Decentralized Access Control Encryption in Public Blockchain

Zhongyuan Yao[1,2](✉), Heng Pan[2], Xueming Si[2,3], and Weihua Zhu[2]

[1] The Henan Key Laboratory on Public Opinion Intelligent Analysis, Zhengzhou 450001, Henan, China
zy454@uowmail.edu.au
[2] Zhongyuan University of Technology, Zhengzhou 450001, Henan, China
[3] Fudan University, Shanghai 200433, China

Abstract. Since its invention, the public blockchain has attracted more attention from both the academia and industry because of its fully decentralization and persistency features. However, the privacy issue in public blockchain is still challengeable. While there exists privacy preservation mechanisms proposed for the public blockchain, almost all of them can only solve partial of the privacy issue, either user privacy or data privacy indeed, in it. In this work, we present a decentralized access control encryption scheme which ensures user and data privacy simultaneously in public blockchain. With our cryptographic solution, the validity of one specific transaction can be publicly verified, while its content can only be retrieved by its intended receivers. Moreover, the origin of this transaction cannot be identified by any participant except the receivers in the network. Our analysis shows that our solution is really suitable to deploy in public blockchain and is proven secure under mathematical assumptions.

Keywords: Public blockchain · User privacy · Data privacy · Access control encryption · Decentralized sanitizers

1 Introduction

1.1 Background

Since it was first introduced in the seminal work [29] in 2008, the blockchain technology has received great interests and world-wide recognition from both the industry and academia. Generally, a blockchain can be valued as a chain of blocks in which all committed transactions are stored. The chain grows continuously when new blocks are generated and appended to it. As a perfect combination of several technologies including cryptographic hash, digital signature and distributed consensus mechanism, the blockchain enjoys desirable characteristics such as decentralization, tamper-resistance, auditability. Although the blockchain technology is initially served as the core mechanism for the Bitcoin, it can also be applied into various scenarios apart from cryptocurrencies.

For example, blockchain enables a transaction to be taken place in a decentralized environment and thus allows financial activities to be processed without any intermediary such as bank [21]. Moreover, with the blockchain technology, the transparency and traceability of ownership of the supply chain management system can be enhanced [16,45]. Additionally, the blockchain technology has witnessed its application in internet of things (IoT) [32], security services [38] and public services [26], etc.

According to the degree of centralization of the blockchain network, the blockchain can be categorized into three types: the public blockchain, the consortium blockchain and the private blockchain [10]. In a public blockchain network, anyone and everyone can join or leave it freely. Moreover, the participant is enabled to perform all core functionalities of the public blockchain including reading, writing and auditing the ongoing activities on the network. Obviously, the public blockchain is "fully decentralized". One example of the public blockchain is the blockchain used in the Bitcoin. Unlike the public blockchain where all nodes need to participate in the consensus process, the consortium blockchain requires a pre-selected set of nodes to control that process. Furthermore, The right to read the consortium blockchain may be public, or restricted to the participants. There may also exist hybrid routes where the root hashes of the blocks is public in consortium blockchain, it allows the public to make a limited number of queries to know the knowledge of partial of the blockchain state. Thus, the consortium is considered "partially decentralized". One popular consortium blockchain framework implementation is the Hyperledger Fabric project hosted by the Linux Foundation. A private blockchain, as its name suggests, is a blockchain where there exists a central trusted party which is in charge of the task of writing to the chain and granting selected parties the right to read the chain. The private blockchain seems more like a traditional centralized system empowered with a certain degree of cryptographic auditability. Therefore, it is still debatable that whether such a private system can be called a "blockchain". In fact, both the consortium blockchain and the private blockchain still require one or some trusted authorities in the system, which is contradict to the untrustworthy feature of the blockchain proposed in Bitcoin. Thus, they can hardly realize the full decentralization purpose introduced in [29].

1.2 Research Problem

Although the blockchain technology has developed rapidly over the past 10 years, it still has insufficiencies. Among them, the most fatal one is the privacy issue, which prevents it from being used in more potential applications where privacy is their first security concern. Indeed, the privacy issue in blockchain contains two sub-problems, the user and transaction privacy respectively. The user privacy says that it is impossible to trace the original issuer of a transaction form the script. While the transaction privacy ensures that the contents included in one transaction can only be accessed by specified users, and is confidential to the blockchain network. We find there are privacy preserving mechanisms which alleviate the privacy threat to the consortium and private blockchain, and some cryptographic tools are introduced to the public blockchain to ensure either user privacy or transaction privacy. However, there exists no thorough solution which enables the public blockchain to preserve both the user and transaction privacy. Since it is the public blockchain that achieves the goal of full decentralization and living in environment with no trust, and it is this two unique properties which make the public blockchain that attractive, we argue that the problem solving the privacy issue in public blockchain is meaningful and extremely urgent. Thus, we value it as our research problem.

1.3 Existed Works

The privacy problem in blockchain has been studied intensively since its invention. In the private blockchain, as it can be treated as a centralized system, many classical cryptographic primitives can be directly deployed to solve the privacy problem. Apart from cryptographic algorithms, some special privacy preserving methods are used in the consortium blockchain. For example, in the Enigma blockchain system [48], data is torn into pieces and then covered using ingenious mathematical technique, this method makes it impossible to recover the related origin data from one piece or a part of the pieces of the data. And in the Hyperledger Fabric project [41], multiple channels are built to separate different ledgers from different organizations and thus force a user to access to the only ledger generated cooperatively by users of its organization. However, since the public blockchain is fully decentralized and there exists no trusted party comparing to the private and consortium blockchain, the aforementioned mechanisms are not applicable to it.

There are some techniques used in the public blockchain to solve either the user or data privacy problem. For example, to hide the user privacy information leaked from the transaction of the public blockchain, the mixing mechanism, first proposed in [11], is introduced to obfuscate the transactions' relationships and thus makes it unlinkable between the sender and receiver of a transaction. There are multiple mixing service providers [1–3] available at present and some of them use centralized mixing mechanism. However, this centralized service imposes new security threat on the blockchain. That is, the mixing service provider must be always honest and highly reliable, which is a rather unrealistic assumption. To relieve the service subscribers of this security concern, the decentralized mixing mechanism is presented in [19,37]. Moreover, there are two approaches to realize this decentralized mixing mechanism. The first one is based on the CoinJion technique described in the Bitcoin Forum [27], and the main idea of it is "When you want to make a payment, find some one else who also wants to and make a joint one together.". The other way is the multi-party computation technique, first presented in [46]. With this method, all parties can only learn knowledge from the output and their own input after the computation. Thus, it is suitable for decentralized tasks where the computation is executed without a trusted party. Although those mixing mechanisms enjoy several benefit when providing user privacy in public blockchain, they still suffer from several limitations such as high mixing service fee, high delay waiting for the service and the risk of being blocked by Sybil attack.

Another tool to preserve the user privacy in public blockchain is the ring signature invented in [36]. With this primitive, a user can produce a valid signature on behalf of a set of possible signers chosen by itself, and no one can tell the real signer form the "ring" of users given only the transcript. One modified version of the ring signature is the traceable ring signature proposed in [18]. It empowers participants with the capability to link two signatures if they are signed by the same signer using the same message and tag and to reveal the anonymity of the signer of two signatures if they are produced with the same tag. The desirable properties, such as anonymity and linkability, provided by the ring signature have cultivated several ring-based privacy preservation mechanisms for the public blockchain, such example can be found in the Monero project [30] and also in [31,42]. The ring signature provides strong user anonymity in public blockchain, however, it also incurs several problems to it. For example, the size of transactions and the signature itself could be very large if too many participants are involved in the "ring".

There are two main approaches to preserve data privacy in the public blockchain, i.e., NIZK proof and homomorphic cryptosystem. First proposed in [39], the non-interactive zero-knowledge proof (NIZK), which does not need the interaction phase between the prover and verifier, is a variant of the zero-knowledge proof (ZKP). The ability to independently prove the correctness of an assertion without leaking extra knowledge makes NIZK poof well suited for creating message privacy preserving protocols. The NIZK proof provides three desirable properties, the completeness, soundness and zero-knowledge. One example of the application of the NIZK proof in blockchain is the Zerocash project issued in [28]. In Zerocash, one special edition of the NIZK proof called zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) proofs [25] is used to hide a payment's address. Furthermore, the coin value is added in the commitment and zero-knowledge proof so that it is arbitrary and publicly verifiable. The zk-SNARKs proof achieves high level of user and data privacy in public blockchain, however, the expense of generating the transaction proofs is hardly affordable in such type of blockchain. The homomorphic cryptosystem supports a cryptographic methodology that satisfies homomorphism, it enables any participant to perform computation on the ciphertexts while preserving the data privacy. In general, homomorphic cryptography performs as black box, when given n ciphertexts and operations, it outputs the encrypted result of the same operations on the corresponding original data. This attractive feature makes homomorphic cryptography well suited for hiding and performing update of the amount and other metadata of a transaction. Typical implementations of homomorphic cryptographic systems which aim to protect data privacy of the public blockchain include the Pedersen commitment scheme [34] and the Paillier cryptosystem [33].

1.4 Our Contribution

In this paper, we present a decentralized access control encryption (DACE) scheme to tackle both the user privacy and data privacy problem in public blockchain simultaneously. With our DACE scheme, the content of one specific transaction can only be retrieved by its intended receivers, and also, the receiver can not identify the exact issuer of that transaction. This work contributes to the development of solving privacy issues in public blockchain in the following two aspects.

Firstly, we first introduce the access control encryption to public blockchain and give the first DACE scheme construction with compact size ciphertext. Our construction borrows idea from the primitive anonymous broadcast encryption [9,24,47], the resulted scheme keeps not only the ciphertext size compact but also the key size of each users in the DACE compact. Which makes our scheme possible to be deployed in the public blockchain considering the resource-constraint characteristic of it.

The main contribution of this paper is giving a decentralized implementation of the ACE scheme and further increasing its reliability in public blockchain. In our paper, we allow the sanitizing key to be shared among n sanitizers, while each node joining the public blockchain can be a sanitizer if it wants to. Only exact t of those nodes can collaboratively transform a ciphertext into a valid sanitized ciphertext which can be correctly decrypted by its receivers. In our construction, each of the n sanitizers is installed with an unique sanitizing key, and it would execute the same sanitizing algorithm on the ciphertext, either not or partially sanitized, received. One ciphertext can only be viewed as a partially sanitized ciphertext and cannot be decrypted by its intended receivers until it is processed by t sanitizers. Unlike previous scheme with only one sanitizer, our construction distributes the sanitizing functionality of the origin ACE among n sanitizers, it is impossible for one of sanitizers in our construction to produce a new access policy, so our construction imposes restriction on the capability of the sanitizer. Besides, as one message sender in our DACE construction can choose the t sanitizers itself to collaboratively produce a valid sanitized ciphertext, even some of the n sanitizers cannot provide service or off-line, our DACE system can never encounter the single-point-failure problem, so our DACE improves the reliability of the sanitizer and even the robustness of original ACE system.

1.5 Paper Organization

The rest of our paper is organized as follows. Section 2 presents useful notations and security assumptions used throughout our paper. We also formally give the definition of the decentralized access control encryption (DACE) scheme and two security models to cover the no-read rule and no-write rule property predefined in previous work in this section. In Sect. 3, in order to make the description of our scheme more easy to understand, we first present a new notion "sanitizing pipeline", then we give concrete construction of our DACE. We give security proofs in Sect. 4 and conclude this paper in Sect. 5.

2 Preliminaries and Definitions

2.1 Notations

Here, for the benefit of consistency, we give the notations used throughout the whole paper. Let $\mathbb{Z}_p$ denote a additive group where p is a large prime, let $\mathbb{G} \subset \mathbb{Z}_p$ be a multiplicative group with large prime order q and generator g, here we have $q|p-1$. For simplicity, we use $[u]$ to represent the successive list $\{0, 1, \cdots, u\}$. There are always three types of users involved in the ACE scheme, we denote them the message sender Se, the message sanitizer San and the message receiver Re separately. For a specific user, he can play the role of both Se and Re, we use ke, kd to represent this user's encryption and decryption key respectively. Assuming there are l layers in the ACE system, when a user in layer $\alpha \in [l]$ can send messages to a receiver in layer $\beta \in [l]$, we use the notation $\alpha \times \beta \rightarrow 1$ to denote such access policy, otherwise $\alpha \times \beta \rightarrow 0$. We use the access policy set $P : [l] \times [l] \rightarrow \{0, 1\}$ to cover the collection of all the access polices defined in the ACE system. When there exists n message sanitizers in our ACE definition, we assume each of them holds a unique secret sanitizing key ks. In order to keep the consistency of the description of our ACE system, we use [u+1,u+n] to denote the list $\{u+1, u+2, \cdots, u+n\}$ and also to represent identities of the n sanitizers, for simplicity, we use the notation $[u+n]$ to represent all identities of users involved in the ACE definition.

2.2 Hard Problems

Definition 1 (K-CCA problem). *For an integer k, and one element x randomly chosen from $\mathbb{Z}_q$, $g \in \mathbb{G}$, given $g, g^x, h_1, h_2, \cdots, h_k \in \mathbb{Z}_q, g^{\frac{1}{x+h_1}}, g^{\frac{1}{x+h_2}}, \cdots, g^{\frac{1}{x+h_k}}$ as inputs, the problem solver needs to output element $g^{\frac{1}{x+h}}$ for some $h \notin \{h_1, h_2, \cdots, h_k\}$. We say an algorithm $\mathcal{B}$ has advantage ϵ within time t in solving the K-CCA problem in $\mathbb{G}$ if*

$$\Pr[\mathcal{B}(g, g^x, h_1, h_2, \cdots, h_k \in \mathbb{Z}_q, g^{\frac{1}{x+h_1}}, g^{\frac{1}{x+h_2}}, \cdots, g^{\frac{1}{x+h_k}}) = g^{\frac{1}{x+h}}] \geq \epsilon,$$

where $h \notin \{h_1, h_2, \cdots, h_k\}$ and the probability is over the random choice of the generator g in $\mathbb{G}$, the random choice of $h_1, h_2, \cdots, h_k \in \mathbb{Z}_q$ and the random bits consumed by $\mathcal{B}$.

Definition 2 ((f, g, F)-GDDHE [13]). *Let $\mathcal{B} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e(\cdot, \cdot))$ be a bilinear map group system and let f, g be two co-prime polynomials with pairwise distinct roots, of respective orders t and n. Let g_0, h_0 be one generator of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. the (f, g, F)-GDDHE problem is, given the tuple*

$$\begin{array}{lll} (\, g_0, g_0^{\gamma}, g_0^{\gamma^2}, \cdots, g_0^{\gamma^{t-1}}, & g_0^{\gamma \cdot f(\gamma)}, & g_0^{k \cdot \gamma \cdot f(\gamma)}) \in \mathbb{G}_1, \\ (\, h_0, h_0^{\gamma}, h_0^{\gamma^2}, \cdots, h_0^{\gamma^{2n}}, & & h_0^{k \cdot g(\gamma)}) \in \mathbb{G}_2 \quad \textit{and} \\ \quad T \in \mathbb{G}_T & & \end{array}$$

to decide whether T is equal to $e(g_0, h_0)^{k \cdot f(\gamma)} \in \mathbb{G}_T$ or is a random element in $\mathbb{G}_T$.

Theorem 1 (Generic security of (f, g, F)-GDDHE [13]). *For any probabilistic algorithm $\mathcal{A}$ that totalizes of at most q queries to the oracles performing the group operations in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and the bilinear map $e(\cdot, \cdot)$, let $\mathsf{Adv}^{gddhe}(f, g, F, \mathcal{A})$ denote the advantage that $\mathcal{A}$ can solve the (f, g, F)-GDDHE, then*

$$\mathsf{Adv}^{gddhe}(f, g, F, \mathcal{A}) \leq \frac{(q + 2(n + t + 4) + 2)^2 \cdot d}{2p}$$

with $d = 2 \cdot \mathsf{max}(n, t + 1)$.

2.3 Defining Our ACE

An ACE scheme with decentralized sanitizers is defined by the following polynomial time algorithms:

Setup(P, λ). On input the security parameter λ and an access policy set $P : [u] \times [u] \rightarrow \{0, 1\}$, the Setup algorithm outputs a master secret key msk and the public parameter pp, which include the description of the message space $\mathcal{M}$, the ciphertext space $\mathcal{C}$ and the sanitized ciphertext space $\mathcal{C}'$.

KeyGen(msk, i, t). On input msk, an identity $i \in [u + n]$ and a user type $t \in \{Se, Re, San\}$, the key generation algorithm KeyGen produces the following different types of keys accordingly:

- $ke_i = \mathsf{KeyGen}(msk, i, Se)$ when the user with identity i is a message sender, that is $t = Se$. ke_i is called the encryption key for that user.
- $kd_i = \mathsf{KeyGen}(msk, i, Re)$ when the user with identity i acts as a message receiver, that is $t = Re$. kd_i is called the decryption key for that user.
- $ks_i = \mathsf{KeyGen}(msk, i, San)$ when the user with identity i plays the role of a message sanitizer, that is $t = San$. ks_i is called the sanitizing key for that user.

Enc(ke_i, m). The encryption algorithm Enc, on input an encryption key ke_i and a message $m \in \mathcal{M}$, outputs a ciphertext $c \in \mathcal{C}$.

Sanit(c, SP_l). For one incoming ciphertext $c \in \mathcal{C}$, a sanitizer in one chosen sanitizing pipeline SP_l would process it using this sanitation algorithm Sanit with its own sanitizing key, and then relay the result to another sanitizer in the same path, and the next sanitizer would do the same as its predecessor. Our ACE scheme with decentralized sanitizers requires that c should be processed by all t sanitizers in the sanitizing pipeline SP_l collaboratively before becoming a valid sanitized ciphertext $c' \in \mathcal{C}'$.

Dec(c', kd_j). On input a sanitized ciphertext $c' \in \mathcal{C}'$ and a decryption key kd_j, the decryption algorithm Dec recovers the message $m' \in \mathcal{M} \cup \{\perp\}$.

2.4 Security Notions for Our DACE

Our DACE scheme must satisfy requirements formalized below:

Definition 3 (Correctness). *For all $m \in \mathcal{M}, i, j \in [u]$ such that $P(i,j) = 1$:*

$$\Pr[\mathsf{Dec}(kd_j, \underbrace{\mathsf{Sanit}(ks_t, \cdots, \mathsf{Sanit}(ks_l, \mathsf{Enc}(ke_i, m)))}_{t}) \neq m] \leq negl(\lambda)$$

with $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$, $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$, $kd_j \leftarrow \mathsf{KeyGen}(msk, j, Re)$ and $ks_l \leftarrow \mathsf{KeyGen}(msk, l, San)$, where $l \in [u+1, u+n]$. The above notation denotes that the encrypted message should be processed by exact t different sanitizers in the same sanitizing pipeline before becoming a valid sanitized ciphertext and then being decrypted to a valid plaintext, otherwise, the probability of a correct decryption should be negligible. The probability is taken over the random coins of all involved algorithms.

Definition 4 (No-Read Rule). *To define the No-Read Rule in our DACE scheme, we consider the following game played between a challenger $\mathcal{C}$ and a stateful adversary $\mathcal{A}$:*

No-Read Rule	
Game Definition	Oracle Definition
1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$;	$\mathcal{O}_G(i, t)$:
2. $(m_0, m_1, i) \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot), \mathcal{O}_E(\cdot)}(pp)$;	*1.* $k_i \leftarrow \mathsf{KeyGen}(msk, i, t)$
3. $b \leftarrow \{0, 1\}$	
4. $c \leftarrow \mathsf{Enc}(\mathsf{KeyGen}(msk, i, Se), m_b)$	$\mathcal{O}_E(i, m)$:
5. $c' \leftarrow \underbrace{\mathsf{Sanit}^{\mathcal{O}_G()}(\cdots, \mathsf{Sanit}^{\mathcal{O}_G()}(ks_1, c))}_{t}$	*1.* $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$;
6. $b' \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot), \mathcal{O}_E(\cdot)}(c')$	*2.* $c \leftarrow \mathsf{Enc}(ke_i, m)$
	3. $c' \leftarrow \underbrace{\mathsf{Sanit}^{\mathcal{O}_G()}(\cdots, \mathsf{Sanit}^{\mathcal{O}_G()}(ks_1, c))}_{t}$

Where $P : [u] \times [u] \to \{0, 1\}$ is the given access policy set and $t \in \{Se, Re, San\}$. When $|m_0| = |m_1|$, $i \in [u]$ and for all queries q to $\mathcal{O}_G$ with $q = (j, Re)$, it holds

$$P(i, j) = 0,$$

we say that the adversary $\mathcal{A}$ wins the No-Read game if its output $b' = b$.
Let $\Pr[\mathcal{A}$ wins the No-Read game$]$ denote the probability the $\mathcal{A}$ wins the predefined game and $\mathsf{Adv}^{\mathcal{A}}_{No-Read}(ACE)$ its advantage to win the game, then an ACE scheme is said to satisfy the No-Read Rule if for all probabilistic polynomial time(PPT) algorithm $\mathcal{A}$

$$\mathsf{Adv}^{\mathcal{A}}_{No-Read}(ACE) = 2|\Pr[\mathcal{A} \text{ wins the game}] - \frac{1}{2}| \leq negl(\lambda).$$

Remark. The No-Read Rule model in [12] also covers the sender anonymity, or key-privacy, property when the second, fourth step of our game definition is changed to

$$(m_0, m_1, i_0, i_1) \leftarrow \mathcal{A}^{\mathcal{O}_G(\cdot),\mathcal{O}_E(\cdot)}(pp), c \leftarrow \mathsf{Enc}(\mathsf{KeyGen}(msk, i_b, Se), m_b)$$

accordingly and the requirement $P(i, j) = 0$ is changed to

$$m_0 = m_1, P(i_0, j) = P(i_1, j).$$

It is easy to find that our model can be extended to guarantee the sender anonymity with the above minimal modification, and the corresponding security proof would not be changed a lot indeed. Here, for simplicity, we first concentrate on the basic No-Read property.

Definition 5 (Extended No-Write Rule). *To define a model capturing the security of the sanitizers, we consider the following game played between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:*

Extended No-Write Rule	
Game Definition	Oracle Definition
1. $(pp, msk) \leftarrow \mathsf{Setup}(1^\lambda, P)$;	$\mathcal{O}_S(i, Se)$:
2. $(m_0, m_1, j) \leftarrow \mathcal{A}^{\mathcal{O}_S(\cdot),\mathcal{O}_E(\cdot)}(pp)$;	*1.* $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$
3. $kd_j \leftarrow KeyGen(msk, j, Re)$	
4. $ke_i^{'} \leftarrow \mathcal{A}^{\mathcal{O}_{San}(\cdot),\mathcal{O}_S(\cdot)}(pp)$	$\mathcal{O}_{\mathcal{R}}(j, Re)$:
5. $b \leftarrow \{0, 1\}$	*1.* $kd_j \leftarrow \mathsf{KeyGen}(msk, j, Re)$
6. $c \leftarrow \mathsf{Enc}(ke_i{}', m_b)$	$\mathcal{O}_{San}(l, San)$:
7. $c' \leftarrow \underbrace{\mathsf{Sanit}(\cdots, \mathsf{Sanit}(ks_1, c))}_{t}$	*1.* $ks_l \leftarrow \mathsf{KeyGen}(msk, l, San)$
8. $b' \leftarrow \mathcal{A}^{\mathcal{O}_{San}(\cdot),\mathcal{O}_E(\cdot)}(c')$	$\mathcal{O}_E(i, msg)$:
	1. $ke_i \leftarrow \mathsf{KeyGen}(msk, i, Se)$;
	2. $c \leftarrow \mathsf{Enc}(ke_i, msg)$

Let Q_S, Q_R and Q_{San} be the set of queries issued by $\mathcal{A}$ to $\mathcal{O}_S$, $\mathcal{O}_R$ and $\mathcal{O}_{San}$ respectively. Let I_S be all the identities $i \in [u]$ such that $(i, Se) \in \mathcal{Q}_S$, J_R be the set of all identities $j \in [u]$ such that $(j, Re) \in Q_R$ and L_{San} be all identities $l \in [u + 1, u + n]$ such that $(l, San) \in Q_{San}$ respectively. We have

– $\forall i \in I_S, j \in J, P(i, j) = 0$,
– *There exists no "sanitizing pipeline" whose users are all included in L_{San}.*

If the adversary's final output $b' = b$, we say that $\mathcal{A}$ wins the Extended No-Write Rule game defined above. Let Pr[*A wins the game*] *denote the probability that $b' = b$ and $\mathsf{Adv}^{\mathcal{A}}_{Ex-No-Write}(ACE)$ denote $\mathcal{A}$'s advantage when $\mathcal{A}$ wins this* *game, then we say an ACE scheme satisfies the Extended No-Write Rule if for all PPT $\mathcal{A}$*

$$\mathsf{Adv}^{\mathcal{A}}_{Ex-No-Write}(ACE) = 2|\Pr[A \textit{ wins the game}] - \frac{1}{2}| \leq negl(\lambda)$$

In fact, we find the above two security models, the simplified no-write rule and the extended no-write rule, are considering the same security issue with only minimal differences. Namely, the former model defines one user i's no-write property in such a manner that i cannot send messages to another user j when the access policy $P(i,j) = 0$ even i, j are all corrupted or i gets help from users who also cannot send messages to j, while in the extended no-write rule model, user i's no-write property is defined similarly but with the exception that i can also gets help from at most $t-1$ sanitizers in one sanitizing pipeline rather than just from other users. Intuitively, the extended no-write rule model defined here should have already covered the simplified no-write rule model and is thus stronger than it.

3 The DACE

In this section, we firs illustrate how to construct a sanitizing cluster and how a new "sanitizing pipeline" with t sanitizers is formed when there are n sanitizers existed in the cluster, we also show you that the whole number of sanitizing pipelines and sanitizers in the sanitizing cluster can be increased in an on-demand manner. After that, we give a description of our ACE scheme with compact ciphertext size and decentralized sanitizers in detail.

3.1 The Sanitizing Cluster and Sanitizing Pipelines

We assume all sanitizers in our DACE system constitute a sanitizing clusters. Our DACE requires that only t sanitizers can collaboratively fulfill the sanitizing algorithm properly and converts one incoming ciphertext into a valid sanitized ciphertext which can then be decrypted by the intended receiver. To save the computational cost of the sanitizers in the sanitizing cluster when they do sanitization, we introduce the notion sanitizing pipeline. A sanitizing pipeline can be valued as a path predefined by the system authority containing a collection of exact t sanitizers chosen by it from the sanitizer cluster. one ciphertext can never be transformed into a valid sanitized ciphertext until it is processed by every nodes in the pipeline chosen in advance by the message sender. The system authority can actually produce as many sanitizing pipelines as it wants, and the collection of all the pipelines is represented as $\{SP\}$ which should be known by all the nodes in the ACE system. Given a polynomial $F(x)$ with degree $t-1$ such that $F(0) = y$, which is the secret to be shared. When one user with identity j wants to join the sanitizing cluster as a sanitizer, the system authority chooses $x_j \in \mathbb{Z}_p$ and computes $y_j = F(x_j)$, then y_j is allocated to this user as one of its secret, then the system authority would also produce a new sanitizing pipeline

sp_l and add this user as one member of this pipeline. Furthermore, as the authority knows all the t sanitizers in sp_l, another secret value $f_j = g^{-\prod_{i \neq j \wedge i \in sp_l} \frac{x_i}{x_i - x_j}}$ is computed by the authority in advance and then distributed to that sanitizer j. When one user with identifier j gets its own secret share (y_j, f_j) and the sanitizing pipeline identifier sp_l, it can work as a valid sanitizer member in the sanitizer cluster.

3.2 Our DACE Scheme

Our ACE scheme is defined by the following algorithms:

Setup(λ): This DACE system setup algorithm is executed by the system authority. Given the security parameter λ, a bilinear map group system $\mathcal{BM} = (p, g, \mathbb{G}, \mathbb{G}_1, e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_1)$ is generated such that $|p| = \lambda$, $g, h \in \mathbb{G}$ are two randomly selected generators of $\mathbb{G}$ and a secret value $\gamma \in \mathbb{Z}_p$ is chosen, sets $w = g^{\gamma}$. The authority also chooses a cryptographic hash function $\mathcal{H} : \{0,1\}^{\lambda} \rightarrow \mathbb{Z}_p$ which will be viewed as the random oracle in the security analysis. The authority also initializes the sanitizing clusters and sanitizing pipelines using the initialization algorithm defined above, after that, assuming there are n sanitizers and $|\{SP\}|$ sanitizing pipelines in the ACE system, notice that each element in $\{SP\}$ contains a list of sanitizers' identities and represents a unique sanitizing pipeline. Assuming there are u users which can play the role of the message sender and the message receiver, and each of them lays in one specific layer, supposing there are μ layers at most in the ACE system, let S_{L_β} denote the collection of identities of users laying in the β-th layer where $1 \leq \beta \leq \mu$, only the authority knows $\mathcal{AC} = (S_{L_1}, S_{L_2}, \cdots, S_{L_\alpha}, \cdot, S_{L_\mu})$, that is, only the authority has the knowledge of which user lays in which layer for all the u users. The authority also knows the whole sanitizing key $y \in \mathbb{Z}_p^*$. The authority defines the key space $\mathcal{KM} = \mathbb{G}_1$, the ciphertext space $\mathcal{C} = \mathbb{G}^6$, the sanitized ciphertext space $\mathcal{C}' = \mathbb{G}^6$ respectively. The public parameter $pp = (\mathcal{BM}, w, \mathcal{H}, \{SP\}, \mathcal{KM}, \mathcal{C}, \mathcal{C}')$, the master secret key $msk = (\mathcal{AC}, \gamma, y)$.

KeyGen(msk, pp, i, L_β, ty): When given pp, msk, one specific users' identity i, the layer L_β this user lays in and its user type $ty \in \{Se, Re, San\}$, the key generation algorithm is executed by the authority as follows;

– When $ty = Se$, that is, the authority needs to generate an encryption key ke_i for the user with identity i. The authority chooses $x_i \xleftarrow{R} \mathbb{Z}_p^*$ for this user with identity i and sets ke_i as;

$$ke_i = (h^{\prod_{i \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(i))}, h^{x_i \prod_{i \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(i))}, g^{-x_i \gamma}, e(g,h)^{x_i}, g^y)$$

– When $ty = Re$, that is, the authority needs to generate a decryption key kd_i for that user. When user with identity i lays in layer L_β, he can receiver messages sent from layers below its own, that is, his decryption key should be able to decrypt messages sent from layers from L_1 to L_β. Here the authority construct the decryption key of this user in such a manner that kd_i contains β

components and each component is response for decrypting ciphertexts from one specific layer;

$$\begin{aligned} kd_i &= (kd_{i0} = g^{\frac{1}{\gamma+H(i)}}, kd_{i1} = h^{\frac{\prod_{l\neq i \wedge l \in S_{L_1}\cup\cdots\cup S_{L_\mu}}(\gamma+H(l))-1}{\gamma}}, \\ kd_{i2} &= h^{\frac{\prod_{l\neq i \wedge l \in S_{L_2}\cup\cdots\cup S_{L_\mu}}(\gamma+H(l))-1}{\gamma}}, \cdots, kd_{i\beta} = h^{\frac{\prod_{l\neq i \wedge l \in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(l))-1}{\gamma}}) \end{aligned}$$

– When $ty = San$, that is, the authority needs to generate a sanitizing key ks_i for that user. To do this, the authority chooses a $m-1$ degree function $F(x)$ such that $F(0) = y$. For each sanitizer j in the specific sanitizing pipeline, denoted by sp_l, the authority allocate a $x_j \overset{R}{\leftarrow} \mathbb{Z}_p^*$ to it and computes $y_j = F(x_j)$, the sanitizing key ks_i of the sanitizer with identity i should be;

$$ks_i = g^{-y_i \prod_{j\neq i \wedge j \in SP_l} \frac{x_j}{x_j - x_i}}$$

Enc(m, ke_i, pp): Our ACE scheme borrows idea from the hybrid encryption scheme, that is, the asymmetric encryption scheme actually encrypts a symmetric encryption key, the real ciphertext is an encryption of the origin message using a symmetric key encryption scheme with the symmetric key encrypted by the previous asymmetric encryption scheme. Here, we only focus on the asymmetric part of our whole ACE and just use $SE_{sk}(m)$ to represent the symmetric encryption part. When given a message $m \in \mathcal{M}$, one message sender with identity i in layer L_β encrypts it as follows;

$$\begin{aligned} &k_0, k_1, r_s \overset{R}{\leftarrow} \mathbb{Z}_p^* \\ &C_1 = g^{-x_i\gamma k_1}, C_2 = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^{y}, \\ &C_3 = h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))}, \\ &C_4 = h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))} \\ &C_5 = e(g,h)^{x_i k_1}, C_6 = e(g,h)^{x_i k_1 r_s} \end{aligned}$$

The symmetric key should be $sk = e(g,h)^{k_0}$, the real ciphertext should be $C_7 = SE_{sk}(m)$. So, the whole ciphertext of our ACE is the tuple $\mathcal{CT} = (L_\beta, C_1, C_2, C_3, C_4, C_5, C_6, C_7)$. The message sender then chooses one sanitizing pipeline SP_l from all pipelines which are hard-wired with this sender.

Sanit($\mathcal{CT}^v, pp, ks_{lv+1}$): Given a ciphertext $\mathcal{CT}^v = (L_\beta, C_1^v, C_2^v, C_3^v, C_4^v, C_5^v, C_6^v, Ci^v)$, no matter whether it is received from the message sender or from a sanitizer's predecessor, this sanitizer does as follows;

$$\begin{aligned} &r_{v+1} \overset{R}{\leftarrow} \mathbb{Z}_p^*, C_1^{v+1} = g^{-x_i\gamma k_1}, C_2^{v+1} = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^{y} ks_{lv+1} (C_1^v)^{r_{v+1}}, \\ &C_3^{v+1} = h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))}, \\ &C_4^{v+1} = h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}}(\gamma+H(i))} (C_4^v)^{r_{v+1}} \\ &C_5^{v+1} = e(g,h)^{x_i k_1}, C_6^{v+1} = e(g,h)^{x_i k_1 r_s} (C_5^v)^{r_{v+1}}, C_7^{v+1} = C_7^v \end{aligned}$$

After this sanitizer proceeds the incoming ciphertext as above properly, it would relay the partially sanitized ciphertext to the next sanitizer laying in the same sanitizing pipeline as itself if it is not the final sanitizer in this pipeline, otherwise, this sanitizer would relay the sanitized ciphertext to the intended receiver.

Notice that all sanitizers in SP_l will do the same as what we described above. When one ciphertext tuple $\mathcal{CT} = (L_\beta, C_1, C_2, C_3, C_4, C_5, C_6)$ goes through the sanitizing pipeline SP_l and is processed by each of the t sanitizers in SP_l, the finally sanitized ciphertext should be represent as:

$$\begin{aligned}
&C_1^t = C_1 = g^{-x_i\gamma k_1}, C_2^t = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y (C_1)^{r_1+r_2+\cdots+r_t} ks_{l1} ks_{l2} \cdots ks_{lt},\\
&C_3^t = C_3 = h^{k_1 x_i \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))},\\
&C_4^t = h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} (C_3)^{r_1+r_2+\cdots+r_t}\\
&C_5^t = C_5 = e(g,h)^{x_i k_1}, C_6^t = e(g,h)^{x_i k_1 r_s} (C_5)^{r_1+r_2+\cdots+r_t}, C_7^t = C_7
\end{aligned}$$

As we can see,

$$\begin{aligned}
&ks_{l1} ks_{l2} \cdots ks_{lt}\\
&= g^{-y_{l1} \prod_{j\neq l1 \wedge j\in SP_l} \frac{x_j}{x_j - x_{l1}}} g^{-y_{l2} \prod_{j\neq l2 \wedge j\in SP_l} \frac{x_j}{x_j - x_{l2}}} \cdots g^{-y_{lt} \prod_{j\neq lt \wedge j\in SP_l} \frac{x_j}{x_j - x_{lt}}}\\
&= g^{-F(0)} = g^{-y}
\end{aligned}$$

When the last sanitizer in SP_l has executed its sanitizing algorithm on one incoming partially sanitized ciphertext, he can just send $CT' = (L_\beta, C_1', C_2', C_3', C_4')$ to the intended receivers, where

$$\begin{aligned}
C_1' &= C_2^t = g^{-x_i\gamma k_1 r_s} g^{-k_0\gamma} g^y (C_1)^{r_1+r_2+\cdots+r_t} ks_{l1} ks_{l2} \cdots ks_{lt}\\
&= g^{-(x_i k_1 (r_s + r_1 + \cdots + r_t) + k_0)\gamma}\\
C_2' &= C_4^t = h^{k_1 x_i r_s \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} h^{k_0 \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))} (C_3)^{r_1+r_2+\cdots+r_t}\\
&= h^{(k_1 x_i (r_s + r_1 + \cdots + r_t) + k_0) \prod_{i\in S_{L_\beta}\cup\cdots\cup S_{L_\mu}} (\gamma+H(i))}\\
C_3' &= C_6^t = e(g,h)^{x_i k_1 (r_s + r_1 + r_2 + \cdots + r_t)}\\
C_4' &= SE_{sk}(m) \text{ where } sk = e(g,h)^{k_0}
\end{aligned}$$

and L_β denotes the layer this message sender lays in.

Dec($kd_j, \mathcal{CT}', pp$**):** When given a properly sanitized ciphertext $\mathcal{CT}'$ and one user's decryption key kd_j, this user would first judge whether he is able to recover the origin message of the received ciphertext by checking whether the layer the receiver lays in is higher than that of the message sender. If the receiver can decrypt the ciphertext, he does as follows;

$$\begin{aligned}
sk' &= \frac{e(C_1', kd_{j\beta})e(C_2', kd_{j0})}{C_3'}, \text{ sets } K = (x_i k_1(r_s + r_1 + \cdots + r_t) + k_0) \\
&= \frac{e(g^{-K\gamma}, h^{\frac{\prod_{l \neq i \wedge l \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(l)) - 1}{\gamma}}) e(h^{K \prod_{i \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(i))}, g^{\frac{1}{\gamma + H(i)}})}{e(g,h)^{K - k_0}} \\
&= \frac{e(g,h)^{K(1 - \prod_{l \neq i \wedge l \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(l)))} e(g,h)^{K \prod_{l \neq i \wedge l \in S_{L_\beta} \cup \cdots \cup S_{L_\mu}} (\gamma + H(l))}}{e(g,h)^{K - k_0}} \\
&= e(g,h)^{k_0} \\
m' &= DE_{sk'}(C_4')
\end{aligned}$$

4 Security Proofs

Theorem 2. *Our DACE scheme holds the No-Read Rule property assuming the $(f, g, F) - GDDHE$ problem is hard in the group system $\mathcal{BM} = (p, g_0, h_0, \mathbb{G}_1, \mathbb{G}_T, e(\cdot, \cdot))$ when the hash function H is modeled as random oracle. Concretely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability ϵ, supposing $\mathcal{A}$ makes at most q_H, q_{ke}, q_{kd} queries to the H hash oracle, encryption key query oracle and decryption key query oracle respectively, then we can construct another algorithm $\mathcal{B}$ that solves the (f, g, F)-GDDHE problem in the given group system with advantage at least $\frac{1}{2} \cdot (\frac{q_H - 1}{q_H})^{q_{kd}} \cdot \frac{1}{q_H} \cdot \epsilon$, where q_H, q_{kd} are defined above.*

Theorem 3. *Our DACE scheme holds the Extended No-Write Rule property assuming the $(f, g, F) - GDDHE$ problem is hard in the group system $\mathcal{BM} = (p, g_0, h_0, \mathbb{G}_1, \mathbb{G}_T, e(\cdot, \cdot))$ when the hash function H is modeled as random oracle. Concretely, if there is an adversary $\mathcal{A}$ which can break our scheme with non-negligible probability ϵ, supposing $\mathcal{A}$ makes at most q_H, q_{ke} queries to the H hash oracle, encryption key query oracle respectively, then we can construct another algorithm $\mathcal{S}$ that solves the (f, g, F)-GDDHE problem in the given group system with advantage at least $\frac{1}{2} \cdot \frac{1}{q_H} \cdot \epsilon$, where q_H, q_{kd} are defined above.*

We omit the details of our two formal proofs there because of the page limitation.

5 Conclusion

In this paper, we present a DACE scheme to solve the privacy issues in the public blockchain. Our construction is also believed to be the first one considering using multiple sanitizers rather than one to enforce the ACE. Our extended no-write rule model and the given corresponding proof show that our DACE is more secure and reliable because of the utilization of decentralized sanitizers. We find our scheme is really suitable to be deployed in such a fully decentralized environment, for example in the public blockchain. We prove the security of our scheme under non-standard assumptions with the help of the random oracle, thus our next work focuses on presenting DACE with constant ciphertext size and decentralized sanitizers secure without random oracle and under standard assumptions.

Acknowledgement. This work is supported by the Henan Key Laboratory of Network Cryptography Technology and the Henan High Education Key Project Foundational Research Plan, the project name is "The research on access control model of cloud-based medical data", the project No. 19A520047.

References

1. Bitblender. https://bitblender.io
2. Bitlaundry. http://app.bitlaundry.com
3. Bitmixer. https://bitccointalk.org/index.php?topic=415396.160
4. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_33
5. Benaloh, J., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, New York (1990). https://doi.org/10.1007/0-387-34799-2_3
6. Bertilsson, M., Ingemarsson, I.: A construction of practical secret sharing schemes using linear block codes. In: Seberry, J., Zheng, Y. (eds.) AUSCRYPT 1992. LNCS, vol. 718, pp. 67–79. Springer, Heidelberg (1993). https://doi.org/10.1007/3-540-57220-1_53
7. Boneh, D., Gentry, C., Waters, B.: Collusion resistant broadcast encryption with short ciphertexts and private keys. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 258–275. Springer, Heidelberg (2005). https://doi.org/10.1007/11535218_16
8. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455–470. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-89255-7_28
9. Boneh, D., Waters, B., Zhandry, M.: Low overhead broadcast encryption from multilinear maps. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014. LNCS, vol. 8616, pp. 206–223. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44371-2_12
10. Buterin, V.: On public and private blockchains (2015). https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains/
11. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–88 (1981)
12. Damgård, I., Haagh, H., Orlandi, C.: Access control encryption: enforcing information flow with cryptography. In: Hirt, M., Smith, A. (eds.) TCC 2016. LNCS, vol. 9986, pp. 547–576. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53644-5_21

13. Delerablée, C.: Identity-based broadcast encryption with constant size ciphertexts and private keys. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 200–215. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76900-2_12
14. Delerablée, C., Paillier, P., Pointcheval, D.: Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In: Takagi, T., Okamoto, E., Okamoto, T., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 39–59. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73489-5_4
15. Fazio, N., Perera, I.M.: Outsider-anonymous broadcast encryption with sublinear ciphertexts. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 225–242. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_14
16. Fernández-Caramés, T.M., Blanco-Novoa, Ó., Froiz-Míguez, I., Fraga-Lamas, P.: Towards an autonomous industry 4.0 warehouse: a UAV and blockchain-based system for inventory and traceability applications in big data-driven supply chain management. Sensors **19**(10), 2394 (2019)
17. Fiat, A., Naor, M.: Broadcast encryption. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 480–491. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48329-2_40
18. Fujisaki, E.: Sub-linear size traceable ring signatures without random oracles. In: Kiayias, A. (ed.) CT-RSA 2011. LNCS, vol. 6558, pp. 393–415. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19074-2_25
19. Genkin, D., Papadopoulos, D., Papamanthou, C.: Privacy in decentralized cryptocurrencies. Commun. ACM **61**(6), 78–88 (2018)
20. Gentry, C., Waters, B.: Adaptive security in broadcast encryption systems (with short ciphertexts). In: Joux, A. (ed.) EUROCRYPT 2009. LNCS, vol. 5479, pp. 171–188. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-01001-9_10
21. Jaoude, J.A., Saadé, R.G.: Blockchain applications - usage in different domains. IEEE Access **7**, 45360–45381 (2019)
22. Kim, J., Susilo, W., Au, M.H., Seberry, J.: Adaptively secure identity-based broadcast encryption with a constant-sized ciphertext. IEEE Trans. Inf. Forensics Secur. **10**(3), 679–693 (2015)
23. Lai, J., Mu, Y., Guo, F., Susilo, W., Chen, R.: Fully privacy-preserving and revocable id-based broadcast encryption for data access control in smart city. Pers. Ubiquit. Comput. **21**(5), 855–868 (2017)
24. Libert, B., Paterson, K.G., Quaglia, E.A.: Anonymous broadcast encryption: adaptive security and efficient constructions in the standard model. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 206–224. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-30057-8_13
25. Lipmaa, H.: Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013. LNCS, vol. 8269, pp. 41–60. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-42033-7_3

26. Lu, H., Huang, K., Azimi, M., Guo, L.: Blockchain technology in the oil and gas industry: a review of applications, opportunities, challenges, and risks. IEEE Access **7**, 41426–41444 (2019)
27. Maxwell, G.: Coinjoin: Bitcoin pricacy for the real world (2013). https://en.bitcoin.it/wiki/CoinJoin
28. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: anonymous distributed e-cash from bitcoin. In: 2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, 19–22 May 2013, pp. 397–411 (2013)
29. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008). https://bitcoin.org/en/bitcoin-paper
30. Noether, S.: Ring signature confidential transactions for monero. IACR Cryptology ePrint Archive 2015, 1098 (2015)
31. Noether, S., Mackenzie, A.: Ring confidential transactions. Ledger **1**, 1–18 (2016)
32. Novo, O.: Scalable access management in iot using blockchain: a performance evaluation. IEEE Internet Things J. **6**(3), 4694–4701 (2019)
33. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
34. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (1992). https://doi.org/10.1007/3-540-46766-1_9
35. Phan, D.H., Pointcheval, D., Shahandashti, S.F., Strefler, M.: Adaptive CCA broadcast encryption with constant-size secret keys and ciphertexts. Int. J. Inf. Secur. **12**(4), 251–265 (2013)
36. Rivest, R.L., Shamir, A., Tauman, Y.: How to leak a secret. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 552–565. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45682-1_32
37. Ruffing, T., Moreno-Sanchez, P., Kate, A.: CoinShuffle: practical decentralized coin mixing for bitcoin. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 345–364. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11212-1_20
38. Salman, T., Zolanvari, M., Erbad, A., Jain, R., Samaka, M.: Security services using blockchains: a state of the art survey. IEEE Commun. Surv. Tutorials **21**(1), 858–880 (2019)
39. De Santis, A., Micali, S., Persiano, G.: Non-interactive zero-knowledge proof systems. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 52–72. Springer, Heidelberg (1988). https://doi.org/10.1007/3-540-48184-2_5
40. Shamir, A.: How to share a secret. Commun. ACM **22**(11), 612–613 (1979)
41. Shen, C., Pena-Mora, F.: Blockchain for cities-a systematic literature review. IEEE Access **PP**(99), 1 (2018)
42. Sun, S.-F., Au, M.H., Liu, J.K., Yuen, T.H.: RingCT 2.0: a compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: Foley, S.N., Gollmann, D., Snekkenes, E. (eds.) ESORICS 2017. LNCS, vol. 10493, pp. 456–474. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66399-9_25
43. Susilo, W., Chen, R., Guo, F., Yang, G., Mu, Y., Chow, Y.: Recipient revocable identity-based broadcast encryption: How to revoke some recipients in IBBE without knowledge of the plaintext. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, 30 May - 3 June 2016, pp. 201–210 (2016)

44. Tassa, T.: Generalized oblivious transfer by secret sharing. Des. Codes Crypt. **58**(1), 11–21 (2011)
45. Toyoda, K., Mathiopoulos, P.T., Sasase, I., Ohtsuki, T.: A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain. IEEE Access **5**, 17465–17477 (2017)
46. Yao, A.C.: Protocols for secure computations (extended abstract). In: 23rd Annual Symposium on Foundations of Computer Science, Chicago, Illinois, USA, 3–5 November 1982, pp. 160–164 (1982)
47. Zhang, L., Wu, Q., Mu, Y.: Anonymous identity-based broadcast encryption with adaptive security. In: Wang, G., Ray, I., Feng, D., Rajarajan, M. (eds.) CSS 2013. LNCS, vol. 8300, pp. 258–271. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03584-0_19
48. Zyskind, G., Nathan, O., Pentland, A.: Enigma: Decentralized computation platform with guaranteed privacy. Computer Science (2015)