Design of Blockchain-Based Many-to-Many Anonymous Data Sharing Scheme

Esra Günsay[a,*], Burcu E. Karakaş[a], N. Gamze Orhon Kılıç[a], Oğuz Yayla[a]

[a]*Institute of Applied Mathematics,*
Middle East Technical University, Ankara, Turkey

Abstract

Many to many data sharing in the group setting in a cloud environment is a challenging problem that is crucial for numerous schemes. To our best knowledge, there is no generic study to allow sharing of confidential information in many to many pattern between different groups. Thus we propose a novel data sharing scheme enabling many to many sharing of encrypted data between different groups with using cryptographic techniques such as traceable ring signatures, multiple receiver key encapsulation, etc. We give a comprehensive security analysis showing our scheme is indistinguishable under user encapsulation keys and chosen plaintext attack secure under discrete logarithm assumption. We propose the implementation of our scheme, and the experimental results show that the proposed scheme is applicable for decentralized data sharing.

Keywords: Data sharing, Blockchain, Anonymity, Traceability, Key encapsulation

1. Introduction

Sharing information among different groups while preserving confidentiality and privacy is a challenging problem, especially in the case of traceability and linkability checks were necessary. We may divide the methods that share data into 3 main categories; one-to-one, one-to-many and many-to-many. In one to many pattern one of the users, usually the data owner shares the corresponding data with several number of receivers. On the other hand, many-to-many pattern allows multiple senders to establish an appropriate consensus and relay information to multiple recipients. Due to its wide range of usage areas, one-to-one and one-to-many data sharing schemes have been studied for years, and there are thousands of articles written on this subject, such as [1, 2, 3]. Though there exist many up-to-date studies [4, 5] enabling transmission of the data in the same group in a many-to-many pattern, only a few recent studies focused on many-to-many data sharing among different groups.

One common approach is integrating a group signature scheme [6] into the system to ensure data confidentiality. Group members can perform the transmission by signing the information of the relevant data using the group signature assigned to them by an authority, sometimes a group manager, or a certificate center. In the case that anonymity is required, leaked information is limited to the signature of an unknown member of that group. In most cases, however, the group manager has the authority to reveal the signer's identity when deemed necessary. This means tough users can access the data anonymously; only the group manager can trace it when a dispute occurs. This approach results in giving excessive power to the group manager, which we do not want in such a decentralized consensus.

As an example of many-to-many data sharing, consider a questionnaire form sharing between different institutes. Members of a group of people want to share a form containing confidential information while hiding their identities from some members of another group of people. One straightforward solution is to define a group encryption-decryption key for each group. This scenario gets more complicated when we restrict the potential receivers to some members of a defined group or ring of users. Moreover, each person should be restricted from submitting the questionnaire more than once in order for an accurate analysis to be made. If someone tries to submit more than one form, they are trying to forge the system by pretending to be someone else, so their identity needs to be revealed.

Another challenging problem is malicious users need

*Corresponding author.
Email addresses: gunsay@metu.edu.tr (Esra Günsay), burcuey@metu.edu.tr (Burcu E. Karakaş), gamze.kilic@metu.edu.tr (N. Gamze Orhon Kılıç), oguz@metu.edu.tr (Oğuz Yayla)

to be revoked from the system, so that dynamic user registration needs to be allowed. One trivial solution that comes to mind for such problems is to use an anonymous signature scheme [7], more precisely, using a group signature with a group manager and carrying out these operations by a trusted party. However, we do not want to assign a manager or trusted third party to achieve a fully decentralized system.

Besides the privacy concerns, while dealing with large-scale data, it is necessary to consider the efficiency problems that may arise in the overall system [8]. Sharing large-scale data on the blockchain causes network delays, inefficient bandwidth utilization, and an inability to make controls promptly. Therefore, it is preferred to share only the necessary information to access the data and perform the relevant security checks on the blockchain network.

The remainder of the paper is organized as follows. Section 2 provides a literature review of the subject and gives an overview of some related works. In Section 3, we give preliminaries to the subject and introduce the subprotocols we used in our framework. Section 4 describes our proposed architecture. Section 4 gives the corresponding security definitions and analyses the security, i.e., gives the proof of correctness and security. In Section 5, we tabulate the complexity of our scheme and give the performance analysis of our implementation with comparisons.

2. Related work

Many recent studies focus on the problem of secure and scalable data sharing. While the vast majority of the existing studies try to solve this problem in the cloud environment [15, 2, 16, 17], some of them propose new solutions with the distributed structure of the blockchain [18, 9, 19]. This section will briefly explain some recent studies related to our problem. The main drawback of these studies is that none of these studies focused on the problem of anonymous transmission of encrypted information between different groups.

In 2020, up to our knowledge, for the first time in the literature, Huang et al. [9] proposed a data-sharing scheme enabling many-to-many sharing among different groups. They used a group signature to sign the hash of the data to achieve traceability. In case of misusage, the group manager has the authority to reveal the identity of the misbehaviored user. To prevent the giving excessive power to group managers, they adopted a blockchain working as a trusted third party. Only the verification information are shared on the active blockchain network. However, the information to be transmitted in their scheme is not encrypted. This scheme cannot solve our problem when it comes to sending a large-scale document by protecting information privacy. Thus we consider this study as a cornerstone of our proposed system.

Agyekum et al. [10] presented pairing-based secure IoT data sharing using identity-based proxy re-encryption. They stored the IBE-based encrypted data on the cloud to achieve data confidentiality and used IBPRE to enable only legitimate users to access shared data. Also, users have access control lists stored on the blockchain to trace authorized users. In their framework, each user assigns unique names to their data which can be replicated and saved in network caches.

In 2022, Hu et al. [11] introduced GSChain, a sensor data sharing scheme in wireless sensor networks for multiple data sharers adopting a consortium blockchain, which will ensure auditability, high transaction rate, and data integrity. They encrypt the data with a sensor key and then define an asymmetric group key agreement protocol to encrypt/decrypt the sensor key, which will be shared on the blockchain together with encrypted data. Here the group key is updateable to enable group changes. Trusting an administrator to create the groups of data sharers is a method we avoid in order to provide anonymity in our own scheme.

Song et al. [12] focused on the traceability problem of data sharing in the power material supply chain systems. Herewith, they proposed to use a single-hop unidirectional proxy re-encryption scheme to ensure security and privacy. Since anonymity was not their concern, they used timestamps to trace data shared in the blockchain network.

Yin et al. [19] adopted blockchain to data sharing in Space-Ground Integrated Network implicitly for data analysis applications for multiple service providers and multiple users. They proposed a solution to cooperative train a particular model, such as federated learning, when different devices in different types of networks are in the case. In order to eliminate the central aggregator, they conducted a blockchain, which simply acts as a proxy. Model parameters are recorded in a distributed ledger to prevent misusage, and a snapshot of the ledger is public to all participants. Unlike our scheme, through the nature of their underlying problem, anonymity and traceability were not their concern.

Recently, Liu et al. [13] focused on sharing location information in a privacy-preserving way. Thus they proposed a multi-layer location-sharing scheme with keeping computational overhead and communication cost constant while the data integrity verification process.

Table 1: Comparisons of known constructions

	Huang et al.[9]	Agyekum et al.[10]	Hu et al.[11]	Song et al.[12]	Liu et al.[13]	Xu et al.[14]	Our Scheme
Pattern	MtM-dg	OtO	MtM-sg	OtO	OtO	OtO	MtM-dg
Anonymity	Partial	Partial	✗	✗	✗	✓	✓
Traceability/Linkability	✓	timestamp	✓	✓	✓	✓	✓
Decentralization	✓	✓	✓	✓	✓	✓	✓
Confidentiality	✗	IBE	sensorkey	PRE	RSA	hashing	AES
Auditability	group manager	PRE	✓	✓	data manager	✓	ring signature

They give data owners the right to manage their data to avoid the possibility of a single point of failure problem. In order to ensure dynamic data uploading, they used RSA based accumulator scheme [20] and used blockchain to eliminate the trust of centralized servers. Roughly, upon receiving a request from a requester, the data manager makes the identity checks and forwards the request to the data owner. Appropriate virtual location encrypted with the public key of the requester and a piece of witness information sent to the manager. Here, the manager manages the aggregation process and transmits the data.

Xu et al. [14] proposed Healthchain to protect large-scale IoT health data. Two different blockchains are deployed for fine-grained access control; Userchain allows users to upload their data via IoT sensors, while the Docchain is for doctors to share related diagnoses. They used a content-addressable distributed file system to store the data, and only the hash of the data is published on the blockchain.

In Table 1, we tabulated the comparison of previous data-sharing schemes with our proposal.

2.1. Our Contributions

In order to solve the problem of sharing sensitive data, which can be a form or questionnaire, in the many-to-many pattern for different groups, such as different institutes or organizations, we offer a novel data-sharing scheme that outsources the encrypted data and shares the encapsulation of the symmetric key together with a signature on the blockchain network. The contributions of our scheme are as follows:

- We introduce a data-sharing framework that achieves data confidentiality and user anonymity while preserving secure access control by data owners on the shared data. First time in the literature, our framework suggests using traceable ring signatures and key encapsulation mechanisms to solve data privacy and access control in sharing data in many-to-many pattern among different groups without trusting a third party.
- We provide a detailed explanation of our proposed scheme together with a comprehensive security analysis. Moreover, we presented the complexity analysis and comparisons of the overall scheme.
- We provide the implementation of our scheme and give the performance comparison with state-of-art. Results show that our framework is applicable to real-world scenarios.

3. Preliminaries

Our data-sharing scheme comprises of 3 entities: ring members, secure cloud, and blockchain network. These entities can be identified as follows:

- *Ring members* are the data owners and the receivers. Note that members represent individuals in the same or different institutions.
- *Secure cloud* is the place we store our encrypted data. With using pre-defined algorithms traceability and linkability checks are done. Also, all shares in the blockchain are made from the cloud.
- *Blockchain network* is where the cloud shared the related information to reach out the stored. A snapshot of the ledger is available to all users whenever they want to access it.

For convenience, we first introduce the underlying cryptographic sub-protocols and then explain the overall scheme.

The traceable ring signature we used in our proposal is proposed by Fujisaki and Suzuki [21]. Fujisaki and Suzuki presented the idea of a traceable ring signature to trace the signature if it is signed by the same user. In the scheme, there is a tag value L for every signature, including the ring members' public keys and an *issue* that stands for an election, survey, etc. The signer signs with his/her secret key and tag L. There are three options for the output of the trace function (Algorithm 4):

- *Linked:* If the signer uses the same tag to sign two same messages.

- *Revealing of signer's public key:* If the signer uses the same tag to sign two different messages.

- *Independent:* If the signer uses different tags to sign different messages.

The key encapsulation mechanism (KEM) we used in our proposal is presented by Smart [22]. It is an efficient key encapsulation mechanism between multiple parties and so-called mKEM. Using mKEM-encapsuation (Algorithm 2), the user can obtain a common key K and encapsulated value C by taking public keys of receivers as inputs and use this key K to encrypt a message. In order to decrypt this message, receivers need to reach the common key K. For this purpose, each receiver uses the decapsulation algorithm (Algorithm 6) with inputs C and his/her secret key.

In our scheme, G is a multiplicative group of order q with generator g. $H : \{0,1\}^* \to G$, $H' : \{0,1\}^* \to G$ and $H'' : \{0,1\}^* \to \mathbb{Z}_q$ are hash functions. Notations are given in Table 2.

Table 2: Notations

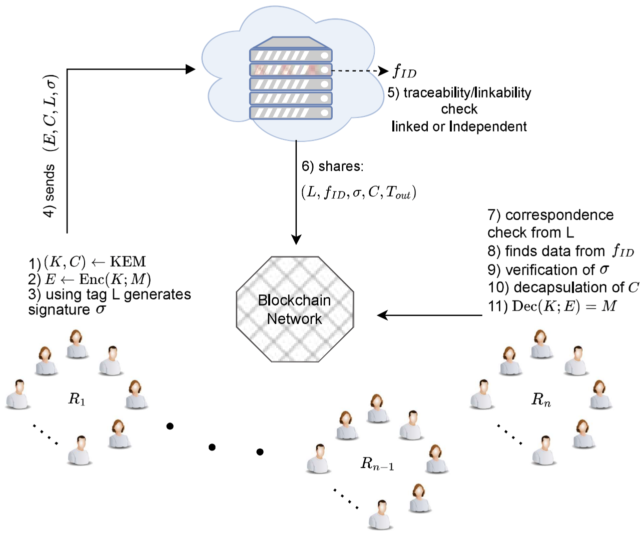
Notation	*Definition*
q	prime number
g	generator of G
N	$(1,\ldots,n)$
$* \xleftarrow{\$} S$	pick a random value * from the set S
KDF	key derivation function
(sk_i, pk_i)	the key pair of i'th user
pk_N	$\{pk_1, pk_2 \ldots, pk_n\}$
M	message (data)
L	tag, i.e., $(issue \mid\mid pk_N)$
f_{ID}	file id
T_{out}	output of traceability check algorithm

4. Proposed scheme

The overall workflow of our data sharing scheme is demonstrated in Figure 1. Every user picks a random value sk over $\mathbb{F}_q^*$ as a secret key and computes his public key $pk = g^{sk}$ by using Algorithm 1. (sk_i, pk_i) is the key pair of i'th user.

Algorithm 1 Key Generation

```
Input: $g \in G$
1: $sk \xleftarrow{\$} \mathbb{F}_q^*$
2: $pk = g^{sk}$
3: return $(sk, pk)$
```



4) sends (E, C, L, σ)
f_{ID}
5) traceability/linkability check linked or Independent
6) shares: $(L, f_{ID}, \sigma, C, T_{out})$
1) $(K, C) \leftarrow \mathrm{KEM}$
2) $E \leftarrow \mathrm{Enc}(K; M)$
3) using tag L generates signature σ
Blockchain Network
7) correspondence check from L
8) finds data from f_{ID}
9) verification of σ
10) decapsulation of C
11) $\mathrm{Dec}(K; E) = M$
R_1
R_{n-1}
R_n

Figure 1: Overall workflow of proposed architecture

Suppose one of the members of a ring, say R_1, wants to share secret information with some members of the other ring, say R_* where $* \in \{1,2,\ldots,n\}$.

1. The i'th user of the ring, say user A, calls Algorithm 2 to encapsulate the keys of multiple receivers. This algorithm takes the public keys of $t-1$ receivers and the public key of user A as input for our case. The results of the algorithm are encapsulated key K and the value C. K is the common key for symmetric encryption, and receivers can use the value C for computing the common key.

Algorithm 2 Encapsulation

```
Input: $\{pk_1, \ldots, pk_t\}$
1: $k \xleftarrow{\$} G$
2: $r = H(k)$
3: $c_0 = g^r$
4: for $i = 1$ to $t$ do
5:     $c_i = k \cdot pk_i^r$
6: end for
7: $K \leftarrow KDF(k)$
8: $C \leftarrow (c_0, c_1, \ldots, c_t)$
9: return $(K, C)$
```

2. Using symmetric encryption key K, the data owner encrypts the corresponding data M as $E = \mathrm{Enc}(K; M)$.

3. For our case, tag L is important because it simply covers the relation of the secret data and receivers. For example, if the encrypted form is related to a breast cancer issue set as this information. The tag is set as $L = (cancer \mid\mid pk_N)$, where $N = (1, \ldots, n)$, and pk_N is the list of public keys of R_1, i.e, ring including the data sender. After that, user A generates ring signature σ on (L, E) via Algorithm 3.

Algorithm 3 Signature generation

```
Input: $(E, pk_N, L)$
1: $h = H(L)$, $\sigma_i = h^{sk_i}$
2: $A_0 = H'(L,E)$ and $A_1 = (\sigma_i/A_0)^{1/i}$
3: for $j = 1$ to $n$ do
4:     if $j \neq i$ then
5:         $\sigma_j = A_0 A_1^j \in G$
6:     end if
7: end for
8: $w_i \xleftarrow{\$} \mathbb{Z}_q$, $a_i = g^{w_i}$, $b_i = h^{w_i} \in G$
9: for $j = 1$ to $n$ do
10:    if $j \neq i$ then
11:        $z_j, c_j \xleftarrow{\$} \mathbb{Z}_q$
12:        $a_j = g^{z_j} pk_i^{c_j}$
13:        $b_j = h^{z_j} \sigma_j^{c_j} \in G$
14:    end if
15: end for
16: $c = H''(L, A_0, A_1, a_N, b_N)$
17: $c_i = c - \sum_{j \neq i} c_j \mod q$, $z_i = w_i - c_i sk_i \mod q$
18: return $\sigma = (A_1, c_N, z_N)$ on $(L, E)$
```

4. User A sends the data file $F = (E, C, L, \sigma)$ to the secure cloud server.

5. The file F is temporarily stored by the server as it arrives. Before storing it permanently, a traceability check is done between the received pair (E, σ) and every pair (E', σ') that has been stored before with the same tag L. The output of the traceability check (Algorithm 4) can be pk_i (revealing of user's public key), linked, or independent.

   If the output is *linked* where the input is (E, σ) and any pair (E', σ'), the server removes the request. If the algorithm returns pk_i for the input: (E, σ) and any pair (E', σ') or "*indep*" for every input, i.e., (E, σ) and each pair (E', σ'), then the server stores the file F permanently.

6. If the returned value indicates independence, then it is a valid request. Thus the server stores the file in its permanent location. Then cloud generates a transaction including $(L, f_{ID}, \sigma, C, T_{out})$.

Algorithm 4 Traceability check

```
Input: $(E, \sigma)$ and $(E', \sigma')$
1: Parse $L$ as $L = (cancer \mid\mid pk_N)$.
2: for all $i$ do
3:     $A_0 = H'(L, E)$, $\sigma_i = A_0 A_1^i$
4: end for
5: for all $i$ do
6:     $A'_0 = H'(L, E')$, $\sigma'_i = A'_0 {A'_1}^i$
7: end for
8: for all $i$ do
9:     if $\sigma_i = \sigma'_i$ then
10:        Store $pk_i$ in $TList$
11:    end if
12: end for
13: if $TList = \{pk_i\}$ then
14:    return $pk_i$
15: end if
16: if $TList = pk_N$ then
17:    return "$linked$"
18: end if
19: if $1 < \#TList < n$ then
20:    return "$indep$"
21: end if
```

7. Users from other rings can learn whether the appended transaction is relevant to them from the issue part of the tag L, with a simple ledger check. For instance, considering an institute collecting data of breast cancer from user forms, expecting user forms, from the *bcancer* tag, may understand that this transaction is related to them.

8. Therefore, user from other ring R_*, say B, reads the file F located at f_{ID} shared in the same transaction. After that B has $(L, f_{ID}, \sigma, C, T_{out})$.

9. User B verifies signature by using Algorithm 5 with the values σ, E, L. If the signature σ is valid, then user continuous with next step.

10. To be able to decrypt the ciphertext E, authorized users need to have access to the secret symmetric key by key decapsulation. The key decapsulation algorithm (Algorithm 6) takes C and sk_x, i.e., secret key of user B, as inputs and outputs encapsulated key K where $x \in \{1, \ldots, t\}$.

11. Since only the authorized users by the sender can only get the symmetric key K, only they can de-

Algorithm 5 Signature verification

```
Input: $\sigma = (A_1, c_N, z_N), pk_N, E, L$
 1: Parse $L$ as $(cancer \,||\, pk_N)$
 2: for all $i$ do
 3:     check $g, A_1 \in G$, $c_i, z_i \in \mathbb{Z}_q$, $pk_i \in G$
 4: end for
 5: $h = H(L), A_0 = H'(L, E)$
 6: for all $i$ do
 7:     $\sigma_i = A_0 A_1^i \in G$
 8:     $a_i = g^{z_i} pk_i^{c_i}$
 9:     $b_i = h^{z_i} \sigma_i^{c_i}$
10: end for
11: if $H''(L, A_0, A_1, a_N, b_N) = \sum_{i \in N} c_i \mod q$ then
12:     return "Valid signature"
13: end if
```

Algorithm 6 Decapsulation

```
Input: $(C, sk_x)$
 1: Parse $C$ as $(c_0, c_1, \ldots, c_t)$
 2: $k = c_x / c_0^{sk_x}$
 3: $r = H(k)$
 4: If $c_0 \neq g^r$, return $\perp$ and halt
 5: $K \leftarrow KDF(k)$
 6: return $K$
```

crypt the ciphertext. User B receives the hidden message as $\mathrm{Dec}(K; E) = M$ using the key K.

5. Security analysis

In this section, we first define the security features that our framework achieves. Then as a subsection, give the thread model via defining a security game for security analysis. In the last subsection, we prove the security of our scheme by applying the formal security game.

5.1. Security Requirements

We will discuss the security features that need to be obtained in our proposed scheme.

- Confidentiality: Public keys of the potential receivers are encapsulation using Algorithm 2. Resulted K is used as a symmetric key to encrypt data M to achieve confidentiality. Since we restricted accessibility to K to achieve confidentiality, no malicious user will ever be able to access the encapsulated symmetric key.
- Integrity: Since the encrypted data is stored on a cloud server using a symmetric key and then signed via traceable ring signature, the data owner could decrypt it using the corresponding secret encryption key to check integrity at any time.
- Anonymity: Our proposed scheme satisfies user anonymity, i.e., it is difficult to reveal the identity of the data sharers and the receivers from used sub-protocols.
- Traceability: Since we used a traceable ring signature as a sub-ptorocol, public keys of the misbehaviored users are released and published on the blockchain network.
- Authentication: Through the key encapsulation mechanism we used as a sub-protocol, users are required to prove their identity as a prerequisite to being allowed access to resources in an information system. So that the authentication is achieved.

5.2. Security Model

The definition of underlying hard problem of our scheme in this paper is discrete logarithm assumption (DLA) is given below.

Definition 1. ***Discrete Logarithm Assumption (DLA):*** *Let $\mathbb{G}$ be a cyclic group of prime order p with generator g. The Discrete Logarithm Assumption states that for any efficient algorithm $\mathscr{A}$, the probability of $\mathscr{A}$ computing x given $y = g^x$ is negligible, denoted as* $\mathrm{DLA}_{\mathscr{A}}$*:*

$$\mathrm{DLA}_{\mathscr{A}} \overset{def}{=} \Pr[\mathscr{A}(y) = x \,|\, y = g^x] \leq \mathrm{negl}(\lambda)$$

where λ is the security parameter.

The security is based on the indistinguishability against secret encapsulated symmetric key and chosen-plaintext attack (CPA), $IND-CPA$. To this aim, we define a security game, which has 4 stages. The game is run between the challenger $\mathscr{C}$ and the adversary $\mathscr{A}$.

1. *Setup phase:* Challenger runs the algorithm related to set up and obtains the public parameters pp, and gives pp to the adversary. Also, chooses a random coin $c \in \{0, 1\}$, and keeps c secret.
2. *Find phase:* The adversary makes the following queries. Note that $\mathscr{A}$ is not authorized to choose random integers in a way that enabling trivial decapsulation of the m or trivial signature verification.

Encapsulation query $Q_{enc} = (pk_N, m, i)$ $\mathscr{A}$ chooses an identity index i of a target receiver and a message m from message space. Sends these values to $\mathscr{C}$. Upon receiving values, $\mathscr{C}$ retrieves the corresponding pk_i. Runs the Algorithm 2, computes (K, C), and returns C. Otherwise, forces $\mathscr{A}$ to choose a random coin $\hat{c} \leftarrow \{0, 1\}$.

As the result of *find phase* $\mathscr{A}$ selects a pk_{i^*} and m_0, m_1.

3. *Challenge phase:* $\mathscr{C}$ computes the following query and submits it to the $\mathscr{A}$.

   Challenge query for encapsulation Upon receiving the $(choice, pk_{i^*}, m_0, m_1)$ from $\mathscr{A}$, $\mathscr{C}$ computes the following query:

   $Q_{ch} = (pk_N, m_*, i^*)$ $\mathscr{C}$ checks that:

   - if $pk_{i^*} \notin pk_N$, $\mathscr{C}$ selects randomly generated values (K^*, C^*), and returns C^*.
   - if $pk_{i^*} \in pk_N$, calls Algorithm 2 for m_c computes (K^*, C^*), and returns C^*.
   - otherwise, forces $\mathscr{A}$ to output a random coin $c \leftarrow \{0, 1\}$. If $c = 0$, computes (K^*, C^*),and returns C^*, otherwise selects randomly generated values as (K^*, C^*), and returns C^*.

4. *Guess phase:* The adversary repeats the Encapsulation query. At the end of this phase, $\mathscr{A}$ provides c^*. $\mathscr{A}$ wins the game in the case of $c^* = c$. Suppose Q_{enc} and Q^*_{enc} queries in find and guess phases. We define the adversary's advantage as $Adv^A_{Game1}(\lambda) = |Pb[c = c^*] - 1/2|$. We say that the scheme is IND-CPA secure if $Adv_{Game1} \leq \text{negl}(\lambda)$ for all p.p.t algorithms $\mathscr{A}$.

5.3. Security Proof

Theorem 1. *The proposed system is IND-CPA secure in the random oracle model under the DDH and DLog assumptions.*

Proof. In Figure 2, we depicted the interaction between the adversary and the challenger. $\mathscr{A}$ is considered as p.p.t algorithm with non-negligible advantage ε in $e^{IND-CPA}$. $\mathscr{A}$ is engaged to define another algorithm $\mathscr{C}$ having a non-negligible advantage in solving DLA. We show the interaction between $\mathscr{A}$ and $\mathscr{C}$ as follows.

The random oracle $G \leftarrow H, H' : \{0, 1\}^*$ is simulated by $\mathscr{C}$ as follows: When an encapsulation query is received, a random number $\mu \leftarrow Z^*_p$ is selected, and a random coin flipped $\nu \rightarrow 1$, with probability $\mathscr{X}$. Otherwise, $\nu \rightarrow 0$, r setted as g^μ, otherwise setted as $(g^x)^\mu$.

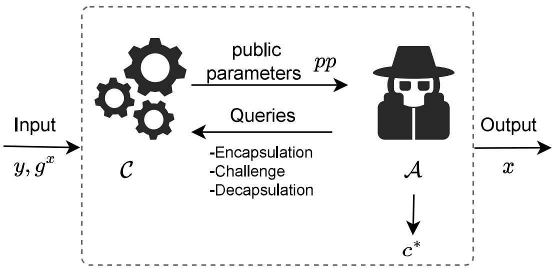


Figure 2: IND-CPA security diagram

Note that resulted r has random distribution. $\mathscr{C}$ continues to simulate the random oracle H', H'' returning random elements in G and $\mathbb{Z}_q$, respectively.

1. *Setup phase:* $\mathscr{A}$ is given $pp = (G, H', H'', g, g^x)$ by $\mathscr{C}$. Chooses a random coin $c \in \{0, 1\}$, and keeps it secret.

2. *Find phase:* $\mathscr{A}$ sends the encapsulation query $Q_{enc} = (pk_N, m, i)$. After receiving Q_{enc}, $\mathscr{C}$ evaluates pk_i, calls Algorithm 2 and computes (K, C), and sends the corresponding C. Later, $\mathscr{A}$ sends the query $Q_{sign} = (E, pk_N, L)$ for a pre-determined tag L. As the result of Q_{sign}, $\mathscr{C}$ returns σ.

3. *Challenge phase:* $\mathscr{A}$ outputs pk_{i^*}, m_0, m_1, where the public key pk_{i^*} corresponds to which user is not trivial. $\mathscr{C}$ selects $c \leftarrow (0, 1)$. Then encapsulation $(K^*, C^*) = (KEM, (c_j))$, where K is the symmetric key used to encrypt m_c, is sent to $\mathscr{A}$.

4. *Guess phase:* $\mathscr{A}$ sends encapsulation queries same as the find phase. $\mathscr{A}$ outputs its guess $c^* \in (0, 1)$.

Note that in this game, it is infeasible for $\mathscr{A}$ to distinguish simulation because $\mathscr{A}$ cannot recognize the randomly generated C^*), and a result of that cannot distinguish the K^*. Therefore, the only chance of $\mathscr{C}$ is to choose a random coin c^*. At the end of this game $\mathscr{A}$ outputs a guess coin c^*. By definition, the probability of the $\mathscr{A}$ to win the game is:

$$Adv^A_{IND-CPA}(\lambda) = |Pb[c = c^* - 1/2]| = \varepsilon, \qquad (1)$$

if m_c is correctly encrypted using symmetric key k, and (K^*, C^*) is a correctly formed encapsulation of k. Consequently, $\mathscr{A}$'s probability to select a c^* s.t. $c = c^*$ is negligable.

6. Complexity analysis

The complexity of our proposed data sharing scheme is based on modular exponentiations, and the scheme does not contain any expensive pairing operations. So, we count the number of modular exponentiations in the encapsulation (Encap), signature generation (SignGen), traceability check (Trace), signature verification (SignVer), and decapsulation (Decap). The costs of symmetric encryption and decryption are neglected.

Table 3: The cost of algorithms in our proposed scheme

Algorithm	*Cost*
Encap	$(t+1)t_e + t_H$
SignGen	$(5n-1)t_e + t_H + t_{H'} + t_{H''}$
Trace	$(2n)t_e + (2n)t_{H'}$
SignVer	$(5n)t_e + t_H + t_{H'} + t_{H''}$
Decap	$t_e + t_H$

Let t_e be the cost of modular exponentiation, t_H, $t_{H'}$, $t_{H''}$ be the cost of hash functions $H : \{0,1\}^* \to G$, $H' : \{0,1\}^* \to G$, $H'' : \{0,1\}^* \to \mathbb{Z}_q$, respectively, and finally t_h be the total cost of all hash functions. n is the number of parties in the ring of the signer, and t is the number of parties with whom the signer is willing to share the data. Table 3 shows the cost of algorithms in our proposed scheme. We have linear complexity $O(n)$. We have tabulated the comparison of two closely related studies in Table 4.

Table 4: Complexity comparisons of known constructions

	Our Scheme	Huang et al.	Agyekum et al.
SignGen + Enc + Encap	$(5n+t)t_e + 2t_H + t_{H'} + t_{H''}$	$12t_e + t_{H_1}$	$t_e + t_G + t_p + 2t_s$
Trace	$(2n)t_e + (2n)t_{H'}$	t_e	-
SignVer + Dec + Decap	$(5n+1)t_e + 2t_H + t_{H'} + t_{H''}$	$11t_e + t_{H_1}$	$3t_G + 2t_s$

In the scheme of Huang et al. [9], the cost of signature generation, signature verification, and tracing are $12t_e + t_{H_1}$, $11t_e + t_{H_1}$ and t_e, respectively where t_{H_1} is the cost of the hash function $H_1 : \{0,1\}^* \to \mathbb{Z}_q$. In their scheme, the user generates the signature, group manager verifies the signature and traces the user. In the proposed scheme, users generate and verify the signature, and the server traces the user. Compared to our scheme, SignGen, SignVer, and Trace algorithms need more modular exponentiation. But we eliminated the trusted third party like the manager by using the traceable ring signature and we added a key encapsulation mechanism to send data encrypted with multiple parties. Therefore we need few modular exponentiations and hash operations. Also, the cost of our scheme is not so different from that of Huang et al. for small n.

On the other hand, in the framework presented by Agyekum et al. [10], for t_G, t_p are the costs of group operation, pairing operation, the complexities of the encryption, re-encryption, decryption-1 and decryption-2 are $t_e + t_G$, t_p, t_G and $2t_G$ respectively. The data owner's and proxy server's signatures are checked for authenticity checks. Since they have not specified the signature method to be used, we denote the cost of the related signature as t_s. Note that traceability checks can be conducted partially by checking timestamped block. Thus, in Table 4, the cost of it is not included. In this scheme, they preferred to use proxy re-encryption method to share data which is different than our approach. While our scheme is many-to-many, this is one-to-one scheme.

7. Performance analysis

We have implemented our data-sharing protocol in Python using the cryptographic features of SageMath. The group has been defined over the curve secp256k1. AES256 CBC mode with a random initialization vector of size *16*-byte is used for encryption. We implemented SHA256 for our hash functions and obtained the results on the curve or field. The codes are available as open-source on GitHub[1].

We tested the implementation for 256 signers and ring sizes $8, 16, 32, 64$, and 128, and gathered the profiling and bench results for all the single operations, block generation, and signature generation. Tests are run on macOS Ventura 13.4 working on an Apple M1 Pro machine with 16 GB of RAM. The following figures provide the time spent (minimum, average, and maximum) in CPU seconds for the single operations and overall flows for the ring size $2^3, \dots, 2^7$.

Figure 3 gives the overall running time of block generation where it implements signature file generation, traceability check, and block generation (if the number of transactions is enough to generate a block.). The overall running time to verify a signature that covers finding the related signatures from the blockchain, signature verification, decapsulation, and decryption is given in Figure 4.

[1]See, `https://github.com/curiecrypt/many2many_decentralized_anonymous_data_sharing.git`.

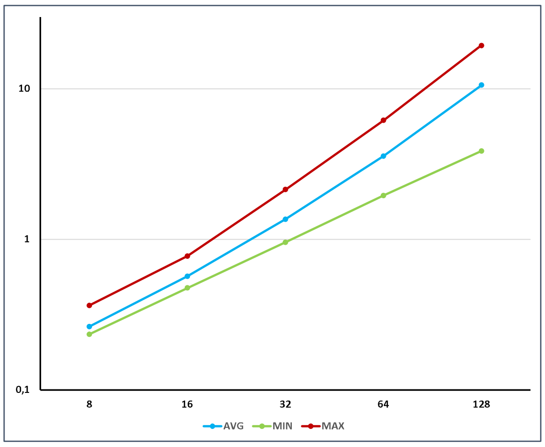


Figure 3: Time spent in CPU seconds for block generation.

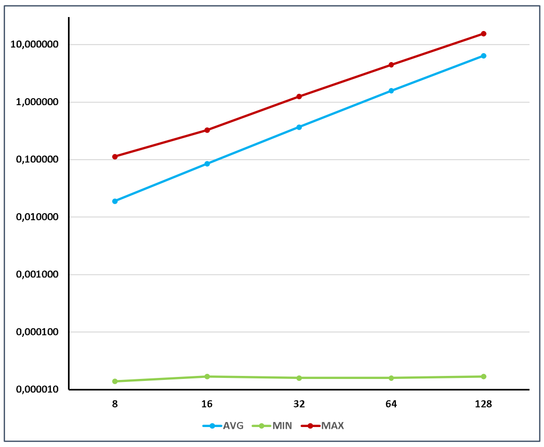


Figure 5: Time spent in CPU seconds for traceability check.

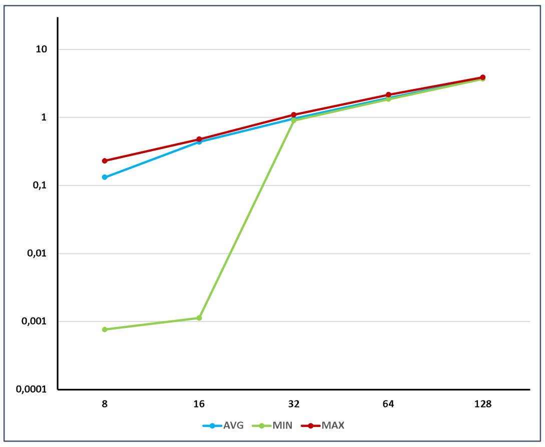


Figure 4: Time spent in CPU seconds for verification.

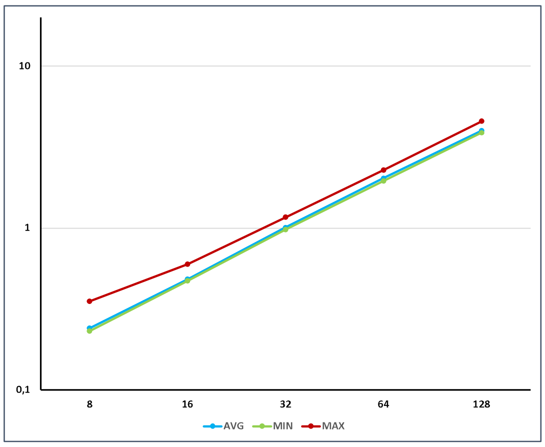


Figure 6: Time spent in CPU seconds for signature generation.

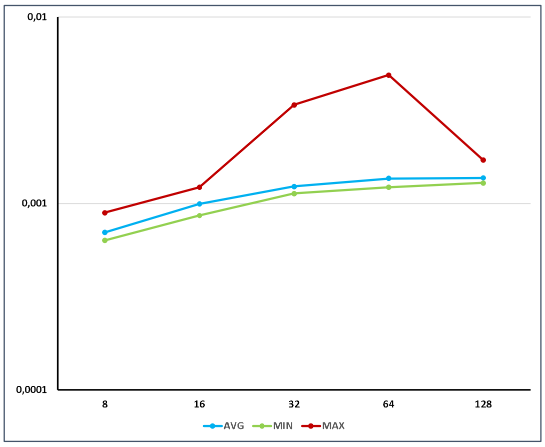


Figure 7: Time spent in CPU seconds for collecting related signatures for verification.

The time spent in CPU seconds for traceability check, signature file generation, collecting related signatures for verification, and verification of a signature (including decapsulation and decryption) are given in Figures 5, 6, 7, and 8, respectively.

Authorship Contribution Statement

Esra Günsay: Defined the problem, found the related subprotocols, and constructed the proposed scheme. Contributed to the complexity analysis. Literature Review. Security Analysis. Writing & Editing.

Burcu E. Karakaş: Found the suitable KEM, and contributed to the protocol construction process. Complexity Analysis. Contributed to the performance analysis. Writing & Editing.

N. Gamze Orhon Kılıç: Implemented the protocol. Performance Analysis. Writing.

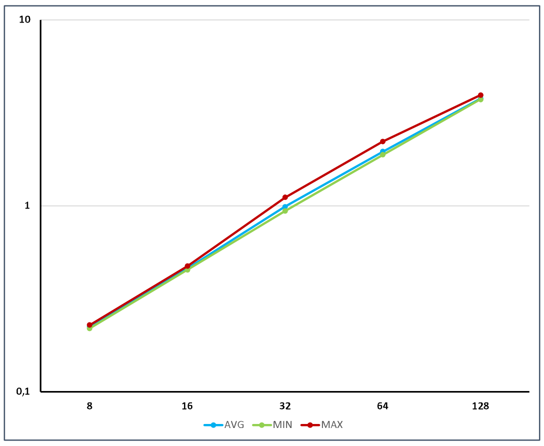


Figure 8: Time spent in CPU seconds for verification of a signature.

Oğuz Yayla: Offered to use a KEM, and contributed to the protocol construction process. Supervision.

References

[1] Y. Yi, N. Zhu, J. He, A. D. Jurcut, X. Ma, Y. Luo, A privacy-dependent condition-based privacy-preserving information sharing scheme in online social networks, Computer Communications 200 (2023) 149–160. `doi:https://doi.org/10.1016/j.comcom.2023.01.010`.
URL `https://www.sciencedirect.com/science/article/pii/S014036642300018X`

[2] Z. Zhu, R. Jiang, A secure anti-collusion data sharing scheme for dynamic groups in the cloud, IEEE Transactions on Parallel and Distributed Systems 27 (1) (2016) 40–50. `doi:10.1109/TPDS.2015.2388446`.

[3] Y. Liu, B. D. Cruz, E. Tilevich, Trusted and privacy-preserving sensor data onloading, Computer Communications 206 (2023) 133–151. `doi:https://doi.org/10.1016/j.comcom.2023.04.027`.
URL `https://www.sciencedirect.com/science/article/pii/S0140366423001421`

[4] X. Qin, Y. Huang, Z. Yang, X. Li, A blockchain-based access control scheme with multiple attribute authorities for secure cloud data sharing, Journal of Systems Architecture 112 (2021) 101854.

[5] S. Saha, O. Landsiedel, M. C. Chan, Efficient many-to-many data sharing using synchronous transmission and tdma, in: 2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS), 2017, pp. 19–26. `doi:10.1109/DCOSS.2017.11`.

[6] S. M. Farooq, S. S. Hussain, T. S. Ustun, A survey of authentication techniques in vehicular ad-hoc networks, IEEE Intelligent Transportation Systems Magazine 13 (2) (2020) 39–52.

[7] G. Yang, D. S. Wong, X. Deng, H. Wang, Anonymous signature schemes, in: M. Yung, Y. Dodis, A. Kiayias, T. Malkin (Eds.), Public Key Cryptography - PKC 2006, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 347–363.

[8] A. Naghib, N. Jafari Navimipour, M. Hosseinzadeh, A. Sharifi, A comprehensive and systematic literature review on the big data management techniques in the internet of things, Wireless Networks (2022) 1–60.

[9] H. Huang, X. Chen, J. Wang, Blockchain-based multiple groups data sharing with anonymity and traceability, Science China Information Sciences 63 (2020) 1–13.

[10] K. O.-B. O. Agyekum, Q. Xia, E. B. Sifah, C. N. A. Cobblah, H. Xia, J. Gao, A proxy re-encryption approach to secure data sharing in the internet of things based on blockchain, IEEE Systems Journal 16 (1) (2022) 1685–1696. `doi:10.1109/JSYST.2021.3076759`.

[11] X. Hu, X. Song, G. Cheng, H. Wu, J. Gong, Efficient sharing of privacy-preserving sensing data on consortium blockchain via group key agreement, Computer Communications 194 (2022) 44–54. `doi:https://doi.org/10.1016/j.comcom.2022.07.035`.
URL `https://www.sciencedirect.com/science/article/pii/S0140366422002833`

[12] J. Song, Y. Yang, J. Mei, G. Zhou, W. Qiu, Y. Wang, L. Xu, Y. Liu, J. Jiang, Z. Chu, et al., Proxy re-encryption-based traceability and sharing mechanism of the power material data in blockchain environment, Energies 15 (7) (2022) 2570.

[13] H. Liu, H. Huang, Y. Zhou, Q. Chen, Improvement of blockchain-based multi-layer location data sharing scheme for internet of things, Computer Communications 201 (2023) 131–142. `doi:https://doi.org/10.1016/j.comcom.2023.02.005`.
URL `https://www.sciencedirect.com/science/article/pii/S0140366423000427`

[14] J. Xu, K. Xue, S. Li, H. Tian, J. Hong, P. Hong, N. Yu, Healthchain: A blockchain-based privacy preserving scheme for large-scale health data, IEEE Internet of Things Journal 6 (5) (2019) 8770–8781.

[15] S. Wang, K. Liang, J. K. Liu, J. Chen, J. Yu, W. Xie, Attribute-based data sharing scheme revisited in cloud computing, IEEE Transactions on Information Forensics and Security 11 (8) (2016) 1661–1673. `doi:10.1109/TIFS.2016.2549004`.

[16] W. Tang, J. Ren, K. Zhang, D. Zhang, Y. Zhang, X. S. Shen, Efficient and privacy-preserving fog-assisted health data sharing scheme, ACM Trans. Intell. Syst. Technol. 10 (6) (oct 2019). `doi:10.1145/3341104`.
URL `https://doi.org/10.1145/3341104`

[17] J. Wang, X. Chen, X. Huang, I. You, Y. Xiang, Verifiable auditing for outsourced database in cloud computing, IEEE Transactions on Computers 64 (11) (2015) 3293–3303. `doi:10.1109/TC.2015.2401036`.

[18] X. Cheng, F. Chen, D. Xie, H. Sun, C. Huang, Design of a secure medical data sharing scheme based on blockchain, Journal of medical systems 44 (2) (2020) 52.

[19] L. Yin, J. Feng, S. Lin, Z. Cao, Z. Sun, A blockchain-based collaborative training method for multi-party data sharing, Computer Communications 173 (2021) 70–78. `doi:https://doi.org/10.1016/j.comcom.2021.03.027`.

[20] D. Boneh, B. Bünz, B. Fisch, Batching techniques for accumulators with applications to iops and stateless blockchains, in: A. Boldyreva, D. Micciancio (Eds.), Advances in Cryptology – CRYPTO 2019, Springer International Publishing, Cham, 2019, pp. 561–586.

[21] E. Fujisaki, K. Suzuki, Traceable ring signature, in: International Workshop on Public Key Cryptography, Springer, 2007, pp. 181–200.

[22] N. P. Smart, Efficient key encapsulation to multiple parties, in: International Conference on Security in Communication Networks, Springer, 2004, pp. 208–219.