DCAP: A Secure and Efficient Decentralized Conditional Anonymous Payment System Based on Blockchain

Chao Lin, Debiao He, Xinyi Huang, Muhammad Khurram Khan, *Senior Member, IEEE*, and Kim-Kwang Raymond Choo, *Senior Member, IEEE*

***Abstract*— Blockchain, a distributed ledger technology, can potentially be deployed in a wide range of applications. Among these applications, decentralized payment systems (e.g. Bitcoin) have been one of the most mature blockchain applications with widespread adoption. While the early designs (e.g. Bitcoin) are often the currency of choice by cybercriminals (e.g., in ransomware incidents), they only provide pseudo-anonymity, in the sense that anyone can deanonymize Bitcoin transactions by using information in the blockchain. To strengthen the privacy protection of decentralized payment systems, a number of solutions such as Monero and Zerocash have been proposed. However, completely Decentralized Anonymous Payment (DAP) systems can be criminally exploited, for example in online extortion and money laundering activities. Recognizing the importance of regulation, we present a novel definition of Decentralized Conditional Anonymous Payment (DCAP) and describe the corresponding security requirements. In order to construct a concrete DCAP system, we first design a Condition Anonymous Payment (CAP) scheme (based on our proposed signature of knowledge), whose security can be demonstrated under the defined formal semantic and security models. To demonstrate utility, we compare the performance of our proposal with that of Zerocash under the same parameters and testing environment.**

***Index Terms*— Decentralized conditional anonymous payment, blockchain, cryptocurrency, smart contract, anonymity, privacy, regulation.**

Manuscript received June 4, 2019; revised October 5, 2019; accepted January 23, 2020. Date of publication January 27, 2020; date of current version February 6, 2020. This work was supported in part by the National Key Research and Development Program of China under Grant 2017YFB0802500, in part by the National Natural Science Foundation of China under Grant 61972294, Grant 61932016, Grant 61822202, Grant 61872089, and Grant 61972094, in part by the Science and Technology Planning Project of Shenzhen under Grant JCYJ20170818112550194, and in part by the Researchers Supporting Project of King Saud University under Grant RSP-2019/12. The work of Kim-Kwang Raymond Choo was supported by the Cloud Technology Endowed Professorship. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Damien Vergnaud. *(Corresponding author: Debiao He.)*

Chao Lin is with the Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China (e-mail: linchao91@qq.com).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China, and also with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China (e-mail: hedebiao@163.com).

Xinyi Huang is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350007, China, and also with the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China (e-mail: xyhuang81@gmail.com).

Muhammad Khurram Khan is with the Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 11564, Saudi Arabia (e-mail: mkhurram@ksu.edu.sa).

Kim-Kwang Raymond Choo is with the Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249 USA, and also with the Department of Electrical and Computer Engineering, The University of Texas at San Antonio, San Antonio, TX 78249 USA (e-mail: raymond.choo@fulbrightmail.org).

Digital Object Identifier 10.1109/TIFS.2020.2969565

I. Introduction

BLOCKCHAIN is a relatively recent trend, particularly fueled by the success of Bitcoin and its capability to build a trust ecosystem for satisfying economic activities in an environment of asymmetric information and uncertain identities (e.g., due to properties such as decentralization, immutability, verifiability and programmability) [1], [2]. Cryptocurrency (also referred to as decentralized payment system) is, perhaps, the most mature Blockchain application at the time of this research, and other emerging applications include Internet of Things (IoT) and supply chain management. According to CoinMarketCap,[1] for example, there are 2,941 different cryptocurrencies on the market, with a market value of more than $219 billion (as of Oct 04, 2019).

Unlike conventional e-cash schemes such as those described in [3], [4], a decentralized payment system (e.g., Bitcoin) does not rely on trusted parties (e.g., a central bank). Such decentralized payment systems use a distributed ledger (i.e., blockchain) to record transactions instead. The blockchain is chronologically chained by a hash and largely replicated by mutually-distrustful nodes. To ensure the consistency of the blockchain ledger, transaction data (including addresses of senders and receivers, and transferred value) in early decentralized payment systems (e.g., Bitcoin [5], Ethereum [6] and Mixcoin [7]) is public. This has corresponding privacy challenges, for example in the privacy of identity and transferred value. Although these systems have in place measures (e.g., pseudonyms[2] or mixing[3]) to ensure identity privacy, an attacker can analyze transaction records in the

[1] https://coinmarketcap.com/

[2] This means that one user in the system can efficiently create and operate many different addresses to enhance their privacy.

[3] This allows users to entrust a set of coins to a pool and later retrieve different coins (with the same total value) from the pool. The pool could be either centralized or decentralized.

blockchain to build the correlation between users' addresses and even obtain the user's real identity [8], [9].

Hence, security researchers have designed privacy-enhanced solutions at the protocol level, and examples include Monero [10] and Zerocash [11]. Specifically, Monero uses ring signatures to hide the sender's address together with the one-time payment mechanism [12] to hide the relationship between the sender and receiver. One can only learn that some unknown number of coins have moved from one of these input public keys without the knowledge of the concrete one(s). In Zerocash, users can place the coins into a "shielded pool" and prove their right of spending these coins via a zero-knowledge method. It means that others can only validate the user's right of spending some specific coins in the pool, without knowing which particular coins.

However, these completely Decentralized Anonymous Payment (DAP) systems cannot be effectively regulated. In other words, such systems can be exploited for criminal activities, such as money laundering and in cybercrime cases (e.g., payment of ransom for ransomware / online extortion cases) [13]. Thus, we posit the importance of designing a decentralized payment system that strikes a balance between achieving reasonable privacy protection and allowing regulation to prevent abuse / criminal exploitation. There have also been research along this line. For example, Wu *et al.* [14] proposed a DAP system that takes into consideration regulation. That is, their proposal not only supports users to anonymously conduct transactions, but they also introduce an audit department to monitor the transaction records. The adopted blind signatures and key derivation mechanism ensure that their proposal achieves transferability, anonymity, and double-spending resilience. However, the construction in [14] requires considerable interaction (in total about 14 times) during the mining, transferring, withdraw phases. In addition, the proposal also requires 1 blind signatures, 4 digital signatures, and 5 asymmetric encryptions. In other words, the complex and expensive communication and computing costs make it challenging to adopt such a system in practice.

A. Paper Contributions

Seeking to address the limitations identified above, we introduce an efficient Decentralized Conditional Anonymous Payment (DCAP) system to strike a balance between privacy protection and regulation. Note that, in this paper, we do not over emphasize the property of decentralization, for our DCAP is realized in a permissioned blockchain. Hence, our system allows the existence of some trusted entities. Concretely, we first focus on the reconstruction of a signature used in transactions to achieve conditional anonymity, and denote a Conditional Anonymous Payment (CAP) scheme with formal semantic and security models. We also give a concrete design of this primitive and prove its security. Then, we provide a concrete construction of the DCAP system (upon the designed CAP scheme) and discuss how the proposal can satisfy such security requirements. Finally, we evaluate and compare the performance of the above primitives with those of Zerocash under the same parameters and testing environment, and implement the designed smart contract in a private Ethereum chain to show its feasibility. The findings demonstrate the viability of DCAP in realizing a decentralized payment system that achieves both practical privacy protection and facilitating regulation.

B. Paper Outline

In Section II, we briefly review the extant literature on DAP systems. In Section III, we present the system model with relevant security requirements, as well as the cryptographic primitives. Next, we present our proposed CAP scheme and its security proofs in Section IV. In Sections V, VI and VII, we present the construction of our DCAP, its security analysis and performance analysis, respectively. Section VIII concludes this paper.

II. Related Literature

In Bitcoin, users could issue transactions under their pseudonyms, which provides only simple unlinkability. Since all Bitcoin transactions are publicly available, users' real identities can be revealed under attacks such as network analysis [15], address clustering [16], and transaction graph analysis [8].

To provide privacy-enhanced solutions for decentralized payment systems, mixing technology has been proposed. It usually depends on shuffling the relationships among transaction inputs and outputs. However, known limitations of such an approach include theft of digital assets and disclosure of user privacy by some malicious central mixing node. Bonneau *et al.* [7] proposed Mixcoin (an improved mixing scheme), attempting to apply signature auditing techniques into tracking violations by the central node. Dash [17] also attempted to solve the privacy leakage problem from the perspective of economics, where the central mixing node is responsible for merging multiple transactions (of the same amount) and any unauthorized / illegal operations will result in penalty (i.e., forfeiting the prepaid deposit).

However, in the discussed solutions, the mixing node knows the relationships among users' addresses and may compromise these users' privacy. Hence, Valenta and Rowan [18] used blind signatures to construct Blindcoin to ensure that the central mixing node only provides mixed currency services and does not have the capability to link the input and output addresses. However, this mechanism requires that the mixed amount is fixed and users need to send the output address to the public log anonymously. Such a mechanism cannot avoid third-party cheating. In addition, these central mixing schemes suffer from limitations such as long waiting delay, additional costs incurred in the mixing, and single point failure of mixed currency nodes. Therefore, decentralized mixing technologies were introduced, and examples include CoinShuffle [19], XIM [20], and CoinParty [21].

Although the above decentralized mixing schemes have relatively simple operations and provide a certain degree of anonymity, there are limitations such as longer waiting delay and higher computational complexity. Thus, solutions based on cryptographic tools have also been proposed in

the literature. Monero, for example, used ring signatures and one-time payment mechanism to achieve anonymity (as previously discussed in Section I). However, Foley *et al.* [22] pointed out that attackers can still determine the relationship between transaction address and the real identity when the anonymous set in the ring signature is small. A large anonymous set could enhance the anonymity, but it will result in a significant increase in the storage costs for transactions. Therefore, Sun *et al.* [23] adopted accumulator tools to design an efficient ring signature mechanism (i.e., RingCT 2.0), seeking to balance both efficiency and privacy concerns.

To mitigate the requirements for significant storage space and redundant addresses in Monero, Miers *et al.* [24] proposed a DAP scheme (i.e., Zerocoin). This scheme uses the commitment to encapsulate the source and the destination of a transaction, and zero-knowledge proof to prove the transactions without revealing their correlation. However, Zerocoin can only be minted and exchanged for fixed denominations, and participants who have taken over the transaction are able to track its flow. As a countermeasure, Sasson *et al.* [11] utilized zero-knowledge succinct non-interactive argument of knowledge (zk-SNARK) in their proposed solution, which can efficiently hide the identities of participants in a transaction, and support transactions of any denomination. However, limitations of the proposed solution include the need to generate system parameters based on trusted nodes, the slow proving process, and lack of any regulation / auditing capability.

Fauzi *et al.* [25] focused on the limitation that addresses in some DAP schemes (e.g., Monero and Zerocash) cannot be removed for unknown balance in these addresses. Specifically, the authors leveraged updatable keys with efficient zero-knowledge arguments to design Quisquis, which not only provides anonymity and hiding of the actual amount, but also effectively deletes addresses of zero amount. Although Quisquis can reduce the storage cost from the perspective of addresses, the involved interactive zero-knowledge proofs during the address replacement will incur considerable storage and communication costs (since the length of proofs is linearly related to the size of the anonymous set). There have also been other efforts focusing on the design of DAP [26]–[28], but few consider regulation (which is necessary to minimize abuse / criminal exploitation).

More recently in 2019, Wu *et al.* [14] considered regulation in their design of a DAP system. Specifically, they proposed a concrete system based on blind signatures and key derivation mechanism, which achieves the transferability, anonymity, and double-spending resilience properties. However, the implementation of their proposal requires considerable communication and computing costs (as previously discussed in Section I); thus, limiting its practicality.

III. Preliminaries

In this section, we present our proposed system model for a DCAP system and the related security requirements, as well as the cryptographic primitives (i.e. Signature of Knowledge and Conditional Anonymous Payments).

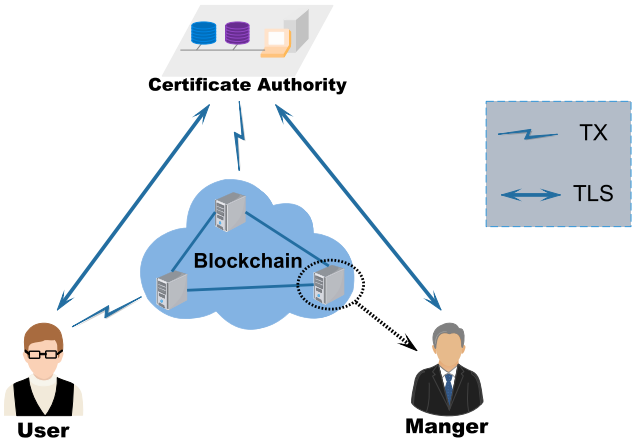


Fig. 1. The system model of decentralized conditional anonymous payments.

A. System Model

There are four entities in our proposed DCAP, that is, Certificate Authority (CA), User, Manager, and Blockchain Network (BN) (see Figure 1). The communication methods among them include TX and Transport Layer Security (TLS), where the former is an on-chain communication (i.e., publishing a transaction using P2P communications) and the latter is an off-chain communication.

- *CA:* This entity is a trusted third party who is responsible for managing the certificates of Users' or Managers' public keys. Concretely, the CA first issues these certificates by embedding them into the transactions (i.e., via the TX communication). Then, it builds the relationships between the issued certificates of public keys and its transaction identity in the smart contract. This can help others to conveniently retrieve the relevant certificates from the blockchain, and the CA to efficiently update the certificates of all the participants (including its own). Note that, the existing public key certificate system (e.g. X.509) can be also used in our DCAP system, for which is independent of our system design.
- *User:* These entities are the main participants of a payment system (e.g., Bitcoin and Ethereum), who own their respective accounts (each account comprises an address and a private key). The address is the User's identification and the private key is used to transfer coins from one address to another address. Note that each User owns its long-term address and can derive as many anonymous addresses as they wish from this long-term address. The concrete function (i.e. Update) will be introduced in Section III-C.2.
- *Manager:* These entities are all trusted third parties, who are responsible for tracing the real identity of a suspicious transaction.[4] Note that the Manager is the only entity who owns this authority to invert an anonymous address to the original long-term one. In addition, these entities

[4]Anomaly detection can be used in cryptocurrencies to decide a transaction suspicious or not. Specifically, tracking and monitoring liquidity movement within and across blockchains could be used for detecting a suspicious transaction by monitoring a set of addresses and where money is going.

maintain a common blockchain and own the right to allow or restrict another user to join the system. They can also publicly reveal the malicious users' long-term addresses in the smart contract; thus, resulting in a blacklist (i.e., these malicious users can no longer successfully issue transactions).

- *BN:* This entity refers to the Blockchain Network, which provides an immutable, undeniable and verifiable data storage. The data structure is with a chain-based storage (i.e. the blockchain) comprising so-called transactions. In our proposal, it is realized as a permissioned blockchain, which is maintained by some permissioned nodes (i.e. Managers in our system). The public certificates are embedded into the transactions for the subsequent retrieval.

B. Security Requirements

According to existing literature [14], [29], a DCAP system needs to satisfy the following fundamental security requirements.

1) *Single Registration:* For practical (independent of security) requirement, participants (e.g., User or Manager) should only need to register once to join the DCAP (i.e., issuing transactions or managing the authorities).
2) *Transaction Anonymity:* To preserve the privacy of honest users, the DCAP should achieve anonymity, in the sense that any other entity (apart from the Manager) is not able to determine users' real identity by analyzing the transactions.
3) *Traceability:* Traceability should be provided, where an authorized Manager (with the tracing key) can trace a malicious user's real identity.
4) *Interception and Modification Resilience:* The integrity of data (e.g., transferred value, participants' anonymous addresses and transmitted message) involved in the transactions should be protected from interception and modification (under the undiscovered modifications).
5) *Double-Spending Resilience:* The DCAP should prevent attackers from spending the same single digital token or coin more than once.
6) *Birthday Collision Resilience:* The DCAP should minimize the possibility of generating two same blocks at the same time. That is, the DCAP should resist block birthday collision to avoid potential disputes between sub-blockchains.
7) *Hijacking Resilience:* The DCAP should prevent attackers from hijacking transactions to ensure a smooth transaction.
8) *Network Analysis Resilience:* The DCAP should prevent attackers from analyzing the network history of publicly announced transactions to trace the profiles corresponding to addresses.
9) *Resilience to Other Attacks:* The DCAP should also prevent attackers from successfully carrying out common attacks, such as 51% attacks, impersonation attacks, and distributed denial of service attacks.

C. Cryptographic Primitives

1) Signatures of Knowledge: Signatures of Knowledge [30], a "proof system", allows one party to convince others about its knowledge of certain values without revealing any useful information. In addition to proving knowledge, it also serves as a signature (which is different from other proofs of knowledge tools). All these signatures of knowledge can be proven secure in the random oracle model and its interactive version is zero-knowledge (given that several rounds with small challenges are used). It consists of the following algorithms.

- $Gen(1^{\lambda})$*:* This algorithm takes as input a security parameter λ, and outputs a public parameter PP.
- $SPK(PP, m, x, R)$*:* This algorithm takes as input public parameter PP, message m and a relation $(x, R) \in \mathcal{R}$, it returns a signature of knowledge π.
- $Verf(PP, m, \pi)$: This algorithm takes as input public parameter PP, message m and signature of knowledge π, it returns 1 if the π is valid; or 0 otherwise.

Definition 1: A pair $(c, s) \in \{0,1\}^k \times \mathbb{Z}_q^*$ satisfying $c = \mathcal{H}(m||R||P||(sP + cR))$ is a signature of knowledge of the discrete logarithm of the element $R \in \mathbb{G}$ to the base P on the message m, and it is denoted as $SPK(x : R = xP)(m)$.

$SPK(x : R = xP)(m)$ can be computed if one knows the secret key $x = log_P(R)$, by randomly choosing $r \in \mathbb{Z}_q^*$ and computing c (i.e. challenge) and s (i.e. challenge-response) as $c = \mathcal{H}(m||R||P||rP)$ and $s = r - cx \ (mod\ q)$, where rP is the commitment to prove knowing the knowledge of $x = log_P(R)$ and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is a strong collision-resistant hash function.

In our proposal, we use this technique to construct a signature of the knowledge of a discrete logarithm to build signatures that involve a more complex statement (i.e., a pair (x, y) satisfying $Q' = xQ, Q'' = xyT + Q, Q = yP$, where $T \in \mathbb{G}$).

Definition 2: A pair $(c, s = (z_1, z_2, R_2)) \in \{0,1\}^k \times \mathbb{Z}_q^*$ satisfying $c = \mathcal{H}(m||Q'||Q''||T||P||(z_1P + cQ')||R_2||(z_2P + cQ'' - R_2 + z_1T))$ (where $T \in \mathbb{G}$) is a SPK of the statement involving in our proposal, and it is denoted as $SPK((x, y) : (Q' = xQ) \wedge (Q'' = xyT + Q) \wedge (Q = yP))(m)$.

$SPK((x, y) : (Q' = xQ) \wedge (Q'' = xyT + Q) \wedge (Q = yP))(m)$ can be computed if one knows the secret key-pair (x, y) (where $x = log_Q(Q'), y = log_P(Q), xy = log_T(Q'' - Q)$), by randomly choosing $a, b \in \mathbb{Z}_q^*$ and computing c and s as $c = \mathcal{H}(m||Q'||Q''||T||P||abP||R_2||bP)$ and $s = (z_1, z_2, R_2)$, where $z_1 = ab - cxy, z_2 = b - cy, R_2 = abT$.

Theorem 1: The above signature of knowledge is a NIZK argument in the random oracle model. It achieves perfect completeness, soundness and special honest-verifier zero-knowledge.

We provide a proof of Theorem 1 in Appendix VIII.

2) Conditional Anonymous Payments: To obtain the property of conditional anonymity, we focus on the reconstruction of a signature used in transactions. That is, we denote a CAP with formal semantic and security models. The functions of a CAP include proving the ownership of a coin without revealing the real identity in legal transactions, and tracing real identities in suspicious transactions. A CAP scheme consists of

six probabilistic polynomial time ($\mathcal{PPT}$) algorithms (namely, CAP = Setup, Enroll, Update, Execute, Verify, Trace). Note that in our definitions, we use the signature of knowledge proof system (P, V) to generate a signature.

- $PP \leftarrow \mathsf{Setup}(1^\lambda)$: This algorithm takes as input a security parameter λ and returns system public parameters PP.
- $(pk_u, sk_u) \leftarrow \mathsf{Enroll}(PP)$: This algorithm is invoked when some user wishes to join the CAP system, creating its account. It takes as input the public parameters PP, and returns a public / private key-pair (pk_u, sk_u) (where pk_u is regarded as the long-term address). Note that the manager also invokes this algorithm to obtain its account (i.e., (pk_m, sk_m)).
- $(apk_u, ask_u) \leftarrow \mathsf{Update}(PP, pk_u, sk_u, pk_m)$: This algorithm is invoked by a user to obtain an anonymous account. It takes as input the public parameters PP, its long-term address pk_u, private key sk_u and the manager's long-term address pk_m, and returns an anonymous account (apk_u, ask_u).
- $tx \leftarrow \mathsf{Execute}(PP, pk_a, sk_a, pk_m, apk_b, \upsilon)$: This algorithm is invoked when some user, say Alice, transfers υ coins to another user, say Bob; thus, creating a conditional privacy-preserving transaction. It takes as input the public parameters PP, Alice's account (pk_a, sk_a), the manager's long-term address pk_m, Bob's anonymous address apk_b and the transferred value υ. First, it invokes the Update algorithm to obtain Alice's anonymous account (apk_a, ask_a). Then, the anonymous account is used to compute a signature of knowledge on the transferred context, that is, $\pi \leftarrow P(PP, x, w)$, where $x = (apk_a, apk_b, pk_m, \upsilon)$, $w = ask_a$. Finally, it returns the transaction $tx = (x, \pi)$.
- $\{0, 1\} \leftarrow \mathsf{Verify}(PP, tx = (x = (apk_a, apk_b, pk_m, \upsilon), \pi))$: This algorithm is invoked to verify whether a transaction (e.g., transferring υ coins from Alice to Bob) is valid. It takes as input the public parameters PP, a candidate transaction tx (including Alice and Bob's anonymous addresses (apk_a, apk_b), the manager's long-term address pk_m, the transferred value υ, and the proof π). If $V(PP, x, \pi) = 1$ then it returns 1 indicating the transaction is valid; otherwise, it returns 0.
- $pk_u \leftarrow \mathsf{Trace}(PP, sk_m, apk_u)$: This algorithm is invoked by the manager to trace the long-term address of a transaction issuer. It takes as input the public parameters PP, the manager's private key sk_m and the anonymous address apk_u in a transaction, and returns the long-term address pk_u corresponding to apk_u.

Completeness: This property requires that the unspent coins can be spent and the spent coins can be traced by the manager. It means that the valid transactions generated by Execute must be accepted (i.e., Verify returns 1) and traced (i.e., Trace returns the long-term time address of any participant in this transaction). Note that here we do not consider the double spend situation, as this is prevented by the consensus mechanism. In other words, we only focus on the correctness requirements of the algorithms in CAP. That is, for all $\lambda \in \mathbb{N}$,

$$Pr\left(\begin{array}{l} PP \leftarrow \mathsf{Setup}(1^\lambda), (pk_a, sk_a) \leftarrow \mathsf{Enroll}(PP), \\ (pk_m, sk_m) \leftarrow \mathsf{Enroll}(PP), \\ (x = (apk_a, apk_b, pk_m, \upsilon), \pi) \\ \quad \leftarrow \mathsf{Execute}(PP, pk_a, sk_a, pk_m, apk_b, \upsilon) : \\ \mathsf{Verify}(x, \pi) = 1 \wedge \mathsf{Trace}(PP, sk_m, apk_a) = pk_a \wedge \\ \mathsf{Trace}(PP, sk_m, apk_b) = pk_b \end{array}\right) = 1.$$

Security: The security of a CAP scheme is characterized by three properties, namely: *transaction anonymity*, *transaction non-malleability* and *transaction traceability*.

Transaction Anonymity: This property guarantees that the ledger and transaction reveal no new information to the adversary, except publicly available information (e.g., the number of transferred coins, and anonymous addresses). That is, no $\mathcal{PPT}$ adversary $\mathcal{A}$ can distinguish between two transactions, where the public keys are chosen by $\mathcal{A}$.

Definition 3: A CAP scheme Π = (Setup, Enroll, Update, Execute, Verify, Trace) satisfies transaction anonymity if no $\mathcal{PPT}$ adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ can distinguish the following experiments with a non-negligible probability.

$\mathrm{EXP}^{\mathrm{b}}_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda)$:
$PP \leftarrow \mathsf{Setup}(1^\lambda), (pk_m, sk_m) \leftarrow \mathsf{Enroll}(PP),$
$(pk_2, sk_2) \leftarrow \mathsf{Enroll}(PP),$
$(apk_2, spk_2) \leftarrow \mathsf{Update}(PP, pk_2, sk_2, pk_m),$
$((pk_0, sk_0), (pk_1, sk_1), state) \leftarrow \mathcal{A}_1^{\mathsf{Enroll}(PP)},$
$(x_b, \pi_b) \leftarrow \mathsf{Execute}(PP, pk_b, sk_b, pk_m, apk_2, \upsilon),$
$b' \leftarrow \mathcal{A}_2(x_b, \pi_b, state).$

That is,

$$|\Pr[\mathrm{EXP}^0_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{EXP}^1_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

Transaction Non-Malleability: To prevent the transactions (i.e., transferred value and participants' anonymous addresses) from being modified, CAP should ensure non-malleability in this transaction. That is, no $\mathcal{PPT}$ adversary $\mathcal{A}$ can modify others' transactions before they are recorded into the decentralized ledger.

Definition 4: A CAP scheme Π = (Setup, Enroll, Update, Execute, Verify, Trace) satisfies transaction non-malleability if no $\mathcal{PPT}$ adversary $\mathcal{A}$ can win the following experiment with a non-negligible probability.

$\mathrm{EXP}_{\mathrm{nm}}(\Pi, \mathcal{A}, \lambda)$:
$(PP, td) \leftarrow \mathsf{Sim}(1^\lambda), (pk_m, sk_m) \leftarrow \mathsf{Enroll}(PP),$
$(pk_a, sk_a) \leftarrow \mathsf{Enroll}(PP), (pk_b, sk_b) \leftarrow \mathsf{Enroll}(PP),$
$(apk_b, ask_b) \leftarrow \mathsf{Update}(PP, pk_b, sk_b, pk_m),$
$(x^*, \pi^*) \leftarrow \mathcal{A}^{\mathsf{SimExecute}}(\cdot)(PP, td, pk_a, pk_m, apk_b, \upsilon).$

Denote Q as the list of instances and corresponding transactions given by SimExecute[5] upon $\mathcal{A}$'s queries. We say

[5]SimExecute is similar to the Execute algorithm, except that the transactions are generated by the simulator.

that $\mathcal{A}$ wins if (i) $apk_a \in x^*$[6]; (ii) $(x^*, \pi^*) \notin Q$; (iii) $\mathsf{Verify}(PP, x^*, \pi^*) = 1$.

Transaction Traceability: A conditional anonymous payment scheme should also provide transaction traceability to trace the sender's long-term address of a malicious transaction. We say the CAP scheme has this property if there is a traceability algorithm Trace such that

$$\left| Pr\left(\begin{array}{l} PP \leftarrow \mathsf{Setup}(1^\lambda);\ (pk_m, sk_m) \leftarrow \mathsf{Enroll}(PP), \\ \forall (x_k, \pi_k) \leftarrow \mathsf{Execute}(PP,\ pk_{k_i},\ pk_m, apk_{k_j}, \upsilon_k), \\ x_k = (apk_{k_i}, apk_{k_j},\ pk_m, \upsilon_k), i, j, k \in \mathbb{N} : \\ \mathsf{Trace}(PP, sk_m, apk_{k_l}) = pk_{k_l}, l \in \{i, j\} \end{array} \right) \right| = 1.$$

IV. Proposed CAP Scheme

To build a concrete CAP scheme for designing the DCAP system, we use a self-updatable pseudonym based method [31] (which provides the traceability function) together with a constructed signature of knowledge (as defined in Section III-C.1, which proves owning of the private key for the updated anonymous address without revealing any other information) on a transferred value $\upsilon \in [0, 2^{32})$.

- Setup: System parameters involved in our construction are $\{E, \mathbb{G}, P, p, q,\ a, b, \mathcal{H}\}$, where p, q are two large prime numbers, E is a non-singular elliptic curve defined by $y = x^3 + ax + b\ mod\ p\ (a, b \in \mathbb{F}_p)$, $\mathbb{G}$ is a cyclic group which consists of all points on E, as well as the point at infinity O, P is the generator of $\mathbb{G}$ with order q, and $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$ is a secure cryptographic hash function.
- Enroll: This algorithm randomly chooses $d_u \in \mathbb{Z}_q^*$ and computes $Q_u = d_u P$. The long-term address is Q_u and the private key is d_u.
- Update: Given a public / private key pair (Q_u, d_u) and the manager's long-term address Q_m, this algorithm randomly chooses $r_u \in \mathbb{Z}_q^*$ and computes $Q'_u = r_u Q_u$, $Q''_u = (r_u d_u) Q_m + Q_u$. The anonymous address is (Q'_u, Q''_u) and the corresponding private key is (r_u, d_u).
- Execute: Given a sender's public / private key pair (Q_a, d_a), the manager's long-term address Q_m, a receiver's anonymous address (Q'_b, Q''_b) and the transferred value υ, this algorithm first invokes the Update to obtain $((Q'_a, Q''_a), (r_a, d_a)) \leftarrow \mathsf{Update}(Q_a, d_a, Q_m)$. Then, it randomly chooses $r, s \in \mathbb{Z}_q^*$ and computes $R_1 = rsP, R_2 = rsQ_m, R_3 = sP$. Finally, it generates the transaction $tx = (x = (\upsilon, Q'_a, Q''_a, Q'_b, Q''_b, Q_m, P),\ \pi = (c, z_1, z_2, R_2))$ via computing $c = \mathcal{H}(\upsilon||Q'_a||Q''_a||Q'_b||Q''_b||Q_m||P||R_1||R_2||R_3)$, where $z_1 = rs - cr_a d_a, z_2 = s - cd_a$.
- Verify: Given the candidate transaction $tx = (x, \pi)$, this algorithm first parses x as $(\upsilon, Q'_a, Q''_a, Q'_b, Q''_b, Q_m, P)$ and π as (c, z_1, z_2, R_2). Then, it computes $R'_1 = z_1 P + cQ'_a, R'_3 = z_2 P + cQ''_a - R_2 + z_1 Q_m$ and $c' = \mathcal{H}(\upsilon||Q'_a||Q''_a||Q'_b||Q''_b||P||R'_1||R_2||R'_3)$, before attempting to verify whether $c = c'$ holds. If it returns 1, then the transaction is valid; otherwise, it is invalid.
- Trace: Given the manager's private key $sk_m = d_m$ and the candidate anonymous key (Q'_u, Q''_u), this algorithm computes $Q''_u - d_m Q'_u$ to obtain the original long-term address Q_u.

Theorem 2: The tuple $\Pi =$ (Setup, Enroll, Update, Execute, Verify, Trace) is a secure CAP scheme, that is, it satisfies the properties of *completeness*, *transaction anonymity*, *transaction non-malleability* and *transaction traceability*.

Proof: To prove *completeness*, given a transaction $tx = (x, \pi)$ generated by Execute (where $x = (\upsilon, Q'_a, Q''_a, Q'_b, Q''_b, Q_m, P), \pi = (c, z_1, z_2, R_2)$), we have $R'_1 = z_1 P + cQ'_a = (rs - cr_a d_a)P + cr_a d_a P = rsP = R_1$, $R'_3 = z_2 P + cQ''_a - R_2 + z_1 Q_m = (s - cd_a)P + c(r_a d_a)Q_m + cQ_a - rsQ_m + (rs - cr_a d_a)Q_m = sP = R_3$. Hence, the equation $\mathcal{H}(\upsilon||Q'_a||Q''_a||Q'_b||Q''_b||P||R'_1||R_2||R'_3) = c$ holds, meaning that the Execute is perfectly completed with Verify. If $Q''_a - d_m Q'_a = (r_a d_a)Q_m + Q_a - d_m r_a Q_a = Q_a$ also holds, then this implies that Execute is perfectly completed with Trace. It is clear that the latter completeness also implies *transaction traceability*.

To prove *transaction anonymity*, we will now describe the following hybrid experiments ($\mathrm{EXP}^0_{\mathrm{an}}$, E_1, E_2, $\mathrm{EXP}^1_{\mathrm{an}}$).

- *Experiment* E_1*:* The experiment E_1 modifies $\mathrm{EXP}^0_{\mathrm{an}}$ by using the simulator $\mathcal{S}$ (see Algorithm 2 in Appendix VIII) to generate the proof when invoking the Execute algorithm. More formally, the challenger produces the argument $tx_0 = (x_0, \pi_0) \leftarrow \mathcal{S}(1^\lambda)$ via $\mathcal{A}$ instead of invoking the SPK generation algorithm when executing the Execute algorithm. Due to the special honest verifier zero-knowledge of adopted SPK scheme, the distribution of the simulated proof is identical to that in $\mathrm{EXP}^0_{\mathrm{an}}$; hence, we have

$$|\Pr[\mathrm{EXP}^0_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{E}_1(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

- *Experiment* E_2*:* The experiment E_2 modifies E_1 when invoking the Execute algorithm. The challenger simulates an instance x_1, which is the same as that in $\mathrm{EXP}^1_{\mathrm{an}}$ and computes $tx_1 = (x_1, \pi_1) \leftarrow \mathcal{S}(1^\lambda)$.
We also have $\{x_0, \pi_0\} \overset{c}{\approx} \{x_1, \pi_1\}$[7] for the proposed SPK, which is a special honest verifier zero-knowledge. Hence, the following is satisfied.

$$|\Pr[\mathrm{E}_1(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{E}_2(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

Due to the zero-knowledge property, it is clear that

$$|\Pr[\mathrm{E}_2(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{EXP}^1_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

As discussed above, we conclude the entire probability that the adversary $\mathcal{A}$ can distinguish the hybrid experiments. That is,

$$|\Pr[\mathrm{EXP}^0_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1] - \Pr[\mathrm{EXP}^1_{\mathrm{an}}(\Pi, \mathcal{A}, \lambda) = 1]| \leq \mathrm{negl}(\lambda).$$

[6]Undoubtedly, $\mathcal{A}$ can generate valid transactions that are unrelated to those x_i's, but requiring that the generated x^* by $\mathcal{A}$ contains the same sender information. In other words, this captures the property of non-malleability.

[7]We denote $X \overset{c}{\approx} Y$ as the two distribution ensembles X and Y are computational indistinguishable.

To prove *transaction non-malleability*, we assume that there exists an adversary $\mathcal{A}$ that could successfully attack this property with a non-negligible probability. It means that $\mathcal{A}$ can generate a new instance and a valid proof (i.e., (x, π)), which it cannot previously prove. This claims that the adopted SPK scheme is malleable (i.e., after a view of some SPK proofs, one could prove some new instances without seeing their corresponding proofs). However, as shown in [32], the simulation sound extractability[8] of a NIZK scheme implies non-malleability. Moreover, Faust *et al.* [33] also proved that the NIZK scheme based on Fiat-Shamir heuristic achieves simulation-sound extractability. This implies that our proposed SPK scheme (which is also a NIZK scheme) also inherits such a property; hence, contradicting the previous hypothesis.

V. Proposed DCAP System

In this section, we will present our construction of the DCAP system based on the CAP scheme presented in the preceding section. In our DCAP system, we need some trusted nodes (i.e., Managers discussed in III-A) to manage the authority of users and also trace the real identity of malicious users. Hence, we suggest using a permissioned type for our DCAP (based on the PBFT consensus mechanism [34]). This allows us to achieve enhanced privacy and support hundreds of thousands transactions per second; thus, meeting the design requirements of a regulatory cryptocurrency system. Before presenting the system design, here we first introduce the transaction format and the smart contract design.

A. Transaction

Similar to Bitcoin, the transaction in our DCAP system is also a UTXO-based model. That is, each transaction output chained in the blockchain can be classified either as an *unspent transaction output* (UTXO) (if it has been referenced by another transaction input recently) or a *spent transaction output* (STXO) otherwise. Specially, a DCAP transaction consists of general and optional fields (as shown in Figure 2). General fields include a list $\overrightarrow{inputs} = (in_1, in_2, \ldots, in_n)$ (i.e., a list of sender's addresses) of transaction inputs, a list $\overrightarrow{v_{in}} = (vi_1, vi_2, \ldots, vi_n)$ of input coins, a list $\overrightarrow{outputs} = (out_1, out_2, \ldots, out_m)$ of transaction outputs, a list $\overrightarrow{v_{out}} = (vo_1, vo_2, \ldots, vo_m)$ of output coins, and the SPK proofs $\overrightarrow{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ for proving the ownerships of input coins. Note that the addresses in $\overrightarrow{inputs}$ and $\overrightarrow{outputs}$ are anonymous, and the concrete anonymization (i.e. invoking the Update algorithm). Optional fields include $init$ and $data$, which are padded to create and invoke a contract respectively. Concretely, the field $init$ is padded with the EVM-code when a transaction is a contract creation transaction (i.e., $\overrightarrow{outputs} = \varnothing$ in this case). The contract will own its address after it is created. One can invoke the contract (i.e., performing message calls) by padding $\overrightarrow{outputs}$ with the goal contract address and $data$ with input data (i.e. parameters) of the message call.

[8]Simulation sound extractability represents that even after seeing many simulated proofs, we can extract a witness whenever the adversary generates a new proof.

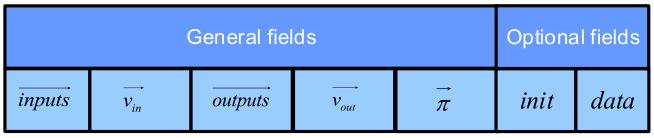


Fig. 2. Format of a DCAP transaction.

In addition to the above general transaction, there are two other transaction types for invoking contracts, namely: "eth_calls" (executed by the local node) and "internal transactions" (invoked between each other's contracts) [6]. These two types are not recorded in the blockchain and hence, reduces the storage cost and hides the calling results (not disclosed in the blockchain). In our proposal, these three types are used to transfer coins, create and trigger smart contracts, which are necessary for the realization of DCAP.

B. Smart Contract

In our DCAP, we mainly use smart contract to conveniently retrieve certificates (i.e., mapping User's public key to the transaction identity of its certificate in the blockchain) and efficiently verify the authority (i.e., using a blacklist to identify revoked users). Key functions include updateTx (only the CA can execute it to map a new transaction identity to the corresponding public key), getTx (can be invoked by anyone to obtain the transaction identity of a queried public key), deleteTx (similar to updateTx, but for deleting existing mapping in order to revoke some user's authority), updateBList (only the Manger can execute it to maintain the blacklist), and isLegal (invoked by Manger to determine whether an address has been revoked).

Here, getTx and isLegal are *view* type of functions, meaning that their invocation does not require any *gas* consumption and transaction confirmation. This reduces the communication delay in the DCAP system. This smart contract also uses the deleteTx algorithm to realize the function of certificate revocation, which cooperates with the updateBList to efficiently manage the user's authority. That is, one's mapping (between its public key and corresponding transaction identity) will be deleted by the CA once its address is blacklisted by the Manager, and hence the revoked user will not be able to publish a transaction anymore. The design of our contract is briefly presented in Algorithm 1.

C. System Design

Our DCAP system design consists of five phases (as shown in Figure 3), namely: **System Initialization**, **Register**, **Transaction Issuance**, **Chain Transaction**, and **Permission Update**. We do not adopt the consensus mechanism (e.g., PoW in Bitcoin) to mine a coin but PBFT for achieving consensus, as we use monetary mechanism similar to those in formal banking systems where trusted participants (e.g., Managers in our DCAP) are responsible for the coin adjustment (i.e., issuing or destroying coin). Here, we will not elaborate on the procedure of coin adjustment but only consider the transferring of coins.

Algorithm 1 Part 1 - Smart Contract on DCAP

Require: Function name, invoked parameters
Ensure: *Setting up functions:*
address CA; *% Define the address of CA*
address $Manager$; *% Define the address of Manger*
mapping (address → uint256) *public* $PK2TX$;
mapping (address → bool) *public* $bList$;
function DCAP()
% Constructor, automatically invokes when this smart contract is deployed.
$CA = msg.sender$; *% Define the deployer as the CA*
$Manager = manager$; *% Initiate the address of the Manager*
function updateTx(address $user$, uint256 $txid$) *public* returns (address $addr$)
% Invoked by CA to map a transaction identity to a public key.
require($msg.sender == CA$); *% Only the CA can update the map*
$PK2TX[user] = txid$;
function getTx(address $user$) *view* returns ($txid$)
% Invoked by any to retrieve a transaction identity to the required public key.
return $PK2TX[user]$;
function deleteTx($target$) *public* returns ($txid$)
% Invoked by CA to delete the target mapping
require($msg.sender == CA$); *% Only the CA can successfully delete the existing mapping*
delete $PK2TX[target]$;
function updateBList(address $user$, bool $state$) *public* returns (address $addr$)
% Invoked by Manger to maintain the bList.
require($msg.sender == Manger$); *% Only Manger can blacklist an address.*
$bList[user] = state$;
function isLegal(address $user$) *view* returns (bool $state$)
% Invoked by Manager to judge an address is revoked or not.
return $bList[user]$;

- *Initialization:* This phase mainly initializes the following system parameters:
  1) *Blockchain Initialization:* The managers determine the public system parameters $\{\mathbb{E}, G, P, p, q, a, b, \mathcal{H}\}$ using the Setup algorithm. These parameters will be embedded into the first block (i.e., the genesis block).
  2) *Contract Deployment:* According to the system parameters in the genesis block, the CA can generate its long-term account ($pk_{ca} = Q_{ca}, sk_{ca} = d_{ca}$) via the Enroll algorithm. Then, the CA uses this account to deploy the designed smart contract (i.e., Smart Contract on DCAP) as a transaction (via Execute) into the blockchain. This smart contract will be recorded into the blockchain after it passes the validation process. Finally, the CA will obtain the smart contract address (denoted as $SCID$).
- *Register:* This phase involves registering two types of entity (i.e., User and Manager). For the registration of a user (e.g., Alice), it first retrieves the system parameters from the blockchain. Then, it generates its long-term account ($pk_a = Q_a, sk_a = d_a$) via invoking Enroll.
  Next, Alice requests the certificate from the CA. To respond to this request, the CA first generates the certificate of pk_a (i.e., computing $cert_a = \mathsf{Sign}(sk_{ca}, pk_a)$, where Sign is the signing algorithm of ECDSA). Then, it embeds $cert_a$ into a transaction that will be broadcasted and chained into the blockchain by the Managers. Then, the CA will obtain the transaction identity $txID_a$, which could be used to retrieve $cert_a$. Finally, the CA invokes the updateTx algorithm to update pk_a and $txID_a$ into the smart contract $SCID$.
  The register of a manager is the same as that of a user. However, the managers co-own the same long-term account in our proposal, and this account is only used for the tracing but not issuing of transactions or consensus. Hence, the register procedure is executed only once and the obtained long-term account ($pk_m = Q_m, sk_m = d_m$) is shared among the managers. Note that we propose managers use group key agreement protocols (e.g. [35], [36]) to generate this account.
- *Transaction Issuance:* Assuming that Alice wishes to transfer v coins to Bob; thus, Bob needs to provide its anonymous address to Alice. Hence, Bob first invokes Update to generate an anonymous account ($ask_b = (r_b, d_b), apk_b = (Q'_b, Q''_b)$). Then, Bob sends apk_b to Alice such that Alice can invoke Execute to produce a pending transaction $tx = (x, \pi)$. Finally, Alice uses its private key sk_a to sign this transaction (i.e., computing $s = \mathsf{Sign}(sk_a, tx)$) and then broadcasts (tx, s) to the managers via TLS.
- *Chain Transaction:* This phase is executed by the managers to chain the pending transactions into the blockchain.
  1) First, they need to verify the validation of the received (tx, s). For example, to verify Alice's (tx_a, s_a), the managers first parse tx_a as $(x_a = (v, Q'_a, Q''_a, Q'_b, Q''_b, Q_m, P), \pi_a)$. Then, they use the private key sk_m to trace the long-term address of $apk_a = (Q'_a, Q''_a)$. That is, they invoke Trace to obtain pk_a. Next, isLegal is invoked to check if pk_a has been blacklisted. If yes, then this pending transaction tx will be discarded; otherwise, the managers continue executing the following step.
     They invoke getTx in $SCID$ to obtain $txID$ that corresponds to the certificate of pk_a. Then, they use $txID$ to obtain the certificate $cert_a$ of Alice's pk_a from this transaction data. Then, they determine whether $\mathsf{Verify}(pk_{ca}, pk_a, cert_a) = 1$ holds (here, Verify is the verification algorithm of ECDSA). If the equation does not hold, then

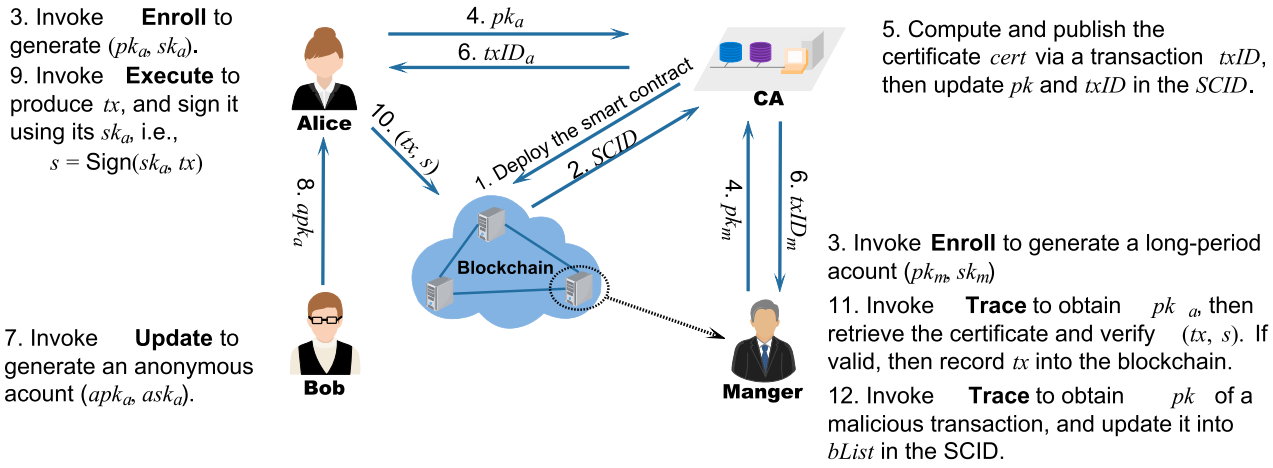


Fig. 3. Architecture of proposed DCAP system.

(tx, s) is discarded; otherwise, pk_a is used to verify if Verify(pk_a, tx, s) $= 1$ holds. If this equation holds, then tx is determined to be a valid pending transaction.

2) Then, the managers utilize the PBFT consensus mechanism to chain the valid pending transactions. That is, a manager is chosen as the current leader to collect these transactions for a pending block. Then, a consensus is reached on this block, if and only if, at least two-third of total managers approve the pending block. Note that, double-spending of a coin can be trivially prevented according to this consensus procedure, for the subsequent double-spending transaction will be discarded by permissioned nodes. Finally, the leader chains this block into the blockchain.

- *Permission Update:* In this phase, the managers can trace the user's real identity (i.e., long-term address) of a suspicious transaction and revoke the authority of these identities. They first invoke Trace using their private key sk_m to obtain the long-term address pk of an anonymous address apk in a suspicious transaction tx. Then, they blacklist this pk via invoking updateBList in the contract $SCID$. Correspondingly, the CA executes the deleteTx algorithm to delete the mapping between the long-term address pk and its certificate. Once the mapping is deleted, the index of a certificate will no longer exist, meaning that this certificate has been revoked or invalided.

VI. Security Analysis

The proposed DCAP system can satisfy all the security requirements described in Section III-B, according to the proposed CAP scheme and the related blockchain-system components (e.g., PBFT).

1) *Single Registration:* In our design of DCAP, each participant (i.e., users and managers) generates their long-term accounts and requests the corresponding certificate from CA. After that, they can publish transactions without any further registration even when other participants join in or leave. In other words, our proposal provides a one-off registration. Obviously, our proposed DCAP only involves 2 interactions (including this one-off registration) and hence more efficient than that in [14] which requires about 14 ones.
2) *Transaction Anonymity:* The transaction anonymity of our CAP scheme is integrated into the DCAP system, in order to prevent the users' real identities from being disclosed. Transactions chained in the DCAP system refer to a SPK proof, which is generated by the anonymous private key (corresponding to the sender's anonymous address). Verification of these transactions only involves the anonymous addresses of sender and receiver, rather than their long-term address. Apart from these anonymous addresses, no related identity information can be obtained from the transactions. Hence, the proposed DCAP can ensure user privacy.
3) *Traceability*. Using our CAP scheme (which supports tracing an anonymous address to its long-term address) and the certificate (which maps a long-term address to its real identity), the manager in our DCAP system can trace the real identities of participants in any transaction. Upon detection of a suspicious transaction, the manager first discloses the anonymous address to obtain the long address (via Trace using its private key) and then retrieves the corresponding certificate from the smart contract.
4) *Interception and Modification Resilience:* Due to the transaction's non-malleability in our CAP scheme, our constructed DCAP system can resist any malicious intent (e.g., modifying the transferred value, anonymous address or transmitted message in a transaction). That is, any modification of these data will cause the invalidation of the transaction, which will be discarded.
5) *Double-Spending Resilience:* Due to the use of PBFT consensus mechanism, our DCAP system can tolerate a third of all permissioned nodes to be faulty. Hence, one needs to hold more than one third of computing power (or stake) if (s)he intends to double-spend. Also,

in our proposal, the obligation of permissioned nodes is performed by the managers (who are all trusted parties). Hence, there exists no group who can process such proportion of computing power (or stake) to control it.

6) *Birthday Collision Resilience:* Our construction refers to a permissioned paradigm (using the PBFT consensus mechanism to chain a block). The mangers as the permissioned nodes are responsible for recording. Hence, there is no 'fork' situation, which effectively avoids blocks' birthday collisions.
7) *Hijacking Resilience:* All transactions are signed by SPK can also resist hijacking attacks, that is, for any $\mathcal{PPT}$ adversary cannot tamper with the context of transactions without invalidating the signatures.
8) *Network Analysis Resilience:* The DCAP system forces users to use a new generated anonymous address in issuing each transaction. This increases address unlinkability and untraceability of transactions without the tracing private key. However, the transferred value in our DCAP is in plaintext, which may be used in the network analysis for tracing the real identity corresponding to anonymous addresses. Hence, we will consider integrating some existing value-hiding technologies such as [26], [28] to further enhance user privacy in DCAP.
9) *Resilience to Other Attacks:* The proposed DCAP can also resist the following attacks.
    a) *51% Attacks:* If a group of attackers holds more than half of the computing power, then the DCAP may be controlled by them. Our proposal can efficiently resist this attack, as our construction is built in a permissioned paradigm. That is, the ledger is maintained by some permissioned nodes (i.e., the managers in our DCAP), which could be dynamically authorized and revoked (once some are compromised).
    b) *Impersonation Attacks:* Only registered users (with the public certificate) can issue a valid transaction in DCAP, and only the managers can revoke the permission of users. Any impersonator disguising as users or managers will be identified since the validation of transactions will fail.
    c) *Distributed Denial of Service Attacks:* DCAP inherits Bitcoin's resilience to DDoS attacks, for example by limiting the block size and checking the maximum number of SPK proofs for transaction input. In addition, the permission update using the smart contract can terminate a user's authority when malicious frequent trading is detected. The operation of revoking can be executed in real-time, as required.

VII. Performance Analysis

To the best of our knowledge, there is no efficient DCAP scheme for cryptocurrency that achieves regulation. Hence, we compare the proposed DCAP system with Zerocash [11], which is the most similar to our work. Zerocash uses zero knowledge proofs (i.e. zk-SNARK) and achieves the highest level of privacy protection. Specifically, we analyze the performance of the designed SPK and CAP, comparing with those of zk-SNARK and Zerocash respectively, as well as deploying a Ethereum test chain to verify the logic design of the involved smart contract.

TABLE I
Comparison With zk-SNARK

Algorithm	Item	zk-SNARK [11]	Our SPK
Setup	Time	5 min 11 s	0.002 ms
	Proving key	896 MB	64 B
	Verification key	749 B	256 B
Prove	Time	1 min 59s	32.08 ms
	Proof	288 B	420 B
Verify	Time	5.4 ms	31.49 ms

A. Comparative Summary

To compare the performance (including storage, communication, and computational costs) between zk-SNARK and Zerocash in [11] and ours, we denote the system security parameter $\lambda = 128$, and use the Barreto-Naehrig (BN) [37] over base field F_p-256 to achieve this security level. Hence, the group elements in $\mathbb{G}$ and $\mathbb{Z}_q$ are presented in 64 bytes and 32 bytes, respectively. Moreover, we adopt the SHA-256 as the secure hash function $\mathcal{H}$, which means that the length of a hash value is 32 bytes. Note that the transferred value v mentioned above is at most 2^{32}, which can be represented in 4 bytes. To ensure the accuracy of our comparison, we choose the same parameters and testing environment as that in [11]. That is, we use the miracl library (a popular cryptographic library, version 7.0) for our simulation. The system configuration is Windows 7 (64 bits) with an Inter(R) Core (TM) i7-4770 CPU of 3.40 GHz and 16-GB RAM.

Comparison with zk-SNARK. To evaluate the communication and storage complexity of our SPK, we count the size of related keys (i.e., proving key and verification key) and the proofs. According to the findings presented in Table I, we observe that the size of keys in our SPK (i.e., 64 B and 256 B respectively) is significantly smaller than that of zk-SNARK (especially the size of proving key). However, the proof size of our SPK is 420 B, which is slightly larger than that of zk-SNARK. For time costs, our SPK is more efficient in the Setup and Prove phases, at the expense of verification efficiency. That is, the time costs of both Setup and Prove phases in our SPK are within milliseconds (i.e., about 0.002 ms and 32.08 ms, respectively), while that in zk-SNARK the time requirements are approximately 5 min 11 s and 1 min 59 s respectively. However, the time cost of Verify in our SPK (requiring about 31.49 ms) is more than that in zk-SNARK (only about 5.4 ms).

Comparison with Zerocash. We also compare our proposed CAP with Zerocash in terms of time, storage and communication costs. As shown in Table II, the time costs of all the algorithms (including Setup, Enroll, Update, Transact, Verify and Trace) in our CAP are within milliseconds (i.e. 0.002 ms, 6.13 ms, 18.59 ms, 32.30 ms, 32.43 ms, and 6.34 ms respectively). We have significantly improved

TABLE II
COMPARISON WITH ZEROCASH

Algorithm	Item	Zerocash [11]	Our CAP
Setup	Time	5 min 11 s	0.002 ms
	PP	896 MB	64 B
Enroll	Time	326.0 ms	6.13 ms
	pk	343 B	64 B
	sk	319 B	32 B
Update	Time	\	18.59 ms
	apk	\	128 B
	ask	\	64 B
Transact	Time	t_{Mint} = 23 us	32.30 ms
		t_{Pour} = 2 min 2.01 s	
	tx	tx_{Mint} = 72 B	548 B
		tx_{Pour} = 996 B	
Verify	Mint	8.3 us	32.43 ms
	Pour	5.7 ms	
Trace	Time	\	6.34 ms

the efficiency in the Setup, Enroll, Transact algorithms (for the time costs in Zerocash are about 5 min 11s, 326 ms, 2 min 2.01 s respectively). Only the time cost of the Verify algorithm in Zerocash (about 5.7 ms) is lower that in our CAP (requiring about 32.43 ms). Note that both the Mint and Pour algorithms in Zerocash belong to executing a transaction, and hence we consider them together into the Transact algorithm. Correspondingly, the Verify algorithm consists of two verifications for the Mint and Pour algorithms. In addition, all the sizes of related variables (e.g., public parameters, key-pairs and transactions) in our CAP enjoy significant efficiency, especially the public parameter size is reduced from 896 MB (in Zerocash) to 64 B (in our CAP). Although an additional Update algorithm is needed in our CAP (with time cost implication of 18.59 ms, and storage or communication costs of 128 B for apk and 92 B for ask), it is a small price to pay for achieving both anonymity and traceability.

B. Prototype Implementation

In order to show the feasibility of the constructed DCAP, we implement the logic procedure on Ethereum which is an open-source blockchain system supporting *Solidity*.[9] Note that our simulation mainly considers the functionality of the designed smart contract, which can be tested in Ethereum (instead of permissioned ones) for convenience. Concretely, we build a private Ethereum chain on a laptop (equipped with a Ubuntu 16.04 system, Intel (R) Core (TM) i7-6700 CPU of 3.40 GHZ and 3-GB RAM). Here, the chosen private chain can avoid transaction fees, but we are still able to obtain the same correct results as that of a public one. Then, we deploy the designed smart contract[10] into this private chain such that the involved functions can be evaluated by using *Web3j*.[11] Next, we test the functionality of the updateTx, getTx, deleteTx, updateBList, isLegal algorithms via 1000 times of invocations to measure the approximate time cost. The test results are shown in Table III, which are consistent to the times required in the generation of a block in Ethereum (within the seconds level, according to the preset difficulty in a private chain). Observe that the getTx and isLeagal algorithms are a type of eth_calls, which only results in network latency without incurring the time generating a block.

[9]Solidity is a special Java script-like language supporting the design of smart contracts – see https://solidity.readthedocs.io/en/v0.5.4/

[10]https://github.com/colyn91/-Smart-Contract-on-DCAP

[11]*Web3j* is a safe Java and Android library for working with Smart Contracts – see https://web3j.readthedocs.io/

TABLE III
TIME COST (IN RM S) OF ALGORITHMS IN THE SMART CONTRACT

Algorithm	updateTx	getTx	deleteTx	updateBList	isLeagal
Average time	9.832	1.891	9.436	7.448	1.926
Max time	12.120	4.938	14.869	10.228	4.004
Min time	7.720	1.065	6.498	3.679	0.926

VIII. CONCLUSION

In this paper, we achieved both anonymity and regulation properties in our decentralized conditional anonymous payment (DCAP) system. Before constructing our DCAP, we defined a conditional anonymous payment (CAP) scheme with the formal semantic and security models. Also, we provided a concrete design of CAP based on our proposed signature of knowledge scheme, and proved it secure in the defined security model. Building on the proposed CAP, we constructed our DCAP and demonstrated how it can satisfy the related security requirements. Then, we evaluated the performance of our prototype by comparing it with that of Zerocash under the same parameters and testing environment. Findings suggested that our proposal is practical for real-world deployment.

A follow-on work is to collaborate with a real-world organization to customize our proposal and implement it in a real-world setting. This will also allow us to identify additional features or properties that need to be included in future version.

APPENDIX
PROOF OF OUR SPK SCHEME

Proof: We will now prove the constructed SPK scheme as follows.

(Perfect) Completeness: Completeness is trivial to demonstrate, and we will omit the details here.

Soundness: Soundness also follows from the special soundness of the Σ-protocol. Let us assume that a $\mathcal{PPT}$ prover P^* produces a valid argument π_2 for an invalid statement, where $\pi = (R_2, c, z_1, z_2)$.

Then, we can construct such an extractor EXTRACT that rewinds P^* to the oracle query $\mathcal{H}(str)$ (where $str = m||Q'||Q''||T||P||(z_1P + cQ')||R_2||(z_2P + cQ'' - R_2 + z_1T)$) for obtaining c (after seeing the argument). It then reorganizes the random oracle such that $\widetilde{c} = H(str)$ with $\widetilde{c} \neq c$ and continues executing P^* under the reorganized random oracle. Another valid argument will be produced under the expected polynomial time, that is, $\widetilde{\pi} = (R_2, \widetilde{c}, \widetilde{z}_1, \widetilde{z}_2)$. For $z_1 = ab - cxy$, $z_2 = b - cy$ and $z_1 = ab - \widetilde{c}xy$, $\widetilde{z}_2 = b - \widetilde{c}y$, the witness can be extracted by computing $y = (z_2 - \widetilde{z}_2)(\widetilde{c} - c)^{-1}$, $x = (z_1 - \widetilde{z}_1)(\widetilde{c} - c)^{-1} \cdot y^{-1}$. Note that both $\widetilde{c} - c$ and y is prime to q based on the assumption that $\mathcal{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ and hence

Algorithm 2 Simulator $\mathcal{S}$

Input: a security parameter 1^{λ}
Output: the proof π
1: $\mathcal{O}_H \leftarrow \mathsf{Sim}(1^{\lambda})$
2: Given an instance (c^*, m, Q', Q'', Q, P) to be proved.
3: Randomly choose $c^* \in \mathbb{Z}_q^*$, $z_1^* \in \mathbb{Z}_q^*$, $z_2^* \in \mathbb{Z}_q^*$.
4: Compute $R_1^* = z_1^* P + c^* Q'$, $R_2^* = z_1^* T + -c^*(Q'' - Q)$, $R_3^* = z_2^* P + cQ'' - R_2^* + z_1^* T$.
5: Set $\mathcal{O}_H(m||Q'||Q''||T||P||R_1^*||R_2^*||R_3^*) = c^*$.
6: **return** $\pi_2 = (R_2^*, c^*, z_1^*, z_2^*)$

the $(\widetilde{c} - c)^{-1}$ and y^{-1} can be computed by $\mathsf{ExtendGCD}$[12] by invoking the extended Euclidean algorithm $\mathsf{ExtendGCD}$, for $gcd(\widetilde{c} - c, q) = 1$ and $gcd(y, q) = 1$.

Special Honest-Verifier Zero-Knowledge: To prove this property, we first construct a simulator $\mathcal{S}$ (see Algorithm 2) that simulates all interactions with any honest verifier V^*, with the exception that $\mathcal{S}$ executes the hash function $\mathcal{H}$ as a random oracle $\mathcal{O}_{\mathcal{H}}$. Then, this simulator produces the argument $\pi = (R_2^*, c^*, z_1^*, z_2^*)$.

One can find that this proof will be accepted by V^* for the simulated hash function as a random oracle. Moreover, the randomnesses including c^*, z_1^*, z_2^* in $\mathcal{S}$ have the same distribution as that in the real environment. Then, R_1^*, R_2^*, R_3^* are uniquely determined by c^*, z_1^*, z_2^* according to the computations $R_1^* = z_1^* P + c^* Q'$, $R_2^* = z_1^* T + -c^*(Q'' - Q)$, $R_3^* = z_2^* P + cQ'' - R_2^* + z_1^* T$. This represents that the distributions of R_1^*, R_2^*, R_3^* are identical to those in the real environment. Thus, the argument $(R_2^*, c^*, z_1^*, z_2^*)$ produced as an output by $\mathcal{S}$ has the same distribution as an honestly produced argument.

REFERENCES

[1] E. Portmann, "Rezension, blockchain: Blueprint for a new economy,"' *HMD Praxis der Wirtschaftsinformatik*, vol. 55, no. 6, pp. 1362–1364, 2018.

[2] C. Lin, D. He, X. Huang, K.-K.-R. Choo, and A. V. Vasilakos, "BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *J. Netw. Comput. Appl.*, vol. 116, pp. 42–52, Aug. 2018.

[3] D. Chaum, "Blind signatures for untraceable payments," in *Proc. CRYPTO*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Santa Barbara, CA, USA: Plenum Press, Aug. 1982, pp. 199–203.

[4] T. Sander and A. Ta-Shma, "Auditable, anonymous electronic cash extended abstract," in *Proc. Adv. Cryptol.-CRYPTO*, in Lecture Notes in Computer Science, vol. 1666, M. J. Wiener, Ed. Santa Barbara, CA, USA: Springer, Aug. 1999, pp. 555–572.

[5] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Consulted*, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[6] G. Wood, "Ethereum: A secure decentralized generalized transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.

[7] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Proc. Financial Cryptogr. Data Secur.-18th Int. Conf. (FC)* in Lecture Notes in Computer Science, vol. 8437, N. Christin and R. Safavi-Naini, Eds. Christ Church, Barbados: Springer Mar. 2014, pp. 486–504.

[8] D. Ron and A. Shamir, "Quantitative analysis of the full bitcoin transaction graph," in *Proc. Financial Cryptogr. Data Secur.-17th Int. Conf. (FC)*, in Lecture Notes in Computer Science, vol. 7859, A. Sadeghi, Ed. Okinawa, Japan: Springer, Apr. 2013, pp. 6–24.

[9] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Proc. IEEE 3rd Int. Conf. Privacy, Secur., Risk Trust IEEE 3rd Int. Conf. Social Comput.*, Boston, MA, USA, Oct. 2011, pp. 1318–1326.

[10] S. Noether, A. Mackenzie, and T. M. Research Lab, "Ring confidential transactions," *ledger*, vol. 1, pp. 1–18, Dec. 2016.

[11] E. B. Sasson *et al.*, "Zerocash: Decentralized anonymous payments from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2014, pp. 459–474.

[12] N. van Saberhagen. (2013). *Cryptonote v 2.0*. [Online]. Available: https://static.coinpaprika.com/storage/cdn/whitepapers/1611.pdf.

[13] B. Qin, L. Chen, Q. Wu, Y. Zhang, L. Zhong, and H. Zheng, "Bitcoin and digital flat currency," *J. Cryptolog. Res.*, vol. 4, no. 2, pp. 176–186, 2017.

[14] Y. Wu, H. Fan, X. Wang, and G. Zou, "A regulated digital currency," *Sci. China Inf. Sci.*, vol. 62, Jan. 2019, Art. no. 32109.

[15] P. Koshy, D. Koshy, and P. D. McDaniel, "An analysis of anonymity in bitcoin using P2P network traffic," in *Proc. Financial Cryptogr. Data Secur.-18th Int. Conf. (FC)*, in Lecture Notes in Computer Science, vol. 8437, N. Christin and R. Safavi-Naini, Eds. Christ Church, Barbados: Springer, Mar. 2014, pp. 469–485.

[16] D. Vandervort, "Challenges and opportunities associated with a bitcoin-based transaction rating system," in *Proc. Financial Cryptogr. Data Secur.-FC Workshops*, in Lecture Notes in Computer Science, vol. 8438, R. Böhme, M. Brenner, T. Moore, and M. Smith, Eds. Christ Church, Barbados: Springer, Mar. 2014, pp. 33–42.

[17] E. Duffiel and D. Diaz. (2015). *Dash: A Privacycentric Cryptocurrency*. [Online]. Available: https://www.dash.org/

[18] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.*, vol. 8976, M. Brenner, N. Christin, B. Johnson, and K. Rohloff, Eds. San Juan, Puerto Rico: Springer, Jan. 2015, pp. 112–126.

[19] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Proc. Eur. Symp. Res. Comput. Secur.*, in Lecture Notes in Computer Science, vol. 8713, M. Kutylowski and J. Vaidya, Eds. Wroclaw, Poland: Springer, Sep. 2014, pp. 345–364.

[20] G. Bissias, A. P. Ozisik, B. N. Levine, and M. Liberatore, "Sybil-resistant mixing for bitcoin," in *Proc. 13th Workshop Privacy Electron. Soc.-WPES*, G. Ahn and A. Datta, Eds. Scottsdale, AZ, USA, Nov. 2014, pp. 149–158.

[21] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "Coinparty: Secure multi-party mixing of bitcoins," in *Proc. 5th ACM Conf. Data Appl. Secur. Privacy (CODASPY)*, J. Park and A. C. Squicciarini, Eds. San Antonio, TX, USA, Ma. 2015, pp. 75–86.

[22] S. N. Foley, D. Gollmann, and E. Snekkenes, Eds., *Computer Security* (Lecture Notes in Computer Science), vol. 10493. Oslo, Norway: Springer, Sep. 2017.

[23] S. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero," in *Proc. 22nd Eur. Symp. Res. Comput. Secur. (ESORICS)*, in Lecture Notes in Computer Science, vol. 10493, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Oslo, Norway: Springer, Sep. 2017, pp. 456–474.

[24] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed E-cash from bitcoin," in *Proc. IEEE Symp. Secur. Privacy*, Berkeley, CA, USA, May 2013, pp. 397–411.

[25] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," in *Proc. 25th Int. Conf. Theory Appl. Cryptol. Inf. Secur. (ASIACRYPT)*, Kobe, Japan, Dec. 2019, pp. 649–678.

[26] S. Ma, Y. Deng, D. He, J. Zhang, and X. Xie, "An efficient NIZK scheme for privacy-preserving transactions over account-model blockchain," *IEEE Trans. Dependable Secure Comput.*, Jan. 2020, doi: 10.1109/TDSC.2020.2969418.

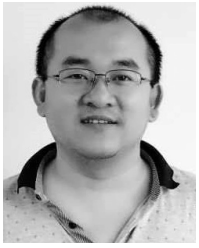
[27] N. Narula, W. Vasquez, and M. Virza, "Zkledger: Privacy-preserving auditing for distributed ledgers," in *Proc. 15th USENIX Symp. Networked Syst. Des. Implement. (NSDI)*, S. Banerjee and S. Seshan, Eds. Renton, WA, USA: USENIX Association, Apr. 2018, pp. 65–80.

[28] G. Fuchsbauer, M. Orrù, and Y. Seurin, "Aggregate cash systems: A cryptographic investigation of mimblewimble," in *Proc. 38th Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, in Lecture Notes in Computer Science, vol. 11476, Y. Ishai and V. Rijmen, Eds. Darmstadt, Germany: Springer, May 2019, pp. 657–689.

[12] Obviously, it is easy to compute $(-a, b)$ such that $(-a) \cdot (\widetilde{c} - c) + b \cdot q = 1$ (or $(-a) \cdot y + b \cdot q = 1$).

[29] V. K. Wei, "More compact E-cash with efficient coin tracing," *IACR Cryptol. ePrint Arch.*, vol. 2005, p. 411, May 2005.
[30] M. Chase and A. Lysyanskaya, "On signatures of knowledge," in *Proc. 26th Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 4117, C. Dwork, Ed. Santa Barbara, CA, USA: Springer, Aug. 2006, pp. 78–96.
[31] H. Eun, H. Lee, and H. Oh, "Conditional privacy preserving security protocol for NFC applications," *IEEE Trans. Consum. Electron.*, vol. 59, no. 1, pp. 153–160, Feb. 2013.
[32] A. D. Santis, G. D. Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai, "Robust non-interactive zero knowledge," in *Proc. 21st Annu. Int. Cryptol. Conf.*, in Lecture Notes in Computer Science, vol. 2139, J. Kilian, Ed. Santa Barbara, CA, USA: Springer, Aug. 2001, pp. 566–598.
[33] S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi, "On the non-malleability of the fiat-Shamir transform," in *Proc. 13th Int. Conf. Cryptol.*, in Lecture Notes in Computer Science, vol. 7668, S. D. Galbraith and M. Nandi, Eds. Kolkata, India: Springer, Dec. 2012, pp. 60–79.
[34] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd USENIX Symp. Operating Syst. Design Implement. (OSDI)*, M. I. Seltzer and P. J. Leach, Eds. New Orleans, LA, USA: USENIX Association, Feb. 1999, pp. 173–186.
[35] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, "Block design-based key agreement for group data sharing in cloud computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 16, no. 6, pp. 996–1010, Nov. 2019.
[36] X. Li, Y. Wang, P. Vijayakumar, D. He, N. Kumar, and J. Ma, "Blockchain based mutual-healing group key distribution scheme in unmanned aerial vehicles ad-hoc network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 11, pp. 11309–11322, Nov. 2019.
[37] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, in Lecture Notes in Computer Science, vol. 3897, B. Preneel and S. E. Tavares, Eds. Kingston, ON, Canada: Springer, Aug. 2005, pp. 319–331.

Chao Lin received the bachelor's and master's degrees from the School of Mathematics and Computer Science, Fujian Normal University, in 2013 and 2017, respectively. He is currently pursuing the Ph.D. degree with the School of Cyber Science and Engineering, Wuhan University. His research interests mainly include authentication of graph data and blockchain security.



Debiao He received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, in 2009. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. His main research interests include cryptography and information security, in particular, cryptographic protocols.

Xinyi Huang received the Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia. He is currently a Professor with the School of Mathematics and Computer Science, Fujian Normal University, China, where he is also the Co-Director of the Fujian Provincial Key Laboratory of Network Security and Cryptology. He has authored over 100 research articles in refereed international conferences and journals. His research interests include applied cryptography and network security. His work has been cited more than 2900 times at Google Scholar (H-index: 27). He is also an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.

Muhammad Khurram Khan (Senior Member, IEEE) is currently working as a Professor of cybersecurity with the Center of Excellence in Information Assurance (CoEIA), King Saud University, Saudi Arabia. He has played a leading role in developing B.S. degree Program in cybersecurity and the Higher Diploma degree in cybersecurity with King Saud University. He has published more than 350 research articles in the journals and conferences of international repute. In addition, he is an Inventor of 10 U.S./PCT patents. He is also the Founder and the CEO of the Global Foundation for Cyber Studies and Research. He has secured several national and international competitive research grants in the domain of cybersecurity. He has edited seven books/proceedings published by Springer-Verlag and the IEEE. His research areas of interest are cybersecurity, digital authentication, IoT security, cyber policy, and technological innovation management. He is a fellow of the IET, U.K., BCS, U.K., and FTRA, South Korea, a Senior Member of the IACSIT, Singapore, and a member of the IEEE Consumer Electronics Society, the IEEE Communications Society, the IEEE Technical Committee on Security & Privacy, the IEEE IoT Community, the IEEE Smart Cities Community, and the IEEE Cybersecurity Community. He is also the Vice Chair of the IEEE Communications Society Saudi Chapter. He is also the Editor-in-Chief of a well-reputed *International Journal Telecommunication Systems* published by Springer for over 26 years, with its recent impact factor of 1.707 (JCR 2019). Furthermore, he is on the Editorial Board of several international journals, including the IEEE COMMUNICATIONS SURVEYS & TUTORIALS, the *IEEE Communications Magazine*, the IEEE INTERNET OF THINGS JOURNAL, and the IEEE TRANSACTIONS ON CONSUMER ELECTRONICS, and so on. He is a Distinguished Lecturer of the IEEE (CESoC).

Kim-Kwang Raymond Choo (Senior Member, IEEE) received the Ph.D. degree in information security from the Queensland University of Technology, Australia, in 2006. He currently holds the Cloud Technology Endowed Professorship at The University of Texas at San Antonio. In 2016, he was named the Cybersecurity Educator of the Year–APAC (Cybersecurity Excellence Awards are produced in cooperation with the Information Security Community on LinkedIn). He is also a fellow of the Australian Computer Society. In 2015, he and his team won the Digital Forensics Research Challenge organized by Germany's University of Erlangen-Nuremberg. He was a recipient of the 2019 IEEE Technical Committee on Scalable Computing (TCSC) Award for Excellence in Scalable Computing (Middle Career Researcher) and the 2018 UTSA College of Business Col. Jean Piccione and Lt. Col. Philip Piccione Endowed Research Award for Tenured Faculty. He is also the Co-Chair of IEEE Multimedia Communications Technical Committee. He is the Outstanding Associate Editor of 2018 for IEEE ACCESS, the British Computer Society's 2019 Wilkes Award Runner-up, the 2019 *EURASIP Journal on Wireless Communications and Networking* (JWCN) Best Paper Award, the Korea Information Processing Society's *Journal of Information Processing Systems* (JIPS) Survey Paper Award (Gold) 2019, the IEEE Blockchain 2019 Outstanding Paper Award, the IEEE TrustCom 2018 Best Paper Award, the ESORICS 2015 Best Research Paper Award, the 2014 Highly Commended Award by the Australia New Zealand Policing Advisory Agency, the Fulbright Scholarship in 2009, the 2008 Australia Day Achievement Medallion, and the British Computer Society's Wilkes Award in 2008.