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 [5], smart grid [14], autonomous vehicles [8], and smart healthcare [3] systems. With the development of 5G, the importance of this technology will greatly improve and be more common. According to Global System for Mobile Communications Foundation (GSMA), 5G connections are expected to grow to 1.8 billion, and the number of total IoT connections is 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 important 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 similar.

As an example, smart health systems are used to securely record, store and share clinical 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, it is of great importance that the necessary access control can be provided to the relevant parties in terms of providing a solution to the user's needs of the system. Considering the technical requirements of such data storage and sharing systems, the use of distributed ledger technologies (DLT) for healthcare systems emerges as a solution [6].

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 substantial to trust the cloud servers as little as possible.

1.1 Related Works

In the literature, there are many recent studies focusing on the privacy concern of data storing and sharing. Some of them use blockchain together with a proxy re-encryption (PRE) [4,10,12]. 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 [7] proposed a blockchain-based IoT data sharing scheme that uses proxy re-encryption. Their system uses dynamic smart contracts to eliminate untrusted third party. To protect data privacy they use proxy re-encryption so that the data is only visible to the participants in the smart contract.

In 2021, Yang et al. [12] presented a blockchain-based data sharing scheme that uses a proxy re-encryption technique based on identity together with certificateless encryption for medical institutions. Their construction is resistant to identity disguise and replay attacks.

Recently, Song et al. [10] adopted blockchain-based data traceability and sharing mechanism for the power material supply chain. They use proxy re-encryption to ensure security and privacy.

Zonda and Meddeb [15] 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.

Feng et al. [4] proposed a blockchain privacy protection scheme based on the zero-knowledge proof for secure data sharing using smart contracts for Industrial IoT. Similar to our approach, they keep the encrypted sensitive data in the cloud and share the hash and the digital signature. 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. [11]. 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.

In 2004, Ben-Sasson et al. [9] proposed Zerocash decentralized anonymous payment (DAP) scheme using zk-SNARKs. It enables users to pay each other privately, e.g hiding the origin and destination of the payment, and transferred amount. That is why we take this study as a cornerstone.

1.2 Our Contribution

In order to solve the problem of data privacy, security, availability, and consistency, we propose a token-based system that allows the anonymous sharing of secret information. The contributions of our scheme are as follows:

– We propose our method on blockchain technology due to its wide range of usage areas using DLT to deploy the trusted central party. Instead of smart contracts, we design a token-based structure to provide both scalability and anonymity concerns. We use the DAP scheme of the Zerocash [9] and revise it as a 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 identities of the participants 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 analysis 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 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 *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 *trapdoored 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 signature key pairs $pk_{\mathrm{sig}, sk_{\mathrm{sig}}}$. 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 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. The 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.

There are five polynomial-time algorithms in the key-private PRE scheme: SetUp, KeyGen, Encrypt, ReEncrypt, and Decrypt. 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, where $\mathbb{G}$ is an additive cyclic group of order q generated by g and $\mathbb{G}_T$ is another group of order q.

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 first-level ciphertext.

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

$$\begin{aligned} rk_{A\to B} &= ((g^{b_2})^{a_1+r}, h^r, \mathbf{e}(g^{b_2}, h)^w, \mathbf{e}(g, h)^w), \\ &= (g^{b_2(a_1+r)}, h^r, Z^{b_2 w}, Z^w), \\ &= (R_1, R_2, R_3, R_4). \end{aligned} \tag{2}$$

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 some $k \in \mathbb{Z}_q$ and $m \in \mathbb{G}_T$ such that $\alpha = g^k$, $\beta = h^k$ and $\gamma = mZ^{a_1 k}$.
2. Compute:
$$\begin{aligned} t_1 &= \mathbf{e}(R_1, \beta) = \mathbf{e}(g^{b_2(a_1+r)}, h^k) = Z^{b_2 k(a_1+r)}. \\ t_2 &= \gamma \mathbf{e}(\alpha, R_2) = mZ^{a_1 k}\mathbf{e}(g^k, h^k) = mZ^{k(a_1+r)}. \end{aligned} \tag{3}$$
3. Choose a random $w' \in \mathbb{Z}_q$.
4. Re-randomize t_1 and t_2 into θ and δ respectively as:
$$\begin{aligned} \theta &= t_1.R_3^{w'} = Z^{b_2 k(a_1+r)}.(Z^{wb_2})^{w'} = Z^{b_2(k(a_1+r)+ww')}. \\ \delta &= t_2.R_4^{w'} = mZ^{k(a_1+r)}.(Z^w)^{w'} = mZ^{k(a_1+r)+ww'}. \end{aligned} \tag{4}$$
5. Publish the ciphertext $E' = (\theta, \delta)$, which is called as the second-level ciphertext.

Decryption: User B can decrypt the second-level ciphertext E' with his secret key (b_1, b_2) as follows:

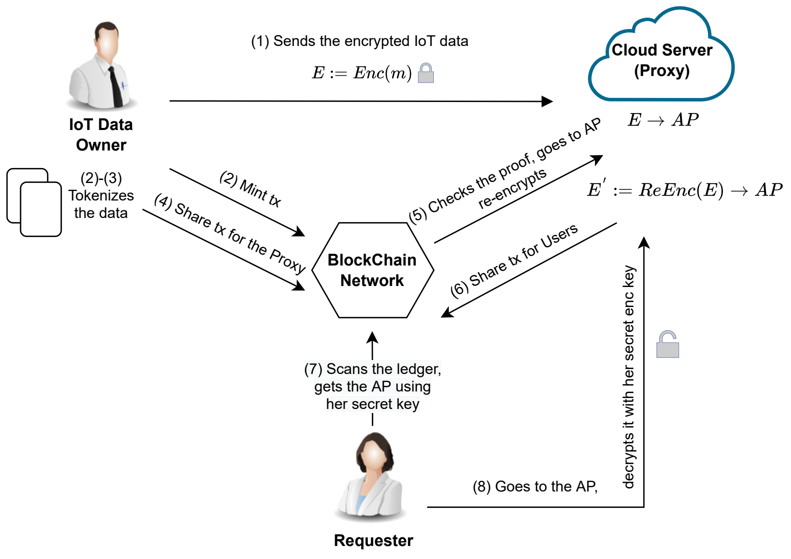
$$m = \delta/\theta^{1/b_2}. \tag{5}$$

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

$$m = \gamma/\mathbf{e}(\alpha, h)^{b_1}. \tag{6}$$

3 Proposed Architecture

Fig. 1 shows the overall architecture we devise to secure storing and anonymous sharing of the measured IoT data.



(1) Sends the encrypted IoT data
$E := Enc(m)$
Cloud Server (Proxy)
$E \rightarrow AP$
IoT Data Owner
(2)-(3) Tokenizes the data
(2) Mint tx
(4) Share tx for the Proxy
(5) Checks the proof, goes to AP re-encrypts
$E' := ReEnc(E) \rightarrow AP$
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 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. 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. Later 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 do not interest 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.

3.1 Architecture Description

We give the pseudocode of the system beginning from minting at Fig. 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 keys 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})

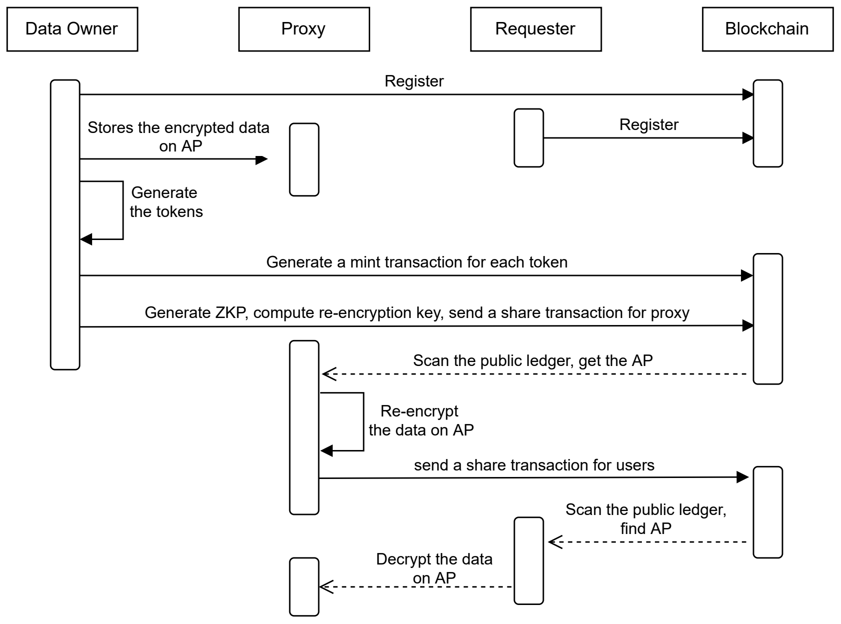

Data Owner
Proxy
Requester
Blockchain
Register
Stores the encrypted data on AP
Register
Generate the tokens
Generate a mint transaction for each token
Generate ZKP, compute re-encryption key, send a share transaction for proxy
Scan the public ledger, get the AP
Re-encrypt the data on AP
send a share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

Fig. 2: Timing diagram of our data sharing scheme.

to encrypt the secret information. We will represent these keys as $\mathrm{addr}_{pk} := (a_{pk}, pk_{enc})$,$\mathrm{addr}_{sk} := (a_{sk}, sk_{enc})$. To be able to give users the flexibility to change their addresses; we use a pseudo-random function $\mathrm{PRF}_{a_{sk}}()$ for address keys. After choosing a random secret address key a_{sk}, a user generates the corresponding address public key as $a_{pk} := PRF_{a_{sk}}(0)$. Remember 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 $E = \mathrm{Enc}(pk^A_{enc}; m)$.

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:

$$\begin{aligned} k &:= \mathrm{comm}_r(a_{pk}^A), \\ \mathrm{cm}^A &:= \mathrm{comm}_s(k||AP), \end{aligned} \tag{7}$$

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 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:

$$\mathrm{tx}_{\mathrm{Mint}} = (k, s, \mathrm{cm}^A) \tag{8}$$

Mint transactions indicate 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:

$$\begin{aligned} k' &:= \mathrm{comm}_{r'}(a_{pk}^B), \\ \mathrm{cm}^B &:= \mathrm{comm}_{s'} = (k'||AP), \\ \mathrm{tx}_{\mathrm{Mint}}^B &= (AP, k', s', \mathrm{cm}^B). \end{aligned} \tag{9}$$

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.(2).

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 no harm to anonymity. For an outside user, the leaked information is just random access pointers. So that the user A encrypts the AP with the public encryption key of the proxy:

$$PC := \mathrm{Enc}(pk_{enc}^{Proxy}; AP || \mathrm{nonce}). \tag{10}$$

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}:

$$UC := \mathrm{Enc}(pk_{enc}^{B}; AP, r', s'). \tag{11}$$

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 $(pk_{\mathrm{sig}}, sk_{\mathrm{sig}})$ to prevent the malleability attacks on the transaction he will share. He computes;

$$\begin{aligned} h_{\mathrm{sig}} &:= \mathrm{CRH}(pk_{\mathrm{sig}}), \\ h_1 &:= \mathrm{CRH}(h_{\mathrm{sig}}). \end{aligned} \tag{12}$$

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

$$\begin{aligned} \sigma_1 &:= \mathrm{Sign}(sk_{\mathrm{sig}}, (rt, \mathrm{cm}^B, h_{\mathrm{sig}}, h_1, \pi_{\mathrm{share}}, PC)). \\ \sigma_2 &:= \mathrm{Sign}(sk_{\mathrm{sig}}, (rt, \mathrm{cm}^B, h_{\mathrm{sig}}, h_1, \pi_{\mathrm{share}}, UC)). \end{aligned} \tag{13}$$

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:

$$\mathrm{tx}_{\mathrm{share}}^{U} := (rt, \mathrm{cm}^B, rk_{A \to B}, pk_{\mathrm{sig}}, h_1, \pi_{\mathrm{share}}, PC, UC, \sigma_1, \sigma_2) \tag{14}$$

Mint

```
• INPUTS:
    - public parameters pp
    - access pointer $AP$
    - corresponding address public key addr$_{pk}$
• OUTPUTS: a token t and mint transaction tx$_{\text{Mint}}$
    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 $\text{tx}_{\text{Mint}}$.
```

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 **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 $\text{tx}_{\text{share}}$.
```

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 $\text{tx}^P_{\text{share}}$.
```

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 $\text{tx}_{\text{Mint}}$ 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 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 $L$ output
       $b = 0$.
    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 $L$ output
       $b = 0$.
    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 $b \wedge b^{'}$.
```

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 $\perp$, verify
           that:
            - $\text{cm}^B \overset{?}{=} \text{comm}_{s'}(AP||\text{comm}_{r'}(a^B_{pk}))$.
    4 If it is valid go to $AP$ on the cloud, compute $m =$
      $\text{Dec}(sk_{enc}; E^{'})$.
```

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. Later, 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:

$$\text{tx}^P_{\text{share}} := (rt, \text{cm}^B, pk_{\text{sign}}, h_1, \pi_{\text{share}}, UC, \sigma_2). \tag{15}$$

Note that 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:

$$(AP, r', s') \overset{?}{=} \text{Dec}(sk^B_{enc}; UC) \tag{16}$$

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

$$\text{cm}^B \overset{?}{=} \text{comm}_{s'}(AP||\text{comm}_{r'}(a^B_{pk})) \tag{17}$$

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

4 Security Analysis

4.1 Correctness Analysis

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:

$$(AP, r', s') = \delta/\theta^{1/b_2}. \tag{18}$$

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

$$m = \gamma/\mathbf{e}(\alpha, h)^{b_1}. \tag{19}$$

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

4.2 Security Analysis

Immutability and integrity. 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.

Privacy and Anonymity. The protocol uses key-private proxy re-encryption. This encryption method is CPA-secure under EDBDH assumption in $\mathbb{G}$. For further information about the key privacy and the related proof, we refer to the original paper [2]. The sensitive data is stored in the cloud in the form of ciphertext. 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.

Note that the system leaks no information about:

- For which access pointer the tokens are generated.
- Which commitment in the CMList corresponds to the $\mathbf{t}^A$.
- For whom the $\mathbf{t}^A$ is shared for, i.e., the address public key of the new token is targetted.
- With which participants' keys the re-encryption key is generated.

So that we achieved full anonymity.

Authentication. Each user has a unique secret address key. Using a pseudo-random function, users can get as many addresses as they want. Since we have fixed the address secret keys, we guarantee the link between the identity and the public key.

Access control. 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.

Non-traceability. For non-traceability, the data in the tokens hold 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.

5 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 in $\mathbb{G}$. To recapitulate, our scheme stores the encrypted IoT data in the cloud to ensure the most efficient way. 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. 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
4. 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
5. 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)
6. Leeming, G., Cunningham, J., Ainsworth, J.: A ledger of me: personalizing healthcare using blockchain technology. Frontiers in medicine **6**, 171 (2019)
7. 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`
8. 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
9. 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)
10. 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`
11. 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
12. 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
13. 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)
14. 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)

15. 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