Regular Paper

Communications of the CCISA
Vol. 25 No. 3 Aug. 2019

Smart Contract-based Decentralized Privacy System for Securing Data Ownership Management*

Yunmin He[1,2], Yu-Chi Chen[1*], Zhong-Yi Guo[1], Raylin Tso[3], and Shaozhen Ye[2]

[1] Department of Computer Science and Engineering, Yuan Ze University, Taiwan

[2] College of Mathematics and Computer Science, Fuzhou University, China

[3] Department of Computer Science, National Chengchi University, Taiwan

* wycchen@saturn.yzu.edu.tw

Abstract

Recently, Zyskind et al. proposed a decentralized personal data management system which keeps privacy through blockchain and off-blockchain storage, so-called the decentralized privacy (DP) system. This system helps users ensure data ownership and fine-grained access control for third-party service providers. However, in this DP system, the permission power is delegated to blockchain and the users' data are stored in the off-blockchain distributed hashtable. Therefore, this induces extra communication overhead to connect these two distinct functionalities. In this paper, we present a conceptually simple solution directly from smart contracts with cryptographic primitives. This system is called the smart contract-based decentralized privacy (SCDP) system to overcome the above-mentioned efficiency issues. We propose the basic SCDP system as a warm-up to introduce the design principle based on symmetric encryption. Moreover, the strong SCDP system is provided by using ciphertext-policy attribute-based encryption to support more flexible scenarios of access control and also eliminate some limitations of the basic system. Finally, we discuss some analyses in the aspects of security, access control, and data segmentation.

Keywords: Decentralized privacy, Smart contracts, Encryption, Ownership, Access control.

* A preliminary version appeared in GCCE 2019.

1. Introduction

In the Bitcoin White Paper [15] released in 2008, Satoshi Nakamoto first proposed the concept of blockchain technology and created the blockchain network the following year. As the core foundation of Bitcoin, blockchain has attracted widespread attentions from financial and academic fields. Generally speaking, the blockchain can be viewed as a distributed ledger which consists of concatenated blocks for recording all the transactions that have occurred in the peer-to-peer network. In the past decade, numerous researchers have found that it cannot only be used as a decentralized payment system, but also be used in multiple fields such as asset registration, supply chain traceability, medical care and identity management.

In fact, there is no central authority in the blockchain based on peer-to-peer network, and all participating nodes hold the same duplicate of database. The most critical function of the blockchain is to enable secure transactions and communications between untrusted participants without a trusted third party. In the typical setting, the blockchain is essentially composed of a series of data blocks chronologically generated and linked. And as shown in Figure 1, each block contains the transaction data (hash value), the time stamp, the target difficulty of current block, and the transaction data (hash value) of the previous block. Once the block is created and added into the blockchain, transactions in all blocks cannot be tampered with or restored, which ensures the security and integrity of the transactions and prevents double-spending problem.

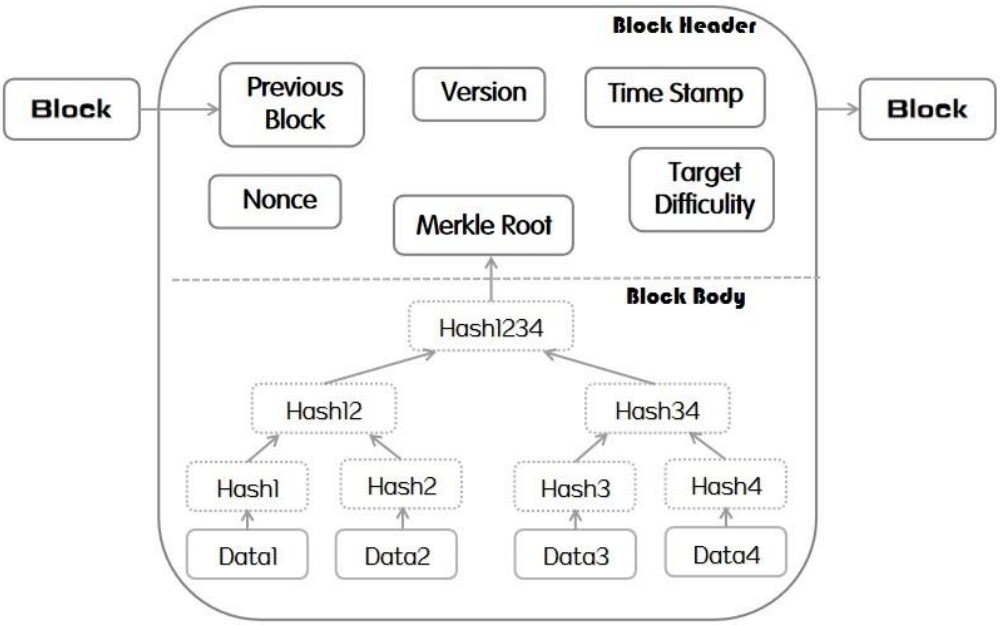

Figure 1: Details of each block.

The maintenance of the whole functionality in the blockchain and the consistency of the node replicas are all guaranteed by the introduced distributed consensus mechanism. Proof-of-Work (PoW), Proof-of-Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT) are widely used consensus mechanisms in various blockchain platforms such as Ethereum [2, 16] and Hyperledger.

Ethereum presented by Vitalik Buterin [16] is the representative of the second-generation blockchain [5], which supports the deployments of complex distributed applications beyond cryptocurrencies and customized smart contracts with the help of Turing-complete scripting language. The Ethereum account consists of a contract account and an externally owned account (EOA) [6], and each account is equipped with a corresponding private key which can control the account. In Ethereum, the operation upon the Ethereum virtual machine (EVM) consumes a certain amount of Ether (ETH, Ethereum's currency) and gas which depending on the complexity of the internal structure. Currently, Ethereum is the most promising and popular blockchain platform for developing smart contracts.

Cryptocurrency. As we all know, the emergence of Bitcoin [15] based on blockchain technology and peer-to-peer networks in the last decade indicates the coming of cryptocurrency. All transactions on Bitcoin are public and do not involve intermediaries, and accordingly there is essentially no transaction fee. Thus Bitcoin has received great attention and popularity from users in cross-border payment and other application scenarios. Since its introduction in 2009, its price has seen meteoric rise and peaked at around 20,000 USD per unit in December 2017. In addition, Bitcoin is now recognized as a legitimate commodity or even currency in many countries. The success of Bitcoin has spawned a host of cryptocurrencies, such as Litecoin, Namecoin [10] and Peercoin [14]. Most of their operating mechanisms are similar to Bitcoin, but with minor modifications. In short, these cryptocurrencies are key applications of blockchain in the financial sector.

Decentralized Privacy. In the context of serious data privacy issues and the rise of blockchain technology, Zyskind et al. [3] proposed a decentralized privacy (DP) system. This is a decentralized personal data management system based on blockchain to ensure user data privacy and data ownership. It converts the blockchain through protocols into a secure automatic access control manager without trusted third-parties. However, as we mentioned in Chapter 1, the DP system implement access control over services through the blockchain, while the storages and transmissions of data in this system involve the off-blockchain distributed hashtable. Therefore, under these operating protocols, additional communication overhead and efficiency issues are revealed. Thus in this paper, we propose the SCDP system motivated by

these concerns.

1.1 Contributions

Motivation. The motivation of this paper is to overcome the efficiency and overhead issues in the DP system described above. Our main results are conceptually simple solutions which use smart contracts (SC) to implement access control and data storage functions in the DP system. Therefore, in this paper, we propose a smart contract-based decentralized privacy (SCDP) system to overcome the above-mentioned issues. In our design, we also add series of cryptography techniques, including symmetric encryption and attribute-based encryption to ensure that this smart contract-based system enables more secure and confidential personal data protection. Therefore, users of this system can own and control their personal data as much as possible without any privacy concerns. Furthermore, our system holds the following benefits:

- Management. Users in this system can add or remove the third-party services at any time. A user can delegate or revoke a set of permissions to services as needed.
- Transparency. All operations executed on the smart contract are completely recorded, and can be viewed and checked by any entities. Each user owns transparency over the collection and usage of his personal data.
- Restriction. All entities in this system can only execute specific operations and cannot execute operations on behalf of other entities in the system or entirely as other entities.

Technique Highlight. In the SCDP system, the smart contract deployed on the blockchain not only enables efficient access control and data storage functions, but also maintains low costs. Therefore, users do not need to spend additional server maintenance overhead. At the same time, we also apply the traditional encryption technology to smart contract-based systems to ensure the security and confidentiality of user data on the blockchain.

Above all, we initially build the basic SCDP system based on classical symmetric encryption technology. The user's data are transferred to the smart contract after being encrypted using the shared secret key. Then the service can only provide functions accurately through decrypting the encrypted data which obtained through the smart contract with the corresponding shared key. Whereas considering more convenient and flexible application scenarios and higher data confidentiality, we reconstructed the basic SCDP system and introduced ciphertext policy attribute-based encryption (CP-ABE) [17]. In this strong SCDP system, unlike the previous basic SCDP system where multiple users co-exist, there is only one user in a single smart contract as its owner. By using CP-ABE, we can guarantee that only the services which hold the permissions of the smart contract and whose attributes satisfy the access structure can

correctly decrypt and obtain the required user data. In a nutshell, we use the smart contract to tackle access control and accordingly achieve secure data ownership in the strong SCDP system, and moreover apply CP-ABE to achieve confidentiality.

1.2 Organization

The remainder of this paper is organized as follows. Section 2 briefly reviews the background techniques of the SCDP system. After that, Section 3 and Section 4 present the basic SCDP system and strong SCDP system separately. Then, the specific analysis are presented in Section 5. Finally, conclusions for this paper are given in Section 6.

2 Background Techniques

2.1 Smart Contract

The concept of smart contract (SC) was originally proposed by Nick Szabo in 1994 [11], whereas this idea did not gain enough attention until the emergence of blockchain. The smart contract can be thought of as automatically executed scripting system written in Turing-complete programming language. It can automatically execute the terms of protocols and generate corresponding evidence once the specified conditions are satisfied, without the intervention of third-party. Therefore, compared with traditional transaction systems which require trusted third-party supervisions to enforce agreements, smart contracts offer more efficient implementations and lower transaction costs. The input and output of the smart contract can include currency, and the transactions which ultimately completed are also irreversible and traceable. In addition to the most usual areas of finance, smart contracts can also be used in fields such as supply chain traceability [9], crowdfunding [18] and voting privacy [12].

As shown in the system described in Figure 2, a smart contract consists of executable code, a private storage, and an account balance [8]. The state of the contract that is stored on the blockchain comprises its balance and storage, and it update with each invocation. After being deployed at an unique address (SC.address) on the blockchain, the code of each smart contract cannot be changed. Users can simply send a transaction to the corresponding smart contract's address when they intend to run the smart contract. The smart contract can execute corresponding operations based on received transactions, including storing balances, reading/writing to its private storage, and sending/receiving currency or messages from

users/other contracts.

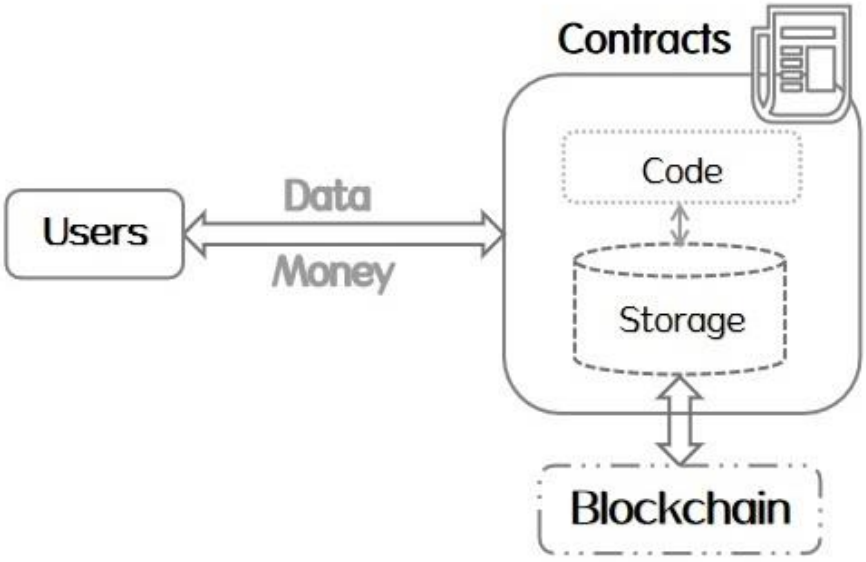

Figure 2: System of smart contract.

Security of smart contract. Based on the inherent characteristics of the blockchain and the security of Ethereum accounts, we know that the smart contract system holds the following security property.

Definition 1. The smart contract is secure if there exist a polynomial adversary A without corresponding conditions such that

$$Pr[A(SC.address) \rightarrow permissions] \leq negl(n)$$

where, $permissions$ indicates that the A can execute certain functions in SC and $negl(n)$ is a negligible function.

2.2 CP-ABE

Attribute-based encryption (ABE) was first proposed by Sahai and Waters in 2005 [1] to achieve fine-grained access control of encrypted data. Its encryption and decryption are determined by the attributes and access structures of the data and participating recipients. The concept of CP-ABE was originally proposed by Goyal et al. [17], yet the earliest architecture was first presented by Bethencourt, Sahai and Waters [4]. In CP-ABE, the encryption party can formulate a corresponding access structure for the encrypted data. And the decryption party can successfully decrypt and obtain correct data if and only if its own attribute sets satisfies the access structure for encryption. Hence CP-ABE is flexible and secure in ensuring that user

groups cannot obtain unauthorized data through collusions. The basic CP-ABE scheme consists of the following four algorithms:

- **Setup**(1^{λ}): Given only the implicit security parameter 1^{λ}, it outputs the public parameters PK and a master key MK.
- **Enc**$(PK, M, \mathcal{S})$: It takes as inputs the public parameters PK, a message M, and an access structure S. We assume that the ciphertext implicitly contains $\mathcal{S}$. Subsequently, the algorithm encrypts M and outputs the ciphertext CT.
- **KeyGen**(MK, A): It takes the master key MK and a set of attributes A which describe the key as inputs and finally generates a private key SK.
- **Dec**(PK, CT, SK)*:* It takes the public parameters PK, the ciphertext CT, and the private key SK as inputs. If the attribute sets A satisfies the access structure $\mathcal{S}$ then the algorithm outputs the original message M. Otherwise, outputs $\perp$ (null).

Correctness. Note that, we can guarantee the correctness of the final result by:

$$\mathbf{Dec}(PK, CT, SK) = \mathbf{Dec}\big(PK, \mathbf{Enc}(PK, M, \mathcal{S}), \mathbf{KeyGen}(MK, A)\big) = M$$

Security of CP-ABE. Shortly speaking, the ciphertext policy attribute-based encryption (CP-ABE) is chosen plaintext attacks (CPA) secure if all polynomial time adversaries $\mathcal{A}$ hold at most a negligible advantage in game $Exp^{CPA}_{CP-ABE,A}$ as follows:

- **Setup.** The challenger runs the **Setup** algorithm and gives the public parameters PK to the adversary.
- **Oracle 1.** The adversary makes repeated private keys corresponding to attributes sets $A_1, A_2, \dots, A_{q1}$.
- **Challenge.** The adversary submits two equal length messages m_0, m_1. In addition, the adversary gives a challenge access structure $\mathcal{S}^*$ such that none of the above sets $A_1, A_2, \dots, A_{q1}$ from Oracle 1 satisfy the access structure. The challenger randomly selects a coin b and encrypts m_b under $\mathcal{S}^*$. The resulting cipher CT^* will be given to the adversary.
- **Oracle 2.** Oracle 1 is repeated with the restriction that none of attributes sets $A_1, A_2, \dots, A_{q1}$ satisfy the access structure corresponding to this challenge.
- **Guess.** The adversary outputs a guess b' of b. It outputs 1 if $b' = b$; otherwise, outputs 0.

Formally speaking, CP-ABE is secure if for all polynomial time adversaries in the above game such that

$$Pr\big[Exp^{CPA}_{CP-ABE,A}(1^{\lambda}) = 1\big] \leq \frac{1}{2} + negl(n)$$

where, $negl(n)$ is a negligible function.

3 Warm-up: Basic SCDP System

3.1 System Model

In order to solve the efficiency and overhead issues in [3], we follow the scheme and strategy of DP system proposed by Zyskind et al., and then propose a smart contract-based decentralized privacy (SCDP) system. The basic model framework of this system is shown in Figure 3. There are three key entities in this abstract model: user, service, and smart contract.

Users provide personal data as the leader of the system and ask for the functions provided by services. Their requirements are to hold ownership and authority over their own personal data while being able to use the required applications. Services obtain the personal data of users and provide corresponding functions. Nonetheless, services' access to data is subject to user management. The smart contract (SC) aims to help realize the interactions between users and services, and execute two main types of transactions in this system: T_{access} for access management; and T_{data} for data transmission and usage. In short, the SC in this decentralized privacy system owns the following characteristics: 1) allows the system to add users (and revoke users), 2) allows users to add services (and delete services), and 3) allows users to grant access of data to service when needed (and revoke at any time). In addition, the specific functions of T_{access} and T_{data} are as follows:

T_{access}: related to the ownership over data and the management of access rights. Users manage their encrypted personal data through T_{access}, and can grant or revoke a set of permissions about their personal data to services at any time.

T_{data}: related to the upload and download of personal data that has been encrypted. Users can upload the encrypted data to the SC through T_{data}. Similarly, authorized services can also obtain the encrypted data through T_{data} when they need to provide the functions required by the users.

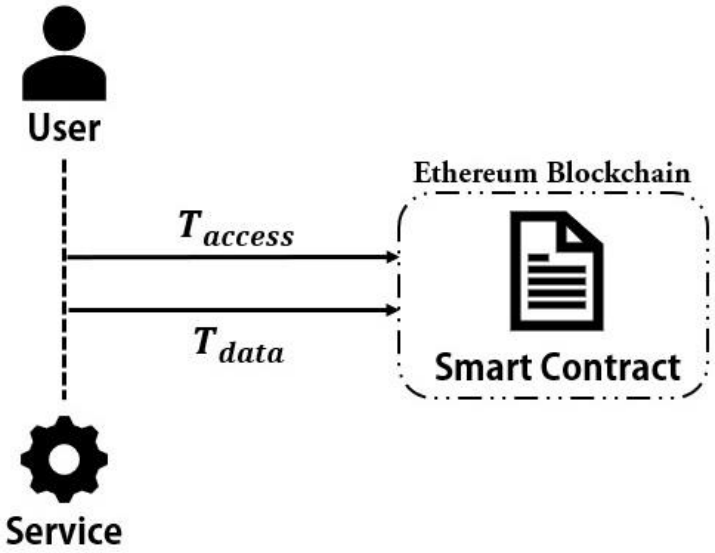

Figure 3: Model framework of the smart contract-based system.

3.2 Security requirements

We now define the security of SCDP system in the presence of $semi-honest$ adversaries. Generally speaking, there are three entities in SCDP system: user, service, and smart contract. In this paper, we assume that both the user and the smart contract are honest parties and will not corrupted by adversary $\mathcal{A}$. Obviously, the security definition we state here is for untrusted third-party service providers. In the SCDP system proposed in this paper, we assume that the $semi-honest$ adversary $\mathcal{A}$ follows the rules for not getting the source code of the smart contract. To define the security of SCDP system, as shown in Figure 4, we construct the following game between a challenger $\mathcal{C}$ and a polynomial adversary $\mathcal{A}$:

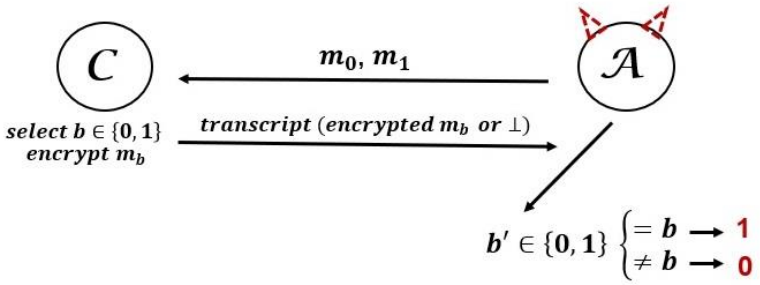

Figure 4: Security game of SCDP system.

First of all, A chooses two messages m_0 and m_1 to send to $\mathcal{C}$ for challenge, $\mathcal{C}$ selects $b \in \{0,1\}$ and then sends the transcript (can be the encrypted data m_b or null $\perp$) to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$. In the case of $b' = b$, $\mathcal{A}$ outputs 1; otherwise, outputs 0. Therefore, the security of the SCDP system is formally defined as follows:

Definition 2. *The SCDP system is secure if there exist a polynomial time adversary* A_{SCDP} *such that*

$$Pr[A_{SCDP}(m_0, m_1, trc) = 1] \leq \frac{1}{2} + negl(n)$$

where, trc denotes the transcript and $negl(n)$ is a negligible function.

3.3 Details of Basic SCDP System

First of all, combining the blockchain, smart contracts and symmetric encryption technology mentioned above, we propose the basic SCDP system whose specific process is shown in Figure 5. The user (u) must generate the key pair of the EOA (u.EOA) and the corresponding private key (u.EPK); the service must generate the key pair of the EOA (s.EOA) and the corresponding private key (s.EPK) before running this decentralized privacy system. In a nutshell, the basic smart contract-based system proposed in this paper consists of four main functions: $Initialization, Data\ Processing, Authority$ and $Data\ Decryption$.

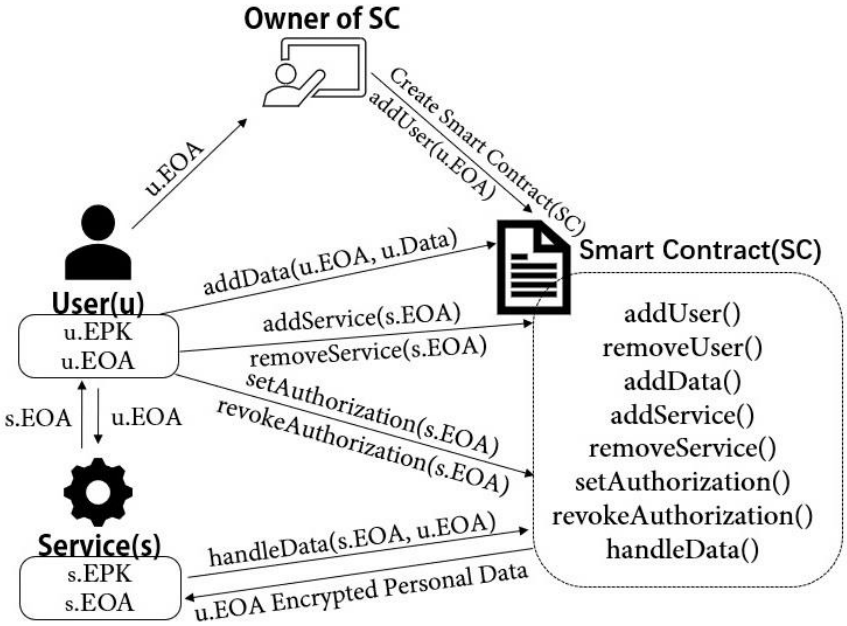

Figure 5: Specific process of basic smart contract-based system.

The specific steps of all functions are described as follows:

- $Initialization$: The owner of smart contract as an Ethereum user will create a smart contract and deploy it upon the Ethereum blockchain at the corresponding smart contract address (SC.Address). The smart contract adds a user into the system when he/she signs up at the first time.
- $Data\ Processing$: After initialization, the SC will send relevant permissions and user's externally owned address (EOA) to the blockchain through T_{access}. Meanwhile, the personal data of user will be encrypted and sent to the smart contract through $\mathrm{T_{data}}$. The encryption key in this scheme is shared between the user and the service.
- $Authority$: The user in the system can add required service and the service's externally owned address (EOA) into the smart contract, and then grant a set of permissions about data to the service through T_{access} if needed. Moreover, the user can also revoke permissions which have been granted if he/she wants. Eventually, the user can remove the service from the system.
- $Data\ Decryption$: Once being authorized, the service can obtain the encrypted personal data through T_{data} and decrypt it with the Shared key mentioned above to provide corresponding services.

Ultimately, if a user requests to revoke his/her account, the owner of smart contract can also remove the user from the smart contract-based system.

4 Strong SCDP System

4.1 Construction

The basic smart contract-based decentralized privacy system solves the efficiency and overhead and issues in [3] to a certain extent, and then provides users with more efficient and economical data ownership control. Nonetheless, it should be noted that this basic smart contract-based system introduces some new defects: The owner of SC is given excessive unnecessary privileges; There are multiple users and services in a single smart contract, which will lead to a series of data permissions confusion and privacy issues; The generation and distribution of shared keys are also critical security issues which require special attention.

In order to improve these defects and enhance the security and confidentiality of the system, we reconstructed the basic smart contract-based system and proposed the strong smart contract-based decentralized privacy system. In this system, a single user corresponds to a single smart

contract, and data encryption is realized through ciphertext-policy attribute-based encryption (CP-ABE) algorithm. The specific process of the strong SCDP system is shown in Figure 6. The symbols in this figure are similar to the introduction part of CP-ABE: PK and MK are public parameters and master keys respectively; SK represents the private key generated in the $KeyGen()$ step; S represents the access structure and A is the attribute set of the service added into the system. And finally, the ciphertext CT is generated by encrypting the personal data M of the user through CP-ABE.

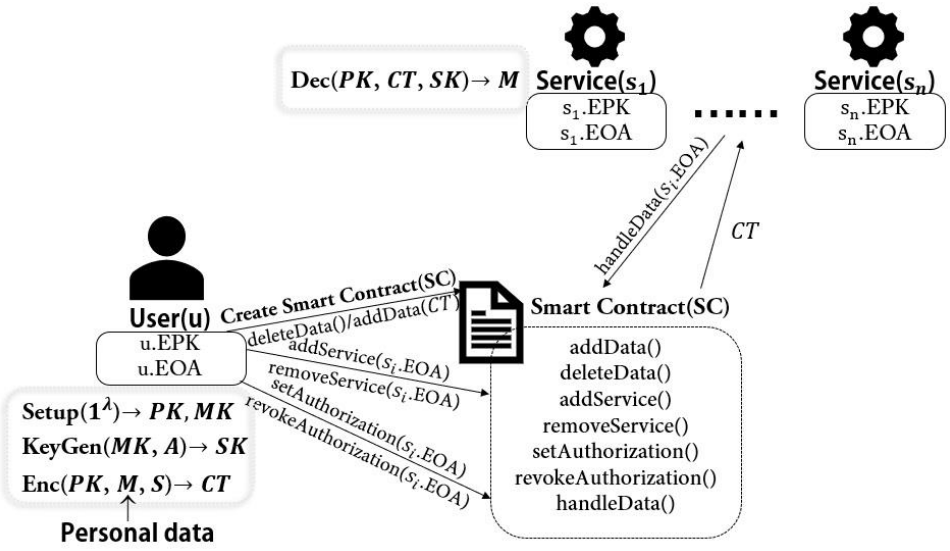

Figure 6: Specific process of strong smart contract-based system.

As shown in Figure 5, the user himself becomes the creator and owner of the SC, and also holds the right to add/remove services from the system. With this configuration, the issue that the owner of the smart contract is given excessive unnecessary privileges no longer exists. The reconstructed smart contract-based system also owns stronger security and flexibility after introducing CP-ABE, which is more suitable to be applied in real-world scenarios.

Specifically speaking, the strong smart contract-based decentralized privacy system is still composed of the following main functions: $Initialization$, $Data\ Processing$, $Authority$ and $Data\ Decryption$. Similarly, the purpose and general structure of these functions are basically unchanged, but couple of specific executions are different. The specific description of the four main functions is stated as follows:

- $Initialization$: The user u of this system initially creates and deploys the personal SC on the blockchain. At the same time, u generates the public parameters PK and master key M through $Setup1^{\lambda}$.
- $Data\ Processing$: To protect privacy, the user u executes $Enc(PK, M, S)$ to convert

personal data M into ciphertext CT and sends it to the blockchain through T_{data}. And, u in this system can also delete related personal data in the system through T_{data}.

- $Authority$: When he/she needs a third-party service, the user u as the owner of the SC adds the service $s_i(1 \leq i \leq n)$ and the service's externally owned address (s_i.EOA) into the system, and then can grant or revoke a set of permissions to s_i through T_{access} at any time. Similarly, u can also easily remove the service s_i from the system.
- $Data\ Decryption$: The service s_i can correctly decrypt the ciphertext CT and gain the user's personal data of the user u if and only if its own attribute set A satisfies the access structure S of u. Then, s_i can provide the functions which u needs through the acquired data.

4.2 Security analysis

In this section, we will show the security proof of strong SCDP system from CP-ABE. Based on the formal security definition proposed in the previous section, in the strong SCDP system, trc denotes the transcript which can be ciphertext CT generated by CP-ABE or null ($\perp$). Therefore, for two different cases of trc, we propose the following lemmas to analyze the security of the system:

Lemma1. In the case of $trc = \perp$, the strong SCDP system is secure if there exist a polynomial adversary $\mathcal{A}_{\mathrm{SCDP}}$ such that

$$Pr[\mathcal{A}_{SCDP}(m_0, m_1, trc) = 1 | trc = \perp] \leq \frac{1}{2} + negl(n)$$

$Proof$. Based on the security of the SC system we defined earlier, we all know the probability that the adversary $\mathcal{A}$ without corresponding conditions can execute certain functions in SC is negligible. Therefore, it is extremely difficult for a $semi-honest$ adversary $\mathcal{A}$ in this system to pretend to be an honest service node which is granted permissions to obtain ciphertext CT. Obviously, in the case of $trc = \perp$, this does not help the adversary $\mathcal{A}$ guess the output b'. Therefore, the probability that b' is equal to b is still $\frac{1}{2} + negl(n)$. Ultimately, the proof of this lemma is done.

Lemma2. In the case of $trc = CT$, the strong SCDP system is secure if there exists a polynomial time adversary $\mathcal{A}_{\mathrm{SCDP}}$ such that

$$Pr[\mathcal{A}_{SCDP}(m_0, m_1, trc) = 1 | trc = CT] \leq \frac{1}{2} + negl(n)$$

$Proof$. To prove Lemma 2, we assume that there exists a polynomial time adversary $\mathcal{A}_{SCDP}$ such that

$$Pr[\mathcal{A}_{SCDP}(m_0, m_1, trc) = 1 | trc = CT] > \frac{1}{2} + negl(n)$$

Then there is an adversary $\mathcal{A}_{CP-ABE}$ who can try to attack CP-ABE through $\mathcal{A}_{SCDP}$ as shown in Figure 7. The adversary $\mathcal{A}_{CP-ABE}$ submits two equal length messages m_0 and m_1 to the challenger $\mathcal{C}$ in $Exp^{CPA}_{CP-ABE,\mathcal{A}}$, thus sends the received transcript trc (the ciphertext CT generated by encrypting m_b in this case) to $\mathcal{A}_{SCDP}$. Finally, the adversary $\mathcal{A}_{CP-ABE}$ will output b' according to the output of $\mathcal{A}_{SCDP}$.

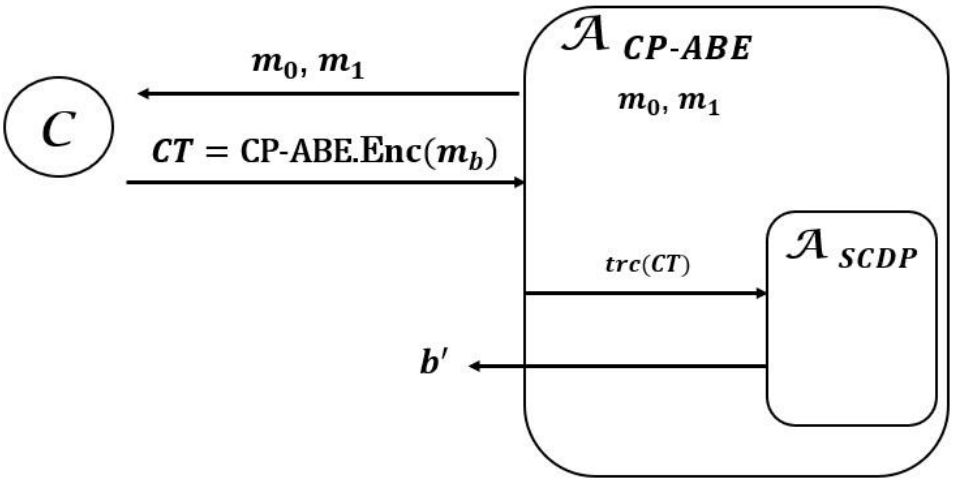

Figure 7: Reduction from $\mathcal{A}_{SCDP}$ to $\mathcal{A}_{CP-ABE}$.

From Figure 7 and the definition of entire reduction we know that

$$Pr[\mathcal{A}_{SCDP}(m_0, m_1, trc) = 1 | trc = CT] = Pr\left[Exp^{CPA}_{CP-ABE,\mathcal{A}}(1^{\lambda}) = 1\right]$$

Accordingly, through our assumption about the ability of $\mathcal{A}_{SCDP}$ we can finally conclude

$$Pr\left[Exp^{CPA}_{CP-ABE,\mathcal{A}}(1^{\lambda}) = 1\right] > \frac{1}{2} + negl(n)$$

Finally by contradiction, the proposed system is secure if CP-ABE is secure. Hence the proof of this lemma is done.

5 Analysis

5.1 Comparisons

We compare our SCDP system with the original decentralized privacy (DP) system in terms of different attributes. The specific comparisons of these two systems are shown in Table 1.

Table 1: Comparisons of two DP systems

Two DP Systems	**Main Attributes**			
	Data ownership	Access control	Blockchain	Off-blockchain
DP	Achieved	Achieved	Required	Required
SCDP	Achieved	Achieved	Required	Not Required

5.2 Costs

It was mentioned above that in Ethereum all operations about SC require Ether and gas. On May 2019, 1 Ether $\approx$ 249.95 USD, and 1 gas $\approx$ 1 wei (0.000000001 eth) was used. The costs of executing various functions of SC in this system are shown in Table 2 (costs of data-related functions are affected by data size).

Table 2: Costs of different SC functions

Function	**Gas USed**	**USD**
Deploy Contract	1820106	0.4549
addService	111533	0.0279
removeService	23698	0.0059
setAuthorization	30677	0.0077
revokeAuthorization	30941	0.0077

5.3 Implementations

5.3.1 Blockchain Implementations

Over the past decade, the blockchain has received widespread attentions in multiple fields as a peer-to-peer distributed ledger of transactions which is simultaneously stored by all participating nodes. There is no central authority in the blockchain, and the functions of the

blockchain are mainly guaranteed by the consensus mechanism to achieve secure transactions between untrusted nodes. As shown in Figure 8, the blockchain is composed of a series of data blocks connected in chronological order. The specific steps of data block and blockchain generation are as follows [7]: Users in the blockchain interact with the peer-to-peer network through the pair of public key and private key. They can sign the transaction by their private key and broadcast it to other nodes on the blockchain network. After that, the neighboring peers will check the validity of the received transaction and eventually discard the invalid transaction. Within a certain time interval, the transactions verified by the above steps are packaged into a candidate block and broadcast back to the network. This process related to data packing and returning is called mining and is done by so-called miners. Then the candidate block will be added to the blockchain after passing verification, otherwise it will be discarded. Finally, a round of update of this blockchain network ends.

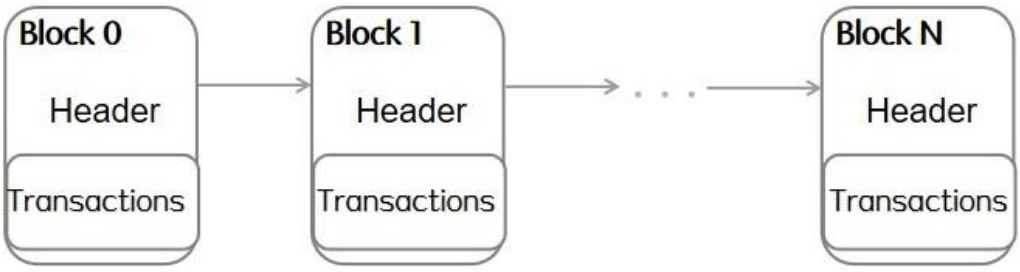

Figure 8: Construction of blockchain.

Ethereum is a public blockchain platform which supports decentralized applications and customized smart contracts with Turing-complete scripting language. Most of Ethereum blockchain protocols are similar to Bitcoin, yet Ethereum is suited for a wider range of applications. The Ethereum platform is mainly composed of the following elements:

Ethereum Account. There are two types of Ethereum accounts: Contract Accounts, controlled by the code stored in the account; Externally Owned Accounts (EOA), controlled by external user private keys. All accounts hold corresponding balance and nonce fields, and contract accounts also have the storage fields and contract code.

Message and Transactions. In Ethereum, A transaction is composed of the account nonce, the signature of the sender, the address of the recipient, the optional additional data fields, the Ether value, and two values called startGas and gasPrice. Messages are similar to transactions, but can only be sent between contracts.

Smart Contracts. The SC is a type of computer protocol which can be self-executed and self-

verified once the specified conditions are met. The deployment process of SC on the Ethereum blockchain is shown in Figure 9. After being successfully compiled, each SC is created and assigned to a unique address. Therefore, users can interact with smart contracts through the recorded contract address and the Application Binary Interface (ABI).

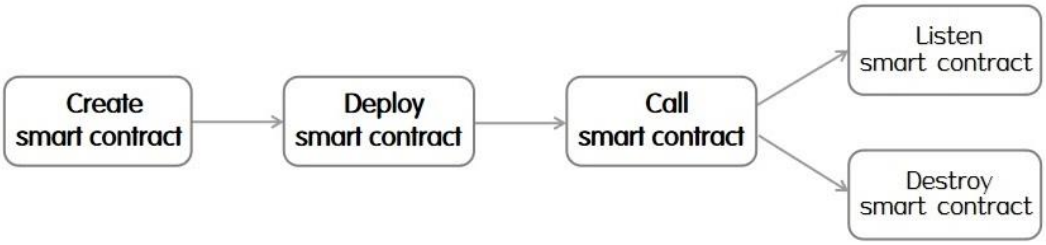

Figure 9: Creation and deployment process of smart contract.

5.3.2 CP-ABE Implementations

The CP-ABE part of the strong SCDP system proposed in this paper is implemented by the scheme proposed by Odelu et al. [19]. This RSA-based [13] CP-ABE scheme satisfies constant size secret keys and ciphertexts (CSKC) and holds lower computation costs. The specific CP-ABE algorithm is mainly composed of the following four steps:

Setup. In this phase, the setup algorithm takes as inputs the security parameter 1^{λ} and the universe of attributes $\mathbb{U} = A_1, A_2, \dots, A_n$. The algorithm chooses two RSA primes p and q to compute $N = pq$ with $p \neq q$. Then, it randomly select the RSA pubic exponent p_i with $gcd(p_i, \emptyset(N))$, and computes q_i such that $p_i q_i \equiv 1 (mod \emptyset(N))$ corresponding to each attribute $A_i \in \mathbb{U}$. After that, two system private keys k and x are picked such that $gcd(k, \emptyset(N)) = 1$, $gcd(k, q_i) = 1$, $gcd(x, q_i) = 1$, for all $i = 1,2 \dots, n$. Further, we select a random number g which satisfies $2 < g < N - 1$ and $gcd(g, N) = 1$. Subsequently, the setup algorithm chooses three one-way collision-resistance hash functions:

$$H_1: \{0,1\}^* \rightarrow \{0,1\}^{\lambda}$$
$$H_2: \{0,1\}^* \rightarrow \{0,1\}^{l_\delta}$$
$$H_3: \{0,1\}^* \rightarrow \{0,1\}^{l_m}$$

where l_δ represents the length of a random string δ under the security parameter 1^{λ} and l_m is the length of the plaintext M.

Then, it computes the public parameters $D_{\mathbb{U}} = g^{D_{\mathbb{U}}}, \mathrm{Y} = g^{x}$ and $R = g^{k}$, where $D_{\mathbb{U}} = \prod_{A_i \in \mathbb{U}} q_i$. Ultimately, the setup algorithm outputs the master public key P and the master secret key K, where

$$K = \{k, x, p, q, q_1, \ldots, q_n\}$$
$$P = \{N, D_{\mathbb{U}}, Y, R, H_1, H_2 H_3, p_1, \ldots, p_n\}$$

Encrypt. In this phase, the encryption algorithm takes as inputs an access structure $\mathbb{S} \subseteq \mathbb{U}$ with $|\mathbb{S}| \neq 0$, the master public key P and the plaintext M, finally outputs the ciphertext CT. In the first place, it picks a random number $\delta_m \in \{0,1\}^{l_\delta}$ to compute $r_m = H_1(\mathbb{S}, M, \delta_m)$. Then, the encryption algorithm computes K_m as

$$K_m = D_{\mathbb{U}}^{r_m \frac{e_{\mathbb{U}}}{e_{\mathbb{S}}}}$$
$$= (g^{d_{\mathbb{U}}})^{r_m \frac{e_{\mathbb{U}}}{e_{\mathbb{S}}}}$$
$$= g^{r_m d_{\mathbb{S}}}$$

where $d_{\mathbb{S}} = \prod_{A_i \in \mathbb{S}} q_i, e_{\mathbb{S}} = \prod_{A_i \in \mathbb{S}} p_i, e_{\mathbb{U}} = \prod_{A_i \in \mathbb{U}} p_i$.

After that $Y_m = g^{kr_m}, R = g^{kr_m}, CT_{\delta_m} = H_2(K_m) \oplus \delta_m, CT_m = H_3(\delta_m) \oplus M$, and $S_m = H_1(\delta_m, M)$ are computed and finally this algorithm outputs the ciphertext $CT = \{\mathbb{S}, Y_m, R_m, CT_{\delta_m}, CT_m, S_m\}$

Key Generation. This algorithm takes the attribute set $\mathcal{A}$, the master public key P and the master secret key K as inputs and outputs the user secret key k_s. It initially computes $d_{\mathbb{A}} = \prod_{i=1}^{n} {q_i}^{a_i}$, where $a_i = 1$ if $\mathrm{A_i} \in \mathbb{A}$ and $a_i = 0$ if $\mathrm{A_i} \notin \mathbb{A}$. Then two random numbers r_u and t_u are picked to compute s_u such that it satisfies $d_{\mathbb{A}} = k s_u + x r_u (mod \emptyset(N))$. Furthermore, it also computes $k_1 = s_u + x t_u (mod \emptyset(N))$ and $k2 = r_u - k t_u (mod \emptyset(N))$. At last, the key generation algorithm outputs the user secret key $k_s = (k_1, k_2)$.

Decrypt In this phase, the decryption algorithm takes as input the secret key $k_s = (k_1, k_2)$. corresponding the attribute set $\mathbb{A}$ and ciphertext CT corresponding to the access policy $\mathbb{S}$, and takes the original plaintext message M as output. Generally speaking, we know that $\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}$ is an integer if and only if the attribute set $\mathbb{A}$ satisfies the access structure $\mathbb{S}$, so in this case the decryption algorithm computes

$$\begin{aligned} K_m &= (Y_m^{k_2} R_m^{k_1})^{\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}} \\ &= (g^{xr_m(r_u - kt_u)} g^{kr_m(s_u + xt_u)})^{\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}} \\ &= (g^{r_m(xr_u + ks_u)} g^{xr_m(-kt_u) + kr_m(xt_u)})^{\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}} \\ &= (g^{r_m d_{\mathbb{A}}})^{\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}} \\ &= g^{r_m d_{\mathbb{S}}} \end{aligned}$$

Otherwise, in the case $\frac{e_{\mathbb{A}}}{e_{\mathbb{S}}}$ is not an integer, then K_m is computationally infeasible. Subsequently, $\delta'_m = H_2(K_m) \oplus C_{\delta_m}$ and $M' = C_m \oplus H_3(\delta'_m)$ are computed.

Last of all, the decryption algorithm checks whether the condition $S_m = H_1(\delta'_m, M')$ holds or not. if so outputs the plaintext M, if not outputs $\perp$ (null).

6 Conclusions

In this paper, we propose an SCDP system which combines SC with cryptographic techniques to address data ownership and privacy issues when users use third-party services. We construct the basic SCDP system and ultimately propose a strong SCDP system which enables secure data ownership management, simultaneously avoiding the additional overhead of connection between management and storage in the previous DP system.

References

[1] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in Annual International Conference on the Theory and Applications of Cryptographic Technique. Springer, 2005, pp. 457–473.

[2] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum project yellow paper, vol. 151, pp. 1–32, 2014.

[3] G. Zyskind and O. Nathan, "Decentralizing privacy: Using blockchain to pro- tect personal data," inSecurity and Privacy Workshops (SPW), 2015 IEEE. IEEE, 2015, pp. 180–184.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in Security and Privacy, 2007. SP'07. IEEE Symposium on. IEEE, 2007, pp. 321–334.

[5] J. Kehrli, "Blockchain 2.0-from bitcoin transactions to smart contract applications," Niceideas, November. Available at: https://www. niceideas. ch/roller2/badtrash/entry/blockchain-2-0-frombitcoin (Accessed:5 January 2018), 2016.

[6] J. P. Cruz, Y. Kaji, and N. Yanai, "Rbac-sc: Role-based access control using smart contract,"IEEE Access, vol. 6, pp. 12 240–12 251, 2018.

[7] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," Ieee Access, vol. 4, pp. 2292–2303, 2016.

[8] K. Delmolino, M. Arnett, A. Kosba, A. Miller, and E. Shi, "Step by step towards creating a safe smart contract: Lessons and insights from a cryptocurrency lab," in International Conference on Financial Cryptography and Data Security. Springer, 2016, pp. 79–94.

[9] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital supply chain transformation toward blockchain integration," in proceedings of the 50th Hawaii international conference on system sciences, 2017.

[10] M. Haferkorn and J. M. Q. Diaz, "Seasonality and interconnectivity within cryptocurrencies-an analysis on the basis of bitcoin, litecoin and namecoin," in International Workshop on Enterprise Applications and Services in the Finance Industry. Springer, 2014, pp. 106–120.

[11] N. Szabo, "Smart contracts," Unpublished manuscript, 1994.

[12] P. McCorry, S. F. Shahandashti, and F. Hao, "A smart contract for boardroom voting with maximum voter privacy," in International Conference on Financial Cryptography and Data Security. Springer, 2017, pp. 357–375.

[13] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and

public-key cryptosystems," Communications of the ACM, vol. 21, no. 2, pp. 120–126, 1978.

[14] S. King and S. Nadal, "Ppcoin: Peer-to-peer crypto-currency with proof-of-stake," self-published paper, August, vol. 19, 2012.

[15] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[16] V. Buterin, "A next-generation smart contract and decentralized application platform,"

[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in Proceedings of the 13th ACM conference on Computer and communications security. Acm, 2006, pp. 89–98.

[18] V. Jacynycz, A. Calvo, S. Hassan, and A. A. S´anchez-Ruiz, "Betfunding: A distributed bounty-based crowdfunding platform over ethereum," in Distributed Computing and Artificial Intelligence, 13th International Conference. Springer, 2016, pp. 403–411.

[19] V. Odelu, A. K. Das, M. K. Khan, K.-K. R. Choo, and M. Jo, "Expressive cp-abe scheme for mobile devices in iot satisfying constant-size keys and ciphertexts," IEEE Access, vol. 5, pp. 3273–3283, 2017.