commitment corresponding to a particular token as if it can break the collision resistancy of the hash function. One can see that the advantage of the adversary $\mathcal{A}$ to win this game is negligible, i.e., $Adv_A^{Game\ 1}(\lambda) = Adv_A^{Game\ 0}(\lambda)$ under the condition of the security of SHA256 hashing function [20]. Therefore adversary's advantage in finding a path in the CMList is negligible.

Game 2: In this game, challenger $\mathcal{C}$ selects new random trapdoors $r^{'}, s^{'}$ and to challenge the commitments in the share transaction computes the new commitment $\mathrm{cm}^* = \mathrm{comm}_{s'}(AP \mid\mid \mathrm{comm}_{r'}(a_{pk_1})$, which is published on share transaction on the blockchain. On the share transaction, the adversary $\mathcal{A}$ can decrypt $(AP, r', s') = \mathrm{Dec}(sk_{enc_*}; UC)$ under the condition of ability to break eBDBH difficulty, later verify $\mathrm{cm} = \mathrm{comm}_{s'}(AP \mid\mid \mathrm{comm}_{r'}(a_{pk}))$ under the condition of breaking hash security.

Thus, if the attacker $\mathcal{A}$ can win Game 2, then an algorithm $\mathcal{C}$ can be constructed to break the eDBDH problem and collusion resistance of the hash function. Therefore, we denote the advantage of $\mathcal{A}$ to win this game as, $Adv_A^{Game\ 2}(\lambda) = Adv_A^{Game\ 1}(\lambda)$. Note that this game is valid for both *share transaction* types in the scheme, i.e., generated by users or generated by the proxy. Based on the difficulty of the eDBDH problem and hash function, our proposed scheme satisfies the security of the *share transaction*.

Lemma 2. *If eDBDH and Decision Linear assumptions hold, our scheme satisfies the anonymity of the both parties, the data owner and the requester.*

Proof. Our protocol uses key-private proxy re-encryption, which is CPA-secure under eDBDH assumption in $\mathbb{G}$ for message domain $\mathbb{G}_T$ [2]. Under the Decision Linear assumption in $\mathbb{G}$, the encryption method we use provides the key privacy.

Lemma 3. *If eDBDH assumption holds, our scheme satisfies the immutability of the data.*

Proof. In our scheme, each user stores the data after encrypting it via a second-level encryption as $E = \text{Enc}(pk^A_{enc}; m) = (g^k, h^k, mZ^{u_1 k}) = (\alpha, \beta, \gamma)$, where u_1 is the part of user's secret signing key. Under the eBDBH assumption, $\mathcal{A}$ cannot decrypt the ciphertext and alter the data. Therefore, based on the security of the eDBDH assumption, the advantage of $\mathcal{A}$ to forge immutability is negligible.

5 Properties and performance analysis

5.1 Properties analysis

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 [25]. 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 [28]. 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 complexity 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 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.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.

Table 2: Cost of parties in our scheme

	Data owner	Proxy	Blockchain	Requester
Cost	$2t_e + 6t_{eT} + 3t_p + 2t_h + 2t_c + 2t_s$	$4t_p + 3t_{eT}$	$t_{eT} + 2t_v + t_h$	$t_c + 2t_{eT} + t_p$

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. The internet of things by 2025. GSMA (2018), `https://www.gsma.com/iot/wp-content/uploads/2018/08/GSMA-IoT-Infographic-2019.pdf`
2. Ateniese, G., Benson, K., Hohenberger, S.: Key-private proxy re-encryption. In: Fischlin, M. (ed.) Topics in Cryptology – CT-RSA 2009. pp. 279–294. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
3. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans. Inf. Syst. Secur. **9**(1), 1–30 (feb 2006). https://doi.org/10.1145/1127345.1127346, `https://doi.org/10.1145/1127345.1127346`
4. Baker, S.B., Xiang, W., Atkinson, I.: Internet of things for smart healthcare: Technologies, challenges, and opportunities. IEEE Access **5**, 26521–26544 (2017). https://doi.org/10.1109/ACCESS.2017.2775180
5. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) Advances in Cryptology — EUROCRYPT'98. pp. 127–144. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
6. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Annual international cryptology conference. pp. 41–55. Springer (2004)
7. Deng, H., Qin, Z., Wu, Q., Guan, Z., Zhou, Y.: Flexible attribute-based proxy re-encryption for efficient data sharing. Information Sciences **511**, 94–113 (2020)

8. Dutta, P., Susilo, W., Duong, D.H., Roy, P.S.: Collusion-resistant identity-based proxy re-encryption: lattice-based constructions in standard model. Theoretical Computer Science **871**, 16–29 (2021)
9. Feng, T., Yang, P., Liu, C., Junli, F., Ma, R.: Blockchain data privacy protection and sharing scheme based on zero-knowledge proof. Wireless Communications and Mobile Computing **2022**, 1–11 (02 2022). https://doi.org/10.1155/2022/1040662
10. Fogli, D., Lanzilotti, R., Piccinno, A.: End-user development tools for the smart home: A systematic literature review. In: Streitz, N., Markopoulos, P. (eds.) Distributed, Ambient and Pervasive Interactions. pp. 69–79. Springer International Publishing, Cham (2016)
11. Ge, C., Susilo, W., Baek, J., Liu, Z., Xia, J., Fang, L.: A verifiable and fair attribute-based proxy re-encryption scheme for data sharing in clouds. IEEE Transactions on Dependable and Secure Computing **19**(5), 2907–2919 (2021)
12. Ibraimi, L., Tang, Q., Hartel, P., Jonker, W.: A type-and-identity-based proxy re-encryption scheme and its application in healthcare. vol. 5159, pp. 185–198 (08 2008). https://doi.org/10.1007/978-3-540-85259-9$_1$2
13. Kirshanova, E.: Proxy re-encryption from lattices. In: PKC. pp. 77–94. Springer (2014). https://doi.org/10.1007/978-3-642-54631-0$_5$, `https://www.iacr.org/archive/pkc2014/83830204/83830204.pdf`
14. Leeming, G., Cunningham, J., Ainsworth, J.: A ledger of me: personalizing healthcare using blockchain technology. Frontiers in medicine **6**, 171 (2019)
15. Liang, K., Au, M.H., Liu, J.K., Susilo, W., Wong, D.S., Yang, G., Yu, Y., Yang, A.: A secure and efficient ciphertext-policy attribute-based proxy re-encryption for cloud data sharing. Future Generation Computer Systems **52**, 95–108 (2015)
16. Liu, Y., Ren, Y., Ge, C., Xia, J., Wang, Q.: A cca-secure multi-conditional proxy broadcast re-encryption scheme for cloud storage system. Journal of Information Security and Applications **47**, 125–131 (2019)
17. Manzoor, A., Braeken, A., Kanhere, S.S., Ylianttila, M., Liyanage, M.: Proxy re-encryption enabled secure and anonymous iot data sharing platform based on blockchain. Journal of Network and Computer Applications **176**, 102917 (2021). https://doi.org/https://doi.org/10.1016/j.jnca.2020.102917, `https://www.sciencedirect.com/science/article/pii/S1084804520303763`
18. Nuñez, D., Agudo, I., Lopez, J.: Proxy re-encryption: Analysis of constructions and its application to secure access delegation. Journal of Network and Computer Applications **87**, 193–209 (2017). https://doi.org/https://doi.org/10.1016/j.jnca.2017.03.005, `https://www.sciencedirect.com/science/article/pii/S1084804517301078`
19. Pareek, G., Purushothama, B.: Proxy re-encryption for fine-grained access control: Its applicability, security under stronger notions and performance. Journal of Information Security and Applications **54**, 102543 (2020)
20. Penard, W., van Werkhoven, T.: On the secure hash algorithm family. Cryptography in context pp. 1–18 (2008)
21. Philip, B.V., Alpcan, T., Jin, J., Palaniswami, M.: Distributed real-time iot for autonomous vehicles. IEEE Transactions on Industrial Informatics **15**(2), 1131–1140 (2019). https://doi.org/10.1109/TII.2018.2877217
22. Sasson, E.B., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE symposium on security and privacy. pp. 459–474. IEEE (2014)
23. Shao, J., Cao, Z., Liang, X., Lin, H.: Proxy re-encryption with keyword search. Information Sciences **180**(13), 2576–2587 (2010)

24. Song, J., Yang, Y., Mei, J., Zhou, G., Qiu, W., Wang, Y., Xu, L., Liu, Y., Jiang, J., Chu, Z., Tan, W., Lin, Z.: Proxy re-encryption-based traceability and sharing mechanism of the power material data in blockchain environment. Energies **15**(7) (2022). https://doi.org/10.3390/en15072570, `https://www.mdpi.com/1996-1073/15/7/2570`
25. Song, Z., Li, Z., Dou, W.: Different approaches for the formal definition of authentication property. In: 9th Asia-Pacific Conference on Communications (IEEE Cat. No. 03EX732). vol. 2, pp. 854–858. IEEE (2003)
26. Wang, X.A., Xhafa, F., Zheng, Z., Nie, J.: Identity based proxy re-encryption scheme (ibpre+) for secure cloud data sharing. In: 2016 International Conference on Intelligent Networking and Collaborative Systems (INCoS). pp. 44–48 (2016). https://doi.org/10.1109/INCoS.2016.83
27. Xu, J., Xue, K., Li, S., Tian, H., Jianan, H., Hong, P., Yu, N.: Healthchain: A blockchain-based privacy preserving scheme for large-scale health data. IEEE Internet of Things Journal **PP**, 1–1 (06 2019). https://doi.org/10.1109/JIOT.2019.2923525
28. Yaga, D., Mell, P., Roby, N., Scarfone, K.: Nistir 8202 blockchain technology overview. National Institute of Standards and Technology, US Department of Commerce, Washington, USA (2018)
29. Yang, X., Li, X., Chen, A., Xi, W.: Blockchain-based searchable proxy re-encryption scheme for ehr security storage and sharing. Journal of Physics: Conference Series **1828**, 012120 (02 2021). https://doi.org/10.1088/1742-6596/1828/1/012120
30. Zhang, P., White, J., Schmidt, D.C., Lenz, G., Rosenbloom, S.T.: Fhirchain: applying blockchain to securely and scalably share clinical data. Computational and structural biotechnology journal **16**, 267–278 (2018)
31. Zhang, Q., Cui, J., Zhong, H., Liu, L.: Toward data transmission security based on proxy broadcast re-encryption in edge collaboration. ACM Transactions on Sensor Networks (TOSN) (2022)
32. Zheng, D., Deng, K., Zhang, Y., Zhao, J., Zheng, X., Ma, X.: Smart grid power trading based on consortium blockchain in internet of things. In: Vaidya, J., Li, J. (eds.) Algorithms and Architectures for Parallel Processing. pp. 453–459. Springer International Publishing, Cham (2018)
33. Zonda, D., Meddeb, M.: Proxy re-encryption for privacy enhancement in blockchain: Carpooling use case. In: 2020 IEEE International Conference on Blockchain (Blockchain). pp. 482–489 (2020). https://doi.org/10.1109/Blockchain50366.2020.00070