This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

Traceable Monero: Anonymous Cryptocurrency with Enhanced Accountability

Yannan Li, *Student Member, IEEE,* Guomin Yang, *Senior Member, IEEE,*
Willy Susilo, *Senior Member, IEEE,* Yong Yu, *Member, IEEE,* Man Ho Au, *Member, IEEE* and Dongxi Liu

Abstract—Monero provides a high level of anonymity for both users and their transactions. However, many criminal activities might be committed with the protection of anonymity in cryptocurrency transactions. Thus, user accountability (or traceability) is also important in Monero transactions, which is unfortunately lacking in the current literature. In this paper, we fill this gap by introducing a new cryptocurrency named *Traceable Monero* to balance the user anonymity and accountability. Our framework relies on a tracing authority, but is optimistic, in that it is only involved when investigations in certain transactions are required. We formalize the system model and security model of Traceable Monero. We present a detailed construction of Traceable Monero by overlaying Monero with two types of tracing mechanisms, tracing the one-time addresses with money flows and tracing the long-term addresses. We prove the security of Traceable Monero and implement a prototype of the system, which demonstrates that Traceable Monero incurs merely a very small overhead in generating and verifying a transaction compared to Monero transactions.

Index Terms—Cryptocurrency, Monero, Blockchain, Anonymity, Accountability.

1 Introduction

Cryptocurrency is a digital currency whose security is mainly based on public-key cryptography. Unlike traditional centralized system [1–4] that suffers from issues of scalability and delay, cryptocurrency systems are decentralized, which enable two parties to conduct transactions directly. The trading and verification of cryptocurrencies are achieved through a mechanism known as a blockchain. Blockchain [5][6] is an append-only public ledger by all the participants collectively in the system in a verifiable and permanent way.

Protecting users' privacy is one of the most enticing features of cryptocurrencies. If transactions are not conducted anonymously, a malicious merchant may sell customers' transaction information to third parties for financial benefit. The loss of privacy in cryptocurrency transactions may eventually lead to spam and other harassment in a user's daily life.

Monero (XMR) [1] is an open-source decentralized cryptocurrency that mainly focuses on privacy and anonymity. It was launched in 2014 and now has been one of the largest cryptocurrencies with a market capitalization of 5.7 billion US dollars (Mar. 2018). Monero protects payee's identity based on CryptoNote protocol [7] and takes advantage of the linkable ring signature [8], which is known as Ring Confidential Transactions (RingCT) protocol [9].

Anonymity is a desirable requirement to preserve users' privacy in cryptocurrency transactions. Unfortunately, anonymity makes investigating illegal transactions more difficult. More specifically, anonymity in cryptocurrencies provides a cover for various illegal activities, such as money laundering, contraband trading and extortion, as the adversaries are hard to be identified and punished. The abuse of anonymous cryptocurrencies for illegal purposes is on the rise in recent years. In 2017, the worldwide cyberattack WannaCry[2][3] hacked more than 300,000 computers across 150 countries by encrypting files and asking for money to ransom them. Victims were required to pay $300 - $600 in Bitcoin to three hardcoded accounts. It is estimated that the financial loss caused by WannaCry incident is about 4 billion dollars and the perpetrators are still unknown. Three months later, the Bitcoins paid by victims have been exchanged for Monero, which utilizes the one-time address and is incredibly hard to trace [4]. Thus, traceability and the surveillance in anonymous cryptocurrency is of great importance. Our motivation is to optimally trace an extortionist in anonymous Monero to deter criminals or simplify investigations when a blackmail event happens. As shown in [10] (Paragraph 2), when the data are at stake, many people are willing to put aside the privacy and reveal some necessary information. Just as the example above, the victims in WannaCry would have enough incentive to adopt the system of anonymous cryptocurrency with traceability.

Our Contributions. We introduce *Traceable Monero*, a solution to balance the anonymity and traceability in cryp-

- *Yannan Li, Guomin Yang and Willy Susilo are with School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia.*
  Email: yl738@uowmail.edu.au; {wsusilo,gyang}@uow.edu.au.
- *Yong Yu is with School of Computer Science, Shaanxi Normal University, Xi'an, 710062, China.*
  E-mail: yuyong@snnu.edu.cn.
- *Man Ho Au is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong.*
  E-mail: csallen@comp.polyu.edu.hk.
- *Dongxi Liu is with Data61, CSIRO, Australia.*
  E-mail: Dongxi.Liu@data61.csiro.au.

Manuscript received March 24, 2018; revised October 20, 2018.

1. http://monero.org/.

2. http://malware.wikia.com/wiki/WannaCry.

3. https://arstechnica.com/gadgets/2017/08/researchers-say-wannacry-operator-moved-bitcoins-to-untraceable-monero/

4. https://cointelegraph.com/news/bitcoin-exchange-shapeshift-helps-police-as-wannacry-attacker-converts-to-monero

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

tocurrency Monero. In Traceable Monero, normal transactions can still be conducted anonymously as in the Monero system except that there exists a tracing authority who is able to revoke a payer's anonymity due to his/her misbehavior. It is worth noting that the tracing authority in Traceable Monero is passive and optimistic, meaning that it will not interfere with any transaction and is involved only when investigation is required. Specifically, our contributions are as follows.

1) We introduce Traceable Monero, a new cryptocurrency system which can achieve conditional anonymity and traceability in Monero simultaneously. We formalize the system model and the security model of Traceable Monero, including *balance*, *anonymity* and *traceability*. We instantiate Traceable Monero, in which we propose two tracing mechanisms atop an improved Monero, and prove that the proposed system achieves all the security requirements in the formalized security model.
2) We implement the proposed scheme. The performance analysis and experimental results demonstrate that the proposed system incurs only a small overhead compared with the underlying Monero system.

1.1 Anonymity in Cryptocurrencies Revisited

Bitcoin [5] provides pseudonymity instead of true anonymity [11] of a user, but the pseudonym mechanism of Bitcoin is not sufficiently strong to protect users' privacy in some real-world applications. Monero is a secure, private and untraceable cryptocurrency system. Besides, there are several other cryptocurrencies proposed to enhance the level of user privacy. Dash[5] was released in 2014 and is the first privacy-focused cryptocurrency designed on top of an improved version of Bitcoin. PIVX[6]is the fourth largest privacy-oriented cryptocurrecy in the sense of market cap. The ecosystem of PIVX is similar to Dash, and it uses PoS, which requires 10,000 tokens to be the Master Nodes. Lately, PIVX proposed zPIV, which combines PIVX and zerocoin [12] to receive a better privacy. Verge(XVG) [7] was proposed in 2014 to improve Bitcoin privacy. It uses TOR (The onion router) and I2P to hide the real IP address. The wraith protocol makes Verge even more anonymous. Zerocash (Zcash) [13] was proposed in 2014, which leverages a nested commitment to protect the payer identity as well as the transaction amount, then a zero-knowledge succinct non-interactive argument of knowledge (zk-SNARKs) [14] proof is generated to let miners validate the commitment without knowing the serial number, addresses or the transaction amount. A formal security proof is provided for Zcash but improving its efficiency is still a challenging problem. In 2017, Sun et al. [15] proposed a new efficient RingCT protocol (RingCT 2.0), in which accumulators with one-way domain [16] were employed to significantly save the storage space. More privacy-enhancing techniques and altcoins to Bitcoin can be found in [17](section V-B)[18], such as Mix and Coinjoin.

5. https://www.dash.org/
6. https://pivx.org/
7. https://vergecurrency.com/

1.2 Traceability in Cryptocurrencies Revisited

Anonymity may hinder the acceptance and adoption of cryptocurrencies. Moreover, anonymous cryptocurrencies also suffer criticism from governments. To reduce the abuse of cryptocurrencies for illegal activities, many countries have put the regulation of cryptocurrencies on agenda [8][9]. Blockchain surveillance has received much attention. Some companies, such as Chainalysis [10] and Elliptic [11] are designing tracing software for them who are seeking to monitor cryptocurrecies.

In academia, there are already some anonymity revocation methods in traditional e-cash. Some e-cash schemes rely on a trusted party in the withdrawal phase that can bind the customer [19] or in opening an account [20]. Camenisch et al. [21] proposed a scheme with passive anonymity-revoking trustees based on a fair blind signature. Kulger et al. [22] proposed a deanonymization without trusted third parties, which detects illegal transactions in an audit phase.

Lately, a few attempts have been put forward for traceability of cryptocurrencies and the proposed methods mainly fall into the following three categories: **(1) Based on transaction analysis.** Most existing tracing methods for Bitcoin are based on statistical approaches to collect and analyze the transactions. The re-use (use the same account in more than one transactions) and co-use (use more than one accounts in a single transaction) of Bitcoin addresses together with their topologies can be used to match some of the accounts to the same user. Moreover, it is shown by Barcelo [23] that, with some external information [24, 25], it is possible to find real identities. Moreno-Sanchez et al. [26] deanonymized transactions in Ripple network using heuristic clustering to group wallets according to the observations. In 2018, M*ö*ser et al. [27] proposed an empirical analysis for Monero traceability. According to on-chain transaction and mining pool analysis, 62% transaction input can be deduced. However, for the aforementioned tracing methods, plenty of transactions are needed to do the statistical analysis. If a user joined the system for few times, he/she could be identified only with small probability. **(2) Based on a central party.** Danezis et al. [28] put forward a cryptocurrency framework called RSCoin, in which the central banks are required to command over the financial policies. In CCS 2017, Cecchetti et al. [29] proposed *Solidus*. The protocol contains several banks to control all the accounts and complete the transactions on behalf of the users. It supports strong confidentiality for bank-intermediated ledgers in which the payer and payee cannot be traced. **(3) Based on cryptographic tools.** The existing tracing methods in this category are very few. For the regulatory purpose, Garman et al. [30] introduced privacy-preserving policy-enforcement mechanisms, in which selective user and coin can be traced in a cryptocurrency. Note that the tracing mechanisms proposed in this paper falls into this category with Monero as an underpinning.

8. https://www.acic.gov.au/sites/g/files/net1491/f/2016/06/acc_ar_2014-15.pdf?v=1467012395.
9. https://themerkle.com/south-korean-regulators-aim-to-increase-bitcoin-trading-supervision/
10. https://www.chainalysis.com/
11. https://www.elliptic.co/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

Organization. The rest of the paper is organized as follows. The system model and security model of Traceable Monero are presented in Sec. 2 and Sec. 3, respectively. Some preliminaries are provided in Sec. 4. We present the detailed construction of the proposed Traceable Monero system in Sec. 5. We analyze the properties of the proposed system in Sec. 6. We then show the elaborate implementation in Sec. 7. Finally, we conclude the paper and provide future directions in Sec. 8.

2 System Model of Traceable Monero

In this part, we introduce the system model and system components of Traceable Monero system.

2.1 System Model

As shown in Fig. 1, four entities namely, users, a P2P network, a tracing authority and a blockchain are involved in Traceable Monero. Users include payers and payees who have long-term addresses. Users generate transactions with a one-time private key and distribute the transactions in the P2P network. The P2P network consists of a number of peers, who are supposed to have strong computation power and resources. The peers can be honest or malicious, but the 51% attack is assumed unworkable in the P2P network. The peers can behave as miners to generate a block for some valid transactions. Miners are expected to achieve the *Consensus* in the system. Peers can also act as users to generate transactions when a peer needs to transfer some digital currencies to others. The tracing authority is responsible for tracing a payer in dubious transactions with his private key. The blockchain is a continuously growing database, which consists of blocks and records the valid transactions in the P2P network.

2.2 System Components

Specifically, Traceable Monero system is composed of the following algorithms.

Setup(1^λ)$\rightarrow$(pp). On input a security parameter $\lambda \in N$, it outputs a public parameter pp of the system.

KeyGen(pp)$\rightarrow$(pk, sk). On input the public parameter pp, it outputs a public-private key pair (pk, sk).

Mint(pk, a)$\rightarrow$ (ck, cn). On input an amount a and the public key pk, this algorithm outputs a coin cn with amount a and the corresponding coin key ck. The coin cn and the public key pk consist of an account denoted by $act = (pk, cn)$. And the corresponding account secret key is defined as $ask = (sk, ck)$.

Spend(m, K_s, A_s, A, R, pk_M)$\rightarrow$(tx, π, S, CT). On input a group of spending addresses A_s together with the corresponding group of secret key K_s, a set of groups of input addresses A including A_s, a set of output address R, some transaction descriptions m $\in \{0,1\}^*$ and tracing authority's public key pk_M, this algorithm outputs a transaction tx, a proof π, a set of serial number S and a ciphertext CT.

Verify(tx, π, S, CT)$\rightarrow$(0/1). On input the transaction tx, the proof π, the set of serial numbers S and the cipertext CT, this algorithm checks the transaction is valid or not and outputs 1 or 0.

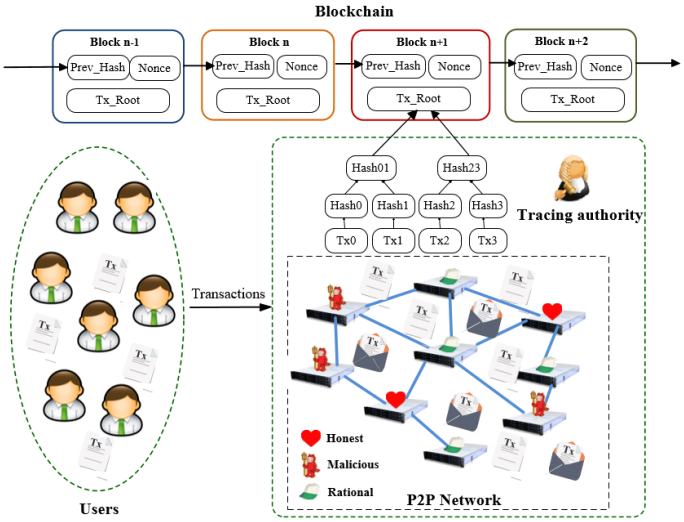


Fig. 1
System model of Traceable Monero

BlockGen($\{tx\}$, blk_{n-1})$\rightarrow$(non, blk_n). On input a set of transactions $\{tx\}$ and the previous block blk_{n-1}, this algorithm outputs a nonce non and a block blk_n.

BlockVer(blk_n)$\rightarrow$(1/0). On input the current block blk_n, this algorithm verifies whether the block is valid or not and outputs 1 or 0.

Trace(sk_M, tx, CT)$\rightarrow$(pk_s, ψ). On input the tracing authority's secret key sk_M, the transaction tx and the ciphertext CT, it outputs the payer's public key pk_s and a tracing proof ψ.

Judge(pk_s, ψ, pk_M, CT)$\rightarrow$(0/1). On input the payer's public key pk_s, the tracing proof ψ, the tracing authority's public key pk_M and the ciphertext CT, this algorithm checks whether the tracing proof is valid and outputs 1 or 0.

3 Properties of Traceable Monero

Regarding the security of Traceable Monero, we assume the involved P2P network in the system is secure and reliable, which can resist the common attacks such as Sybil attack, selfish mining attack etc. In practice, we can employ the P2P network of Monero as the P2P network of Traceable Monero. Thus, we only focus on the security issues related to the transactions. Below we first present the security threats to traceable Monero, and then introduce the formal security definitions.

3.1 Threat Model

As analyzed above, focusing on secure transactions, we assume an adversary may launch the following attacks [15] in traceable Monero.

Double-Spending Attack: Double-spending attack says that the money in an account is spent in more than one transaction.

Over-Spending Attack: Over-spending attack attempts to spend a larger amount of money than that in an account in a transaction.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

Anonymity Attack: Anonymity attack is that one can identify the payer in a transaction without tracing authority's private key.

Forgery Attack: Forgery attack indicates that a malicious payer can spend money in the accounts without his control. That is, to forge a transaction without the corresponding private key of an account.

Linkability Attack: Linkability attack states that two transactions issued by the same payer can be linked without tracing authority's private key. Note that linking two transactions performed by the same payer is easier than identifying the payer in those transactions.

Traceability Attack: Traceability attack is that a payer successfully conducts a transaction but cannot be identified by the tracing authority.

3.2 Security Model

In this subsection, we formalize the security model, whose goal is *Balance*. *Balance* aims to cover over-spending attack, double-spending attack and forgery attack. Formal security definitions of Traceable Monero are defined as follows.

Definition 1 (Balance) [15]. Balance requires that a malicious payer cannot (1) spend the money more than that in his account and (2) spend the money without his control. A traceable Monero is balanced if for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\mathcal{A}\ Wins\colon \begin{matrix} pp\leftarrow \mathsf{Setup}(1^\lambda);(\{act_i'\}_{i=1}^{\mu},\{\mathcal{S}_i\}_{i=1}^{v}) \\ \leftarrow \mathcal{A}^{AddGen,ActGen,Spend,Corrupt}(pp) \end{matrix}\right]\leq negl(\lambda),$$

in which the definitions of the oracles *AddGen*, *ActGen*, *Spend* and *Corrupt* are as follows:

- *AddGen*(i): On input a query number i, pick a randomness τ_i, run algorithm $(sk_i, pk_i) \leftarrow \mathsf{KeyGen}(pp;\tau_i)$ and return an address pk_i.
- *ActGen*(i, a_i): On input an address index i and an amount a_i, run algorithm $(cn_i, ck_i) \leftarrow \mathsf{Mint}(pk_i, a_i)$, then add i and account $act_i = (pk_i, cn_i)$ to initially empty lists $\mathcal{I}$ and $\mathcal{G}$ respectively, and output (act_i, ck_i) for pk_i, where pk_i was generated by *AddGen*.
- *Spend*$(\mathrm{m}, A_s, A, R, pk_M)$: On input a transaction string m, input addresses A containing A_s and output addresses R, run $(tx, \pi, S, CT) \leftarrow \mathsf{Spend}(\mathrm{m}, , K_s, A_s, A, R, pk_M)$ and return (tx, π, S, CT) after adding it to list $\mathcal{T}$. We presume at least one of the addresses in A_s has not been corrupted.
- *Corrupt*(i): On input a query number $i \in \mathcal{I}$, determine the serial number s_i of account act_i with address pk_i using account secret key ask_i, then add s_i and (s_i, a_i) to lists $\mathcal{C}$ and $\mathcal{B}$ respectively, where a_i is the amount of the account with address pk_i, and finally return τ_i.

Finally, $\mathcal{A}$ outputs all her spending with some new accounts $(act_1^{'}, act_2^{'}, \cdots, act_\mu^{'}, \mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_v)$ such that $\mathcal{S}_i = (tx_i, \pi_i, S_i, CT_i)$, where all spends are payed to, w.l.o.g., the challenger with address pk_c, i.e., $tx_i = (\mathrm{m}_i, A_i, A_{\{pk_c\}})$ for all $i \in [v]$. $\mathcal{A}$ wins the game if her outputs satisfy the following conditions:

1. $\mathsf{Verify}(tx_i, \pi_i, S_i, CT_i) = 1$ for all $i \in [v]$.
2. $\mathcal{S}_i \notin \mathcal{T} \wedge S_i \subset \mathcal{S}$ for all $i \in [v]$, and $S_j \cap S_k = \emptyset$ for any different $j, k \in [v]$.
3. Let $S_j = \{s_{i,j}\}$ and $E = \bigcup_{i=1}^{v} \{a_{i,j} : (s_{i,j}, a_{i,j}) \in \mathcal{B} \wedge s_{i,j} \in S_i \cap \mathcal{C}\}$, it holds that $\sum_{a_{i,j}\in E} a_{i,j} < \sum_{i=1}^{v} a_{out,i}$, where $a_{out,i}$ denotes the balance of an output account in $\mathcal{S}_i$.

3.3 Other Desirable Properties

In this subsection, we list several other required properties of the traceable Monero, including *Perfect Correctness*, *Anonymity* and *Traceability*, in which *Anonymity* is to resist linkability attack and anonymity attack and *Traceability* is for against traceability attack. Formal definitions of these properties are defined as follows.

Definition 2 (Perfect Correctness). All transactions generated by Spend can be accepted by Verify and any proof produced by Trace can pass Judge. Formally, a traceable Monero has the property of perfect correctness if for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix} \mathsf{Verify}(tx,\pi,S, \\ CT)=1 \\ \mathsf{Judge}(pk_s,\psi,pk_M, \\ CT)=1 \end{matrix} : \begin{matrix} (pp)\leftarrow\mathsf{Setup}(1^\lambda); \\ (pk,sk)\leftarrow\mathsf{KeyGen}(pp) \\ (cn,ck)\leftarrow\mathsf{Mint}(pk,a) \\ (\mathsf{m},A,R)\leftarrow\mathcal{A}(pp,A_s,K_s); \\ (tx,\pi,S)\leftarrow\mathsf{Spend} \\ (\mathsf{m},K_s,A_s,A,R,pk_M) \\ (pk_s,\psi)\leftarrow\mathsf{Trace}(sk_M,tx,CT) \end{matrix}\right]=1.$$

Definition 3 (Anonymity). A traceable Monero is anonymous if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, it holds that

$$\left|\Pr\left[b'=b\colon \begin{matrix} pp\leftarrow\mathsf{Setup}(1^\lambda);(\mathrm{m},A_{s_0},A_{s_1},A,R)\leftarrow \\ \mathcal{A}_1^{AddGen,ActGen,Spend,Corrupt,Trace}(pp); \\ b\leftarrow\{0,1\},(tx^*,\pi^*,S^*,CT^*)\leftarrow \\ Spend(\mathrm{m},K_{s_b},A_{s_b},A,R,pk_M);b^{'}\leftarrow \\ \mathcal{A}_2^{Spend,Corrupt,Trace}(pp,(tx^*,\pi^*,S^*,CT^*)) \end{matrix}\right]-\frac{1}{2}\right|\leq negl(\lambda)$$

where all oracles are defined as before, $A_{s_i} \in A$ and $A_{s_i} \subset \mathcal{G}$ for $i \in \{0, 1\}$ and *Trace* oracle is defined as follows.

- *Trace*(tx, π)[12]. This is an oracle that on query a transaction tx together with a proof π, it returns a public key pk_s. If the query (tx, π) is the challenging pair, then the oracle returns $\perp$.

Besides, the following conditional also needs to be satisfied:

– For all $i \in \{0, 1\}$, none of the accounts in A_{s_i} have been corrupted.

– No query in the form of $(\cdot, A_s, \cdot, \cdot)$ s.t. $A_s \cap A_{s_i} \neq \emptyset$ has been issued to *Spend* oracle.

Definition 4 (Traceability). Traceability ensures that the tracing authority can always successfully trace the real payer, who conducted the transaction, and the tracing authority can generate a valid proof to this trace for public verification. A protocol has the property of traceability if for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix} \mathsf{Trace}(sk_M,tx,CT) \\ =(pk_s,\psi) \\ \mathsf{Judge}(pk_s,\psi,pk_M, \\ CT)=0 \end{matrix} : \begin{matrix} (pp)\leftarrow\mathsf{Setup}(1^\lambda);(tx,\pi,S)\leftarrow \\ \mathcal{A}_1^{AddGen,ActGen,Spend,Corrupt}(pp) \\ \mathsf{Verify}(tx,\pi,S,CT)=1 \end{matrix}\right]\leq negl(\lambda).$$

12. We note that tx contains transaction description m, the input address set A and output address R, as defined in system component

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

4 PRELIMINARIES

We recall preliminaries used throughout this paper in this section.

4.1 Bilinear Maps

Let G_1, G_2 and G_T denote cyclic groups of the same prime order p. A bilinear pairing $e : G_1 \times G_2 \rightarrow G_T$ is a map satisfying the following properties [31]:

Bilinear: $e(u^a, v^b) = e(u, v)^{ab}$, for all $u \in G_1$, $v \in G_2$ and $a, b \in Z_p$.

Non-degenerate: $e(g_1, g_2) \neq 1$, where g_1 and g_2 are generators of group G_1 and G_2 respectively.

Computational: $e(u, v)$ can be computed efficiently for all $u \in G_1$ and $v \in G_2$.

The aforementioned bilinear maps are asymmetric ones. It is called a symmetric bilinear pairing when G_1=G_2, which is widely used in a number of protocols.

4.2 Intractable Assumptions

Let λ be a security parameter and $G =< g >$ denotes a cycle group of prime order p. Then we define the following assumptions in such group.

1) ***Discrete Logarithm (DL) Assumption.***

DL problem is that, given a tuple $(g, g^a) \in G$ and output $a \in Z_p$. DL assumption holds if for any polynomial-time algorithm $\mathcal{A}$, the following advantage $\mathbf{Adv}^{\mathbf{DL}}_{\mathcal{A}}$ is negligible in λ.

$$\mathbf{Adv}^{\mathbf{DL}}_{\mathcal{A}}(\lambda) = Pr\Big[\mathcal{A}(g, g^a) \rightarrow a\Big]$$

2) ***Decisional Diffie-Hellmam (DDH) Assumption*** [31]

DDH problem states that given a tuple $(g, g^a, g^b, g^{(1-x)ab+xc}) \in G$ and output $x \in \{0, 1\}$. DDH assumption holds if for any polynomial-time algorithm $\mathcal{C}$, the following advantage $\mathbf{Adv}^{\mathbf{DDH}}_{\mathcal{C}}(\lambda)$ is negligible in λ.

$$\mathbf{Adv}^{\mathbf{DDH}}_{\mathcal{C}}(\lambda) = \Big|Pr\Big[\mathcal{C}(g, g^a, g^b, g^{ab}) =1\Big] - Pr\Big[\mathcal{C}(g, g^a, g^b, g^c) =1\Big]\Big|$$

4.3 Accumulators with One-Way Domain

An accumulator [16] can accumulate a set of elements into a single value and for each given element, there exists a witness to prove that it has been incorporated into the accumulator indeed. Specifically, let $\mathcal{F} = \{F_\lambda\}$ be a sequence of families of functions and $\mathcal{X} = \{X_\lambda\}$ a sequence of families of finite sets, which satisfy $F_\lambda = \{f : U_f \times X_f \rightarrow U_f\}$ and $X_\lambda \subseteq X_f$ for all $\lambda \in N$. An accumulator family contains the following algorithms.

- $\mathsf{ACC.Gen}(1^\lambda)$. This is a probabilistic algorithm that takes as input a security parameter λ, and outputs a description desc and some auxiliary information.
- $\mathsf{ACC.Eval}(\mathsf{desc}, X)$. This is a probabilistic algorithm that takes as input the description desc and $X \subset X_\lambda$. It outputs an accumulated value $v = f(u, X)$, where $f \in F_\lambda, u \in U_f$ and $f(u, X) = f(\cdots f(u, x_1) \cdots x_n), X = \{x_1, \cdots, x_n\} \subset X_\lambda$.
- $\mathsf{ACC.Wit}(\mathsf{desc}, x, X)$. This is a probabilistic algorithm that takes as input the description desc, $X \subset X_\lambda$ and $x \in X$. It outputs a witness ω, where $v = f(w, x)$.

An accumulator satisfies the following properties.

- Efficient generation: The $\mathsf{ACC.Gen}$ is efficient that runs in polynomial time.
- Efficient evaluation: Any $f \in F$ is computable in polynomial time in λ.
- Quasi-commutativity: For all $\lambda \in N$, $f \in F_\lambda$, $u \in U_f$ and $x_1, x_2 \in X_\lambda$, it holds that $f(f(u, x_1), x_2) = f(f(u, x_2), x_1)$.

The pair $(\mathcal{F}, \mathcal{X})$ is an accumulator with one-way domain if it has the following properties.

- Collision-resistance: An accumulator is one-way domain if for all $\lambda \in N$ and an adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix} X \subset X_\lambda \wedge x \in X_f \setminus X \\ x \in U_f \wedge f(\omega, x) = f(u, X) \end{matrix} : \begin{matrix} f \leftarrow F_\lambda; u \leftarrow U_f \\ (x, \omega, X) \leftarrow \\ \mathcal{A}(f, U_f, u) \end{matrix}\right] \leq negl(\lambda)$$

- One-way domian: Let $\{Y_\lambda\}, \{R_\lambda\}$ be two sequences of families of sets associated with $\{X_\lambda\}$ and each R_λ is an efficient verifiable, samplable relation over $Y_\lambda \times X_\lambda$. It is infeasible to efficiently compute a witness $y' \in Y_\lambda$ for an x sampled from X_λ with W. Specifically, for any adversary $\mathcal{A}$,

$$\Pr\Big[(y', x) \in R_\lambda : (y, x) \leftarrow W(1^\lambda); y' \leftarrow \mathcal{A}(1^\lambda, x)\Big] \leq negl(\lambda)$$

4.4 Signature of Knowledge

Signature of Knowledge (SoK) for a NP-relation $\mathcal{R}$ with the corresponding language $L = \{y : \exists x,\ s.t. (x, y) \in \mathcal{R}\}$ consists of the following algorithms.

- $\mathsf{Gen}(1^\lambda)$: On input a security parameter λ, this algorithm outputs a public parameter par.
- $\mathsf{Sign}(m, x, y)$: On input a message m and a pair $(x, y) \in \mathcal{R}$, it outputs a SoK π.
- $\mathsf{Verf}(m, \pi, y)$: On input a message m, a SoK π and a statement y, it outputs $0/1$.

A SoK is *SimExt-secure* [32] if it satisfies correctness, simulatability and extractability.

Correctness. For any message m and a pair $(x, y) \in \mathcal{R}$, it holds that

$$\Pr\left[\mathsf{Verf}(m, \pi, y) = 1 : \begin{matrix} \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda) \\ \pi \leftarrow \mathsf{Sign}(m, x, y) \end{matrix}\right] \geq 1 - negl(\lambda),$$

Simulatability. There exists a polynomial-time simulator Sim=(SimGen,$\mathsf{SimSign}$) s.t. for any PPT adversary $\mathcal{A}$,

$$\Big|\Pr[b = 1 : (\mathsf{par}, td) \leftarrow \mathsf{SimGen}(1^\lambda); b \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{par})] - \Pr[b = 1 : \mathsf{par} \leftarrow \mathsf{Gen}(1^\lambda); b \leftarrow \mathcal{A}^{\mathsf{Sign}}(\mathsf{par})]\Big| \leq negl(\lambda)$$

where td is an additional trapdoor in Sim to simulate the signatures without the witness.

Extractability. There exists an extractor Ext s.t. for any PPT adversary $\mathcal{A}$,

$$\Pr\left[\begin{matrix} (x, y) \in \mathcal{R} \vee (m, y) \in Q \\ \vee \mathsf{Verf}(m, y, \pi) = 0 \end{matrix} : \begin{matrix} (\mathsf{par}, td) \leftarrow \mathsf{SimGen}(1^\lambda) \\ (m, y, \pi) \leftarrow \mathcal{A}^{\mathsf{Sim}}(\mathsf{par}) \\ x \leftarrow (\mathsf{Ext}(\mathsf{par}, td, m, y, \pi) \end{matrix}\right] \leq negl(\lambda)$$

where Q is a list of queries to $\mathsf{SimSign}$ Oracle that $\mathcal{A}$ made.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

4.5 Commitment

A commitment scheme allows a committer to commit to a selected message, which is hidden to others but can be revealed by the sender. Specifically, a commitment scheme consists of the following polynomial-time algorithms (CGen,Com,Open).

- $\mathsf{CGen}(1^\lambda)$: On input a security parameter λ, this algorithm outputs a public commitment key ctk.
- $\mathsf{Com}(ctk, m, r)$: On input a commitment key ctk, a message $m \in \{0,1\}^*$ and some randomness r, this algorithm outputs a commitment c.
- $\mathsf{Open}(c, m, r)$: On input a commitment c, the message m and the randomness r, it checks $c \stackrel{?}{=} \mathsf{Com}(ctk, m; r)$

A commitment scheme is secure if it satisfies the properties of binding and hiding defined as follows.

Hiding. Hiding requires that the commitment c reveals nothing about m. Specifically, for any polynomial-time adversary $\mathcal{A}$,

$$\left|\Pr\left[\mathcal{A}(c)=b: \begin{array}{c} ctk \leftarrow \mathsf{CGen}(1^\lambda) \\ (m_0,m_1) \leftarrow \mathcal{A}(ctk) \\ b \leftarrow \{0,1\}; \\ c \leftarrow \mathsf{Com}(ctk, m_b) \end{array}\right] - \frac{1}{2}\right| \leq negl(\lambda).$$

Binding. It is infeasible for a committer to generate a commitment c that can be opened as two different messages m_0, m_1. Specifically, for any polynomial-time adversary $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} Com(m_0,r_0) = \\ Com(m_1,r_1) \\ \wedge\, m_0 \neq m_1 \end{array} : \begin{array}{c} ctk \leftarrow \mathsf{CGen}(1^\lambda) \\ (m_0,m_1,r_0,r_1) \leftarrow \mathcal{A}(ctk) \end{array}\right] \leq negl(\lambda).$$

We now introduce the classic Pedersen commitment [33]. On input a parameter λ, CGen generates a cyclic group G with prime order q, two generators $g, h \in Z_q$ and outputs $ctk = (G, g, h)$. To commit to a message $m \in Z_q$, the committer generates a random $r \in Z_p$ and computes $c = \mathsf{Com}(ctk, m; r) = g^m h^r$. To Open a commitment c, the committer reveals m, r and everyone can check if $c = \mathsf{Com}(ctk, m; r)$. Pedersen commitment also satisfies the homomorphic property.

Homomorphic. For $\lambda \in N$, $m_0, m_1, r_0, r_1 \in Z_q$, it holds that,

$$Com(m_0, r_0) \cdot Com(m_1, r_1) = Com(m_0 + m_1, r_0 + r_1).$$

where + is the operation in Z_q and $\cdot$ is the operation in group G.

4.6 Variant ElGamal Encryption

In our protocol, we leverage a variant of the original ElGamal encryption[34] with the following components.

$\mathsf{KeyGen}(\lambda)$. On input a security parameter λ, it generates a cyclic group G with prime order q. $g \in G$ is a generator of G. Choose a random $x \in Z_q^*$ as a secret key, it outputs the public key as $z = g^x$. The key pair is (x, z)

$\mathsf{Enc}(z, m)$. On input a public key z and a message m, first choose a random α and encrypt m as $C = (c_1, c_2) = (z^\alpha, mg^\alpha)$. The ciphertext of m is (c_1, c_2).

$\mathsf{Dec}(x, C)$. The entity who has the knowledge of the secret key x can decrypt the ciphertext and recover the message m as $m = c_2 / {c_1}^{x^{-1}}$

It is shown in [35] that in ElGamal encryption, the public key and the base can be interchangeable and both schemes are IND-CPA secure under DDH assumption. In our construction, it is easy to generate zero-knowledge proof with the variant ElGamal encryption

5 Traceable Monero System

In this part, we firstly introduce how the proposed Traceable Monero system works, and then describe the concrete protocol of the proposed Traceable Monero and finally give an instantiation of the Signature of Knowledge used in the system.

5.1 Workflow of Traceable Monero

The proposed traceable Monero system is based on an improved Monero. As shown in Fig. 2, the main processes of the proposed Traceable Monero system are Transaction generation, Transaction on chain and Tracing.

Transaction generation: To launch a transaction (spending), a payer needs to generate keys and mint coins first. When generating one-time key pairs for a payee, a tag is also produced for each one-time address, which is used to trace the long-term public key of the payee when he acts as a malicious payer. Every payer and payee involved in a transaction have their own tags, where a payee's tag is computed by his/her payer in the current transaction and a payer's tag is generated in the previous transaction when he was a payee. To mint a coin for a one-time address before conducting a transaction, the payer needs to compute a homomorphic commitment with the currency amount and a random number to hide the amount of currency in a transaction. A payer employs a linkable ring signature to sign the transaction with his one-time secret key to hide his identity and broadcasts the transaction Tx into the P2P network.

Transaction on chain: This part is the same as the underlying Monero [13]. Specifically, when miners receive the transaction generated by the users in the system, they validate the transactions first and generate a block for the valid transactions he collects. Then, the miners compete to broadcast a block on the blockchain via *Consensus*, say proof-of-work. Anyone can validate the correctness of the proof-of-work efficiently.

Tracing: To trace the long-term public key of a payer, the tracing authority decrypts the ciphertext in the tag using his own private key to obtain the payer's long-term public key. When tracing a one-time address of a payer, the tracing authority decrypts the ciphertext in the transaction and gets the index s in a ring, and thus can find the corresponding one-time public keys. Moreover, with the exposure of a real input (i.e., paying) account, the tracing authority can trace back to the accounts, from which the money is transferred to the current account, and trace forward, to locate the subsequent accounts that receive payment originated from the current account and thus the tracing authority can trace the money flow.

5.2 Traceable Monero Construction

A concrete construction is presented in this part, in which several cryptographic building blocks are leveraged.

Suppose there are n groups of input accounts $A = \{(pk_{in,i}^{(k)}, cn_{in,i}^{(k)})\}_{1\leq i\leq n, 1\leq k\leq m}$ in the sytem and the real payer is the s-th group denoted by $A_s = \{(pk_{in,s}^{(k)}, cn_{in,s}^{(k)})\}_{1\leq k\leq m}$. The public keys in the accounts can be regarded as a matrix with each group of public keys in a column and each row k of the public keys being accumulated into a single value v_k by an accumulator f. Each group of public keys are set as $\{pk_{in,i}^{(k)} \cdot u^i\}$. An extra row of $\widetilde{pk_i} \cdot u^i$ is computed in the last line of the matrix to guarantee the total balance in each transaction.

Traceable Monero system. Define $f=$ (ACC.Gen, ACC.Eval, ACC.Wit) as an accumulator with one-way domain G_q and SoK=(SoK.Gen,SoK.Sign,SoK.Verf) as a signature of knowledge. The concrete protocol is defined as follows on the basis of the underlying f and SoK.

Setup(1^λ). Let the description of the accumulator be desc = $\mathsf{ACC.Gen}(1^\lambda)$ and denote par as par = $\mathsf{SoK.Gen}(1^\lambda)$. Choose $h_0, h_1, h_3, \tilde{h}, u \in G_q$ randomly and the public parameter is $pp = (1^\lambda, \mathsf{desc}, \mathsf{par}, h_0, h_1, h_3, \tilde{h}, u, H)$, where $H : \{0,1\}^* \to G_q$ denotes a collision resistant hash function. [14].

13. https://getmonero.org/

14. We note that the parameters h_0, h_1, h_3 can be generated as $h_i = H(`traceable\_monero\_i')$, which are publicly verifiable.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

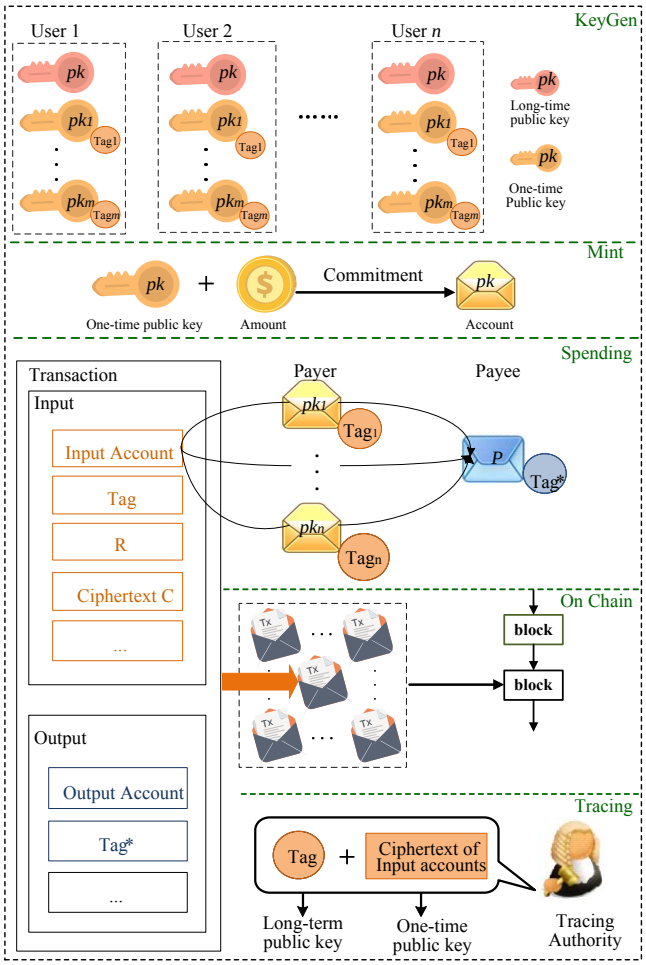


Fig. 2
Workflow of the proposed Traceable Monero

KeyGen(pp). This algorithm generates public keys as users' addresses in the system as follows. Let $(\omega \in G_q, z = {h_0}^{\omega})$ be the key pair of the tracing authority. A payer wants to pay to a payee whose long-term public key is $(A' = {h_0}^{a'}, B' = {h_0}^{b'})$, where the pair (a', b') is the corresponding long-term private key. When pay to the payee, the payer generates a one-time public key as well as a corresponding tag for the payee. Specifically, to generate the one-time address, the payer chooses a random $r' \in G_q$ and computes $R' = {h_0}^{r'}$. Let $h = H({A'}^{r'}, B')$. The payee's one-time public key is $y = B'{h_0}^{h}$, and the corresponding one-time private key is $x = b' + H({R'}^{a'}, B')$. To produce a tag, the payer first encrypts the payee's long-term public key B' with tracing authority's public key z using variant ElGamal encryption as $ct = z^h$. And then compute the tag with the SoK as follows:

$$SoK\left\{\left((A_i, B_i), r'\right) : \bigvee_{i=1}^{l} R' = {h_0}^{r'} \wedge y = B'_i {h_0}^{H({A'_i}^{r'}, B'_i)} \wedge ct = z^{H({A'_i}^{r'}, B'_i)}\right\}$$

In the above setting, in order to trace the real spender with the help of the tags, we modified the underlying CryptoNote to add the part of the public key B' into the hash function H. By doing so, the payee can still recover the corresponding private key and the payer can bind (A', B') with the one-time key y, as it is hard to find another (A'', B'') to have the same y. More detailed proof is shown in section 6.

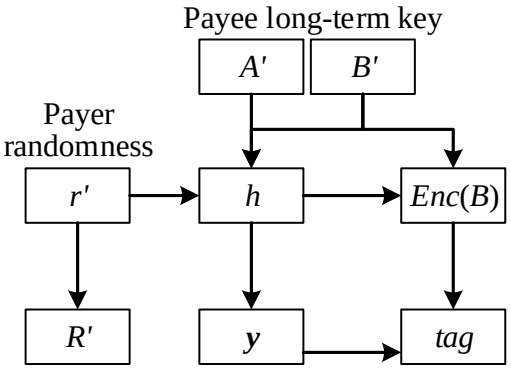


Fig. 3
Generating one-time address

Mint(pk, a). This algorithm mints a coin for a public key address with an amount a as follows. Pick a random $r \in Z_q$, and compute Pedersen commitment c with the amount a and the randomness r as $c = \mathsf{Com}(ctk, a; r) = {h_0}^{r} {h_1}^{a}$. The output is $(cn, ck) = (c, (r, a))$. We denote the account as $act = (pk, cn)$, and the corresponding secret key is $ask = (sk, ck)$.

Spend(m, K_s, A_s, A, R, z). Without loss of generality, we define the following parameters. The secret keys are denoted as $K_s = \left\{ask_{in,s}^{(k)} = (sk_{in,s}^{(k)}, (r_{in,s}^{(k)}, s_{in,s}^{(k)}))\right\}_{1 \le k \le m}$. Let the output address be $R = \{pk_{out,j}\}_{1 \le j \le t}$. To spend a coin, a payer needs to do the following,

1) The payer chooses a random $r_{out,j} \in Z_p (1 \le j \le t)$ and mints the coin for an output address $pk_{out,j} \in R (1 \le j \le t)$ as $cn_{out,j} = c_{out,j} = \mathsf{Com}(ctk, a_{out,j}; r_{out,j}) = {h_0}^{r_{out,j}} {h_1}^{a_{out,j}}$, where $\sum_{k=1}^{m} a_{in,s}^{(k)} = \sum_{j=1}^{t} a_{out,j}$. The payer adds the output account $act_{out,j} = (pk_{out,j}, cn_{out,j})$ to A_R, and sends the coin key $ck_{out,j} = (r_{out,j}, a_{out,j})$ to the payee holding address $pk_{out,j}$.
2) The payer sets $\widetilde{sk_s} = \sum_{k=1}^{m} sk_{in,s}^{(k)} + \sum_{k=1}^{m} r_{in,s}^{(k)} - \sum_{j=1}^{t} r_{out,j}$ and computes $\widetilde{pk_i}$ as

$$\widetilde{pk_i} = \prod_{k=1}^{m} pk_{in,s}^{(k)} \cdot \prod_{k=1}^{m} cn_{in,i}^{(k)} / \prod_{j=1}^{t} cn_{out,j}.$$

It can be concluded that $\widetilde{pk_s} = {h_0}^{\widetilde{sk_s}}$ if $\sum_{k=1}^{m} a_{in,s}^{(k)} = \sum_{j=1}^{t} a_{out,j}$ holds.
3) The payer generates a proof π to show that the transaction is spent properly as follows. The payer computes the accumulated value $v_k = \mathsf{ACC.Eval}(\mathsf{desc}, \{y_i^{(k)} \cdot u^i\})$ and the witness $w_s = \mathsf{ACC.Eval}(\mathsf{desc}, \{y_i^{(k)} \cdot u^i | i \ne s\})$ for $1 \le k \le m+1$ to show that $z_s^{(k)} = y_s^{(k)} \cdot u^s$ is indeed accumulated in v_k. For $1 \le k \le m$, the payer computes $s_k = \tilde{h}^{x_s^{(k)}}$ according to the secret key $x_s^{(k)}$ of the account as the serial number to uniquely defined the address. Choose a random $\tilde{a}$, and encrypt the index γ with tracing authority's public key z as $ct_1 = z^{\tilde{a}}, ct_2 = h_3^{\gamma} {h_0}^{\tilde{a}}$. The ciphertext is $CT = (ct_1, ct_2)$. Then invoke Sign algorithm to produce SoK π on tx as follows.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

$$SoK\left\{\begin{array}{c}(\{\omega_k, z_k, x_k\}_{k=1}^m, \gamma, \tilde{a}) : ct_1 = z^{\tilde{a}} \wedge ct_2 = {h_3}^{\gamma} {h_0}^{\tilde{a}} \wedge \\ f(\omega_{m+1}, z_{m+1}) = v_{m+1} \wedge z_{m+1} = {h_0}^{x_{m+1}} u^{\gamma} \wedge \\ f(\omega_1, z_1) = v_1 \wedge z_1 = {h_0}^{x_1} u^{\gamma} \wedge s_1 = \tilde{h}^{x_1} \wedge \\ \vdots \\ f(\omega_m, z_m) = v_m \wedge z_m = {h_0}^{x_m} u^{\gamma} \wedge s_m = \tilde{h}^{x_m}\end{array}\right\}(tx)$$

Finally, output (tx, π, S, CT), where $S = \{s_1, s_2, \cdots, s_m\}$ and $CT = (ct_1, ct_2)$.

Verify(tx, π, S, CT)**.** Given $A = \{(pk_{in,i}^{(k)}, cn_{in,i}^{(k)})\}_{1 \le i \le n, 1 \le k \le m}$, $A_R = (pk_{out,j}, cn_{out,j})_{1 \le j \le t}$, and the ciphertext CT from the transaction tx, one can compute $\widetilde{pk_i} = \prod_{k=1}^m pk_{in,s}^{(k)} \cdot \prod_{k=1}^m cn_{in,i}^{(k)} / \prod_{j=1}^t cn_{out,j}$, $v_k = \mathsf{ACC.Eval}(\mathsf{desc}, \{y_i^{(k)} \cdot u^i\})$ and $v_{m+1} = \mathsf{ACC.Eval}(\mathsf{desc}, \{\widetilde{pk_i} \cdot u^i\})$. Then verify the SoK with $(v_1, \cdots, v_{m+1})$, serial number S, transaction tx and proof π by checking

$$\mathsf{Verf}(tx, ct_1, ct_2, (v_1, \cdots, v_{m+1}, s_1, \cdots, s_m), \pi) \stackrel{?}{=} 1.$$

BlockGen$(\{tx\}, blk_{n-1}) \rightarrow (non, blk_n)$[15]. A miner collects some valid transactions in the P2P network and generates a Merkle Hash Tree for those transactions. Then the miner includes the root of the Merkle Hash Tree $Root$ together with the hash of the previous block Pre_Hash in the current block. Finally, the miner needs to compute a proper nonce non to make the hash of the block less than some target value.

BlockVer$(blk_n) \rightarrow (1/0)$. On input the current block blk_n, the miners in the P2P network check whether the hash value of the block is less than the target value and output 1 or 0.

Trace(ω, tx, CT**).** Two tracing mechanisms are provided in this system. One is to trace one-time public key and the other one is to trace long-term public key as a supplement.

To trace a one-time public key by given the ciphertext $CT = (ct_1, ct_2)$ from a transaction tx, the tracing authority decrypts the ciphertext using his secret key ω by computing $p = {ct_2 / {ct_1}^{\omega^{-1}}}$. Then he tries to find $\gamma \in [1, n]$ by testing which γ satisfies $p = {h_3}^{\gamma}$, and then the corresponding public key can be traced. By producing a proof ψ' as $SoK\Big\{(\omega) : z = {h_0}^{\omega} \wedge ct_1 = (\frac{ct_2}{{h_3}^{\gamma}})^{\omega}\Big\}(\gamma)$, the tracing authority can prove himself that γ is indeed the index of the payer in the group.

To trace a long-term public key, the tracing authority decrypts the tag by computing $B' = {y / {ct'}^{\omega^{-1}}}$ and thus can get the corresponding long-term public key B'. By producing the $SoK\Big\{(\omega) : z = {h_0}^{\omega} \wedge ct' = (\frac{y}{B'})^{\omega}\Big\}$ as a proof ψ, the tracing authority can prove that the tag is opened correctly by the authority himself and B_i is indeed the payer. Note that the proof ψ can be publicly verified.

Judge$(pk_s, \psi(\psi'), z, CT)$. On input the payer's public key, the proof generated by the tracing authority, the tracing authority's public key z and the ciphertext, anyone can validate the proof ψ and ψ' by checking the following two equations respectively.

$$\mathsf{Verf}(B', (z, y, ct), \psi) \stackrel{?}{=} 1$$

$$\mathsf{Verf}(\gamma, (z, ct_1, ct_2), \psi') \stackrel{?}{=} 1$$

5.3 Extension

In this subsection, we show some extensions of our construction.

15. Note that this algorithm and **BlockVer** are the same as the underlying Monero. We put them here to make the whole construction self-contained. More details can be found at https://getmonero.org/ and [9]

5.3.1 Sharing tracing capability

To avoid single point of failure, authority's tracing capability of the can be shared among several authorities according to an access structure [36], such that a qualified subset of the authorities can trace suspicious transactions. In this case, we provide a set of tracing authorities, in which users may not trust all of them, but can choose the subset of the Tracing Authorities they believe in to conduct the tracing. An easy way to achieve this is shown as follows. Suppose there are n Tracing Authorities whose public key is $z_1 = h_0^{\omega_1}, \cdots, z_n = h_0^{\omega_n}$, the payer can choose t out of n Tracing Authorities and aggregate their public key into a single $z = \prod_{i=1}^t z_t$. Then the t Tracing Authorities can decrypt to reveal the payer when tracing is needed.

5.3.2 Permissioned Blockcahin

The blockchain in our system is public blockchain, and we also consider the permissioned blockchain, such as consortium blockchain, which can be a ledger among members (e.g. banks). More specifically, in the Consortium blockchain, each bank can generate transactions on the chain but can hide their transaction metadata from other banks, and a tracing authority is required by regulation to determine illegal transactions, such as money laundering. In this way, the tracing result can be linked to the real-world identity and the lawbreaker can be punished.

5.4 Instantiation of the SoK

The SoK mentioned in subsection 5.2 can be instantiated by Fiat-Shamir paradigm [37] incorporated with zero-knowledge protocols in [16]. For the SoK in *KeyGen*, the instantiation of h in the proof needs to employ zk-snark [14]. For the SoK in Spend, more details are shown below.

For simplicity, we use the symmetric bilinear pairing in our construction. Let g_0, g_1, g_2 be generators of group G_1 and $h_0, h_1, h_2, h_3, \tilde{h}, u$ be generators of G_q, which is a subgroup of Z_p^*. Firstly, PoK_1 is to prove the knowledge of $(x_k, z_k, \gamma, \tilde{a})$ s.t. $z_k = {h_0}^{x_k} \cdot u^{\gamma}, s_k = \tilde{h}^{x_k}, ct_1 = z^{\tilde{a}}$ and $ct_2 = {h_3}^{\gamma} g^{\tilde{a}}$ in a zero-knowledge approach. Specifically, we have,

$$PoK_1\Big\{(x_k, z_k, r_k, t, \gamma, \tilde{a}) : C_k = {g_0}^{z_k} {g_1}^{r_k} \wedge z_k = {h_0}^{x_k} \cdot u^{\gamma} \wedge s_k = \tilde{h}^{x_k} \wedge D = {h_1}^{\gamma} {h_2}^{t} \wedge ct_1 = z^{\tilde{a}} \wedge ct_2 = {h_3}^{\gamma} g^{\tilde{a}}\Big\}$$

This protocol can be split into the following two sub-protocols,

$$PoK_{1,1}\Big\{(z_k, r_k) : C_k = {g_0}^{z_k} {g_1}^{r_k}\Big\}$$

$$PoK_{1,2}\Big\{(x_k, r_k, t, \gamma, a) : C_k = {g_0}^{{h_0}^{x_k} \cdot u^{\gamma}} {g_1}^{r_k} \wedge s_k = \tilde{h}^{x_k} \wedge D = {h_1}^{\gamma} {h_2}^{t} \wedge ct_1 = z^{\tilde{a}} \wedge ct_2 = {h_3}^{\gamma} g^{\tilde{a}}\Big\}$$

The first protocol is a standard protocol to prove the discrete logarithm of a statement while the second one is to prove the knowledge of a double discrete logarithm [35].

Secondly, PoK_2 is to show that z_k is indeed accumulated in the value v_k without leaking any knowledge of z_k and its witness w_k. Specifically, we have,

$$PoK_2\Big\{(w_k, z_k, r_k) : e(w_k, {g_0}^{z_k} {g_0}^{\alpha}) = e(v_k, g_0) \wedge C_k = {g_0}^{z_k} {g_1}^{r_k}\Big\}$$

To instantiate PoK_2, we choose $\tau_1, \tau_2 \in Z_p$, compute $w_{k,1} = {g_0}^{\tau_1} {g_1}^{\tau_2}, w_{k,2} = w_k {g_1}^{\tau_1}$, and implement the following *PoK*,

$$PoK_2'\Big\{(\tau_1, \tau_2, z_k, \delta_1, \delta_2, r_k) : w_{k,1} = {g_0}^{\tau_1} {g_1}^{\tau_2} \wedge w_{k,1}^{z_k} = {g_0}^{\delta_1} {g_1}^{\delta_2} \wedge \frac{e(w_{k,2}, {g_0}^{\alpha})}{e(v_k, g_0)} = e(w_{k,2}, g_0)^{-z_k} e(g_1, g_0)^{\delta_1} e(g_1, g_0^{\alpha})^{\tau_1} \wedge C_k = {g_0}^{z_k} {g_1}^{r_k}\Big\}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

where $\delta_1 = \tau_1 z_k$ and $\delta_2 = \tau_2 z_k$.

6 Security Proofs

In this section, we provide detailed security proofs of balance, anonymity and traceability of the proposed scheme.

6.1 Proof of Balance

Theorem 1. *The proposed Traceable Monero achieves the property of balance if DL assumption holds,* ACC *is an accumulator with one-way domain and* SoK *is SimExt-secure.*

Proof(Sketch). We borrowed the idea from [15] to achieve the proof. If there is an adversary $\mathcal{A}$ that breaks the balance of the proposal with a non-negligible probability ϵ, then we can construct another algorithm $\mathcal{A}'$ to solve DL problem.

Given a DL instance $(h_0, h_1 = g^{\alpha})$, $\mathcal{A}'$ runs $\mathsf{ACC.Gen}(1^{\lambda})$ and $\mathsf{Gen}(1^{\lambda})$ to generates desc and par. Then choose β and γ randomly to compute $\tilde{h} = {h_0}^{\beta}$ and $h_3 = {h_0}^{\gamma}$. $\mathcal{A}$ can make queries to AddGen, ActGen and Corrupt Oracles for at most q_{ad}, q_{ac} and q_{co} times respectively. $\mathcal{A}'$ chooses x_i for each $i \in [q_{ad}]$. $\mathcal{A}'$ chooses a $j^* \in [q_{ad}]$, and sets the queried addresses as follows. For $i \neq j^*$, $pk_i = {h_0}^{x_i}$ and for $i = j^*$, $pk_i = {h_1}^{x_i}$. Now $\mathcal{B}$ simulates the following oracles.

- $\mathsf{AddGen}(i)$. Return the address pk_i generated as the way mentioned above for the i-th query.
- $\mathsf{ActGen}(i, a_i)$. Given i and an amount a_i, choose a random r_i and set $c_i = {h_0}^{r_i} {h_1}^{a_i}$. Add $(cn_i, ck_i) = (c_i, (r_i, a_i))$ to list $\mathcal{I}$, $i, act_i = (pk_i, cn_i)$ to list $\mathcal{G}$ and $s_i = \tilde{h}^{sk_i}$ to list $\mathcal{S}$ respectively.
- $\mathsf{Spend}(\mathsf{m}, A_s, A, R)$. Given the inputs, $\mathcal{A}'$ generates (tx, π, S) as the way in the proposal if $act_{j^*} \notin A_s$. Otherwise, $\mathcal{A}'$ generates π by simulating SoK. Specifically, $\mathcal{A}'$ invokes Sim and generates the simulated spending without using the witness, which is indistinguishable from $\mathcal{A}$'s view.
- $\mathsf{Corrupt}(i)$. On input the query i, if $i = j^*$, then $\mathcal{A}'$ aborts. Otherwise, $\mathcal{B}$ returns x_i and puts s_i as well as (s_i, a_i) to list $\mathcal{C}$ and $\mathcal{B}$.

Finally, $\mathcal{A}$ outputs $(act'_1, \cdots, act'_{\mu}, \mathcal{S}_1, \cdots, \mathcal{S}_v)$, where $\mathcal{S}_i = (tx_i, \pi_i, S_i)$ and all transactions are paid to $\mathcal{A}'$.

Now let's see the way to solve a DL instance if $\mathcal{A}$ wins the game above. We denote $E_i = \{a_{i,j} : (s_{i,j}, z_{i,j}) \in \mathcal{B} \wedge s_{i,j} \in \mathcal{S}_i \cap \mathcal{C}\}$ for simplicity.

- Case 1: $\forall i \in [v]$, $\mathcal{S}_i \backslash \mathcal{C} = \varnothing$. From the winning condition $\sum_{a_{i,j} \in E} a_{i,j} < \sum_{a_{i=1}^{v} \in E} a_{out,i}$, we know that there exists some $i^* \in [v]$ s.t. $\sum_{a_{i^*,j} \in E_{i^*}} a_{i^*,j} < \sum_{a_{i=1}^{v} \in E} a_{out,i^*}$. From the serial number, we can find the group $\{act_{i^*,j}\}$ of the spent accounts in A_{i^*}, where $act_{i^*,j} = (pk_{i^*,j}, cn_{i^*,j})$ and the secret key is $ask_{i^*,j} = (sk_{i^*,j}, (r_{i^*,j}, a_{i^*,j}))$. Then we can compute $\widetilde{pk}_{i^*}$ as $\widetilde{pk}_{i^*} = \prod_{j=1}^{m} pk_{i^*,j} \cdot \prod_{j=1}^{m} cn_{i^*,j} / cn_{out,i^*} = {h_0}^{\sum_{j=1}^{m} sk_{i^*,j} + \sum_{j=1}^{m} r_{i^*,j} - r_{out,i^*}} \cdot {h_0}^{\alpha(\sum_{j=1}^{m} a_{i^*,j} - a_{out,i^*})}$. We also know from the definition that $\mathcal{S}_i \notin \mathcal{T}$ for all $i \in [v]$, thus we can use Ext to extract a witness $\left(\{(x_{i^*,j}, z_{i^*,j}, sk_{i^*,j})\}_{j=1}^{m}, (\tilde{w}_{i^*}, \tilde{z}_{i^*}, \tilde{sk}_{i^*}), \gamma_{i^*}\right)$ With $\widetilde{sk}_{i^*}$, the DL instance (h_0, h_1) can be solved by computing

$$\alpha = \frac{\widetilde{sk}_{i^*} - (\sum_{j=1}^{m} sk_{i^*,j} + \sum_{j=1}^{m} r_{i^*,j} - r_{out,i^*})}{\sum_{j=1}^{m} a_{i^*,j} - a_{out,i^*}}.$$

- Case 2: $\forall i \in [v]$, $\mathcal{S}_i \backslash \mathcal{C} \neq \varnothing$. Let $\mathcal{J} = \{i \in [v] : \mathcal{S}_i \backslash \mathcal{C} \neq \varnothing\}$, $\mathcal{S}_{\mathcal{J}} = \bigcup_{i \in \mathcal{J}} (\mathcal{S}_i \backslash \mathcal{C})$ and the size of $\mathcal{S}_i \backslash \mathcal{C}$ is l_i. Then check if $s_{j^*} \in \mathcal{S}_{\mathcal{J}}$. If so, there exists $i^* \in \mathcal{J}$ s.t $s_{j^*} \in \mathcal{S}_{i^*}$. Now we can extract a witness including sk_{j^*} s.t. $pk_{j^*} = {h_0}^{sk_{j^*}} = \alpha^{x_{j^*}}$. Thus the DL problem can be solved by computing $\alpha = sk_{j^*} / x_{j^*}$.

Thus, the proof is completed.

6.2 Proof of Anonymity

In this subsection, we describe the proof of anonymity.

Theorem 2. *The proposed Traceable Monero achieves the property of anonymity if the SoK satisfies extractability, homomorphic commitment is perfect hiding, DDH assumption holds and encryption scheme is IND-CPA.*

Proof. We make use of game-based framework to present our proofs. We denote $\Pr[\mathsf{Win}_i]$ as the winning probability of an adversary (guessing correctly) in Game_i [15].

Game 0. This is the challenge game defined in section 2 (cf. Definition 6) and the challenge transaction is denoted as (tx^*, π^*, S^*, C^*), where $tx^* = (\mathsf{m}, A, A_R^{(b)})$, $S^* = (s_1^{(b)}, s_2^{(b)}, \cdots, s_m^{(b)})$ and $C^* = (ct_1, ct_2^{(b)})$. $\mathcal{A}$ outputs a guess b', from subsection 3, we can easily get

$$\Pr[\mathsf{Win}_0] = \Pr[b' = b]$$

Game 1. Game 1 is the same as Game 0 with one difference. The simulator replaces the algorithm Gen by SimGen to generate par and Sign by SimSign to generate the SoK. The *Spend* and challenge queries are computed without using the witness. Specifically, for a challenge query $(\mathsf{m}, A_{s0}, A_{s1}, A, R)$, the simulator calculates the statement and the corresponding witness as in **Game 0**, while computes the SoK: $\pi^* = \mathsf{Sim}(tx^*, ct_1, ct_2^{(b)}, (v_1, v_2, \cdots, v_{m+1}^{(b)}, s_1^{(b)}, \cdots, s_m^{(b)}))$ without the knowledge of the witness. Clearly, if there is a difference in the adversary's winning probability between **Game 0** and **Game 1**, then the adversary can be used to construct an algorithm to violate SimExt of SoK. The adversary's winning probability in **Game 1** satisfies the following equation due to the SimExt security of our employed SoK, where λ is the security parameter of this system.

$$|\Pr[\mathsf{Win}_1] - \Pr[\mathsf{Win}_0]| \leq negl(\lambda)$$

Game 2. Game 2 is the same as Game 1 with one difference. During the challenge phase, the simulator chooses the output address $pk_{out,j} \in R$ randomly and uniformly. Specifically, the simulator sets $h_1 = h_0^{\alpha}$ for some randomness α and changes the output address $pk_{out,j}$ by picking a random $\hat{r}_j$ from Z_p and setting $cn_{out,j} = {h_0}^{\hat{r}_j}$ for $1 \leq j \leq t$. Thus, the output account $A_R = \{(pk_{out,j}, cn_{out,j})\}_{j=1}^{t}$ is independent of b. In this case, we also have $\widetilde{pk}_i = \prod_{i=1}^{m} pk_i^{(k)} \cdot \prod_{i=1}^{m} cn_i^{(k)} / \prod_{j=1}^{t} cn_{out,j}$ is independent of b and so is $v_{m+1} = \mathsf{ACC.Eval}(\mathsf{desc}, \{\widetilde{pk}_i \cdot u^i\}_{i=1}^{n})$. Clearly, if there is a difference in the adversary's winning probability between **Game 1** and **Game 2**, we can use the adversary to construct an algorithm to break the perfect hiding property of homomorphic commitment. The adversary's winning probability in **Game 2** satisfies

$$\Pr[\mathsf{Win}_2] = \Pr[\mathsf{Win}_1]$$

Game 3. Game 3 is the same as Game 2 with one difference. The simulator changes the setting of the serial numbers when answering the challenging queries by choosing s_k $(k \in [m])$ randomly from G_q. We claim that no accounts in A_{sb} is corrupted and no *spend* queries (m, A_s, A, R) s.t. $A_s \bigcap A_{sb} = \phi$ is permitted, so the serial numbers $S^* = (s_1, \cdots, s_m)$ are fresh to the adversary. Clearly, the serial numbers are uniformly distributed from the adversary's view if DDH assumption holds.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

That is to say, if there is a difference in the adversary's winning probability between **Game 2** and **Game 3**, we can use the adversary to construct an algorithm to break the DDH problem. The adversary's winning probability in **Game 3** satisfies

$$|\Pr[\mathsf{Win}_3] - \Pr[\mathsf{Win}_2]| \leq negl(\lambda)$$

Game 4. Game 4 is the same as Game 3 with one difference. The simulator changes the ciphertext CT by encrypting a random number ξ. More specifically, the simulator sets $ct_1 = z^a, ct_2 = h_3{}^{\xi} g^a$, where a is also a random element. When answering a query, the simulator modifies Trace oracle into Trace' such that the simulator can run Ext to extract the public key pk from the SoK instead of decrypting the ciphertext. If the queried transaction tx is in the list $\mathcal{T}$, then the index can be returned by looking up the list. Otherwise, the simulator can get a witness by the SimExt security of the SoK. Then by IND-CPA security of the encryption scheme, the simulator can get a unique public key as a response. Clearly, if there is a difference in the adversary's winning probability between **Game 3** and **Game 4**, we can use the adversary to construct an algorithm to violate SimExt of SoK and IND-CPA of the encryption scheme. The adversary's winning probability in **Game 4** satisfies

$$|\Pr[\mathsf{Win}_4] - \Pr[\mathsf{Win}_3]| \leq negl(\lambda)$$

Wrapping up. The security analysis above shows that, the challenge transaction (tx^*, π^*, S^*, C^*) is independent of b if SoK is SimExt secure, the encryption scheme is IND-CPA secure and DDH assumption holds, thus this setting leaks no information about b to the adversary. There is only a negligible difference in winning probability for an adversary between **Game 0** and **Game 4**. So the probability of $\mathcal{A}$ in winning the *Anonymity* game is $\frac{1}{2} + \epsilon$, where ϵ is negligible.

6.3 Proof of Traceability

In this subsection, we provide the proof of traceability.

Theorem 3. *The proposed Traceable Monero achieves traceability if the SoK in the construction is sound and the encryption and SoK is with perfect correctness.*

Proof. Take one wallet from the ring as an example to prove the traceability. If the SoK π of the transaction tx in *Spend* algorithm is valid and sound, it implies that ct_1 and ct_2 are in the required form, thus the perfect correctness property of the leveraged public key encryption scheme guarantees ${h_3}^{\gamma}$ could be uniquely recovered. Moreover, γ is the index of the one-time public key in the ring, which is a small set and can be identified easily. Once the one-time public key y is decided, by the soundness of SoK in *KeyGen*, we can extract the element r' and the key pair (A_i', B_i') and R', y, ct are in the required form. We now need to prove that given a one-time key $y = B_i' {h_0}^{H({A_i}'^{r'}, {B_i}')}$, it is computational infeasible to find another key pair (A_i'', B_i'') as a collision that satisfies $y = B_i'' {h_0}^{H(A_i''^{r'}, {B_i}'')}$. We can see from the equation that, for a fixed y, one can choose an arbitrary B_i'', and the possibility to find a A_i'' s.t. $log_{h_0}(y/B_i'') = H(A_i''^{r'}, {B_i}'')$ is negligible, if we treat the hash function H as a random oracle. In this way we say that the (A_i', B_i') in this proof should be the real payee chosen by the payer. And from the correctness of variant ElGamal encryption, we can ensure that the decryption is correct, which means we can get $B' = y/{ct'}^{x^{-1}}$. The other part of *Trace* output is a SoK ψ as a proof generated by the racing authority with his private key to show that the tracing output is generated correctly according to the protocol. From the correctness and soundness of the SoK scheme, the proof ψ can be verified correctly. Thus our proposal achieves traceability if SoK is sound as well as the encryption scheme and SoK are with perfect correctness.

7 Performance of the System

We first show the efficiency analysis of our proposed traceable Monero system, then provide a comparison of the efficiency between our construction and that of the Monero. The parameters in Table 1 are as follows. m: the number of input accounts in each group; n: the number of group of input accounts; l: the length of the element in group Z_p; exp_1: an exponentiation operation in group G_1; exp_T: an exponentiation operation in group G_T ; exp_q: an exponentiation operation in group $G_q \subset Z_p$; $|G_1|$: the length of the element in group G_1, similarly for $|Z_p|$, $|G_q|$ and $|G_T|$.

In Table 1, we only pay attention to the expensive operations in the schemes, including exponentiation, multi-exponentiation and bilinear pairing, which are mainly involved in computing an accumulator and a SoK, and ignore the cost of other light computations. We count the operations in a concrete instantiation of SoK based on Fiat-Shamir framework [37]. We also take advantage of the trick the pre-computation, and neglect the time-consumption of the operations that can be pre-computed. For example, $\omega_{k,1} = {g_0}^{\tau_1} g_1^{\tau_2}$ and $e(g_1, g_0)^{\alpha}$ in PoK_2' can be pre-computed, since the elements in these equations are public parameters or can be selected by the payer in advance.

The details of the result are as follows. For a payer, $((n-1)+1)(m+1)\cdot$ $\exp_1$ is required in the computation of an accumulator, in which for each $1 \leq k \leq m+1$, $(n-1)$ $\exp_1$ is needed in the computation of a witness, and one $\exp_1$ to compute the accumulator v_k with the witness. Note that the cost of computing a mult-exp is not equal to that of the n times of an exp operation, but relies on the specific n. For example, a double-exponentiation has a cost of about 1.2 times, rather than 2 times, that of a single exponentiation by taking advantage of Shamir's Simultaneous Squaring Multi-Exponentiation Algorithm [34] and a triple-exponentiation has a cost of about 1.5 times that of a single exponentiation [38]. Besides, it needs $(m+1)$ $\exp_T$ and $(m+1)$ pairing in PoK_2' and 1.2 $l(m+1)$ $\exp_q$ in $PoK_{1,2}$, where l is involved because the double discrete logarithm in $PoK_{1,2}$ needs to be committed in bits. All the other operations in SoK can be pre-computed. The cost of a verifier is mainly on the operations of verifying a response in SoK and v_k. For the communication cost, we count the parameters that need to be stored and transferred to a verifier in SoK. We can see from table 1, the efficiency of Traceable Monero is comparable to that of the underlying Monero [15].

The implementation results are reported as follows. All the algorithms are conducted on a desktop with 64-bit Win 10 operating system and 16.0 GB RAM. The processor is Intel(R) Core(TM) i7-7700 CPU @ 3.6 GHz. The programs are written in C++ language with Visual Studio 2010 compiler. We invoke interfaces in Miracl library [16] to realize the operations in big integer and elliptic curve groups. Tate Pairing is used to implement an asymmetric bilinear map [17], $e : G_1 \times G_2 \to G_T$, where $|G_1| = |G_2| = order$, where $order$ is a 1024-bit prime. For each algorithm, we executed 100 times and get an average running time.

We first test the overhead in *Spend*[18] and *Verify* protocol compared to the underlying Monero. And we can see from Fig. 4 that the overhead in *Spend* and *Verify* are almost constant, i.e., 0.70 ms and 1.07 ms respectively. This is consistent with our empirical analysis, as the overhead is for generating and verifying the encryption on the column number γ, which doesn't rely on the number of the input accounts. We also test the efficiency of *Verify* protocol, which is much more important,

16. https://certivox.org/display/EXT/MIRACL.

17. We note that symmetric bilinear maps are also suitable in this implementation, however, the operations in asymmetric bilinear groups are more efficient.

18. For the overhead in *Spend* protocol, we only consider the time to spend a coin rather than to mint a coin.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

TABLE 1
Computational cost of our protocol and RingCT 2.0 [15]

Scheme	Spender	Verifier	Communication
[15]	$((n-1)+1)(m+1)\cdot exp_1$ $+(m+1)\cdot exp_T$+$(m+1)\cdot p$ +1.2$l(m+1)\cdot exp_q$	3.4$l(m+1)\cdot exp_q$ $(4.2+n+1.2l)(m+1)\cdot exp_1$ $4(m+1)\cdot exp_T+3(m+1)\cdot p$	$(l+4)(m+1)\|G_1\|$ $(3l+8)(m+1)\|Z_p\|$ $3l(m+1)\|G_q\|$+$(m+1)\|G_T\|$
Ours	$((n-1)+1)(m+1)\cdot exp_1$ $+(m+1)\cdot exp_T$+$(m+1)\cdot p$ +1.2$l(m+1)\cdot exp_q$	5.6$l(m+1)\cdot exp_q$ $(4.2+n+1.2l)(m+1)\cdot exp_1$ $4(m+1)\cdot exp_T$+3(m+1)$\cdot$	$(l+4)(m+1)\|G_1\|$ $(5l+8)(m+1)\|Z_p\|$ $5l(m+1)\|G_q\|$+$(m+1)\|G_T\|$

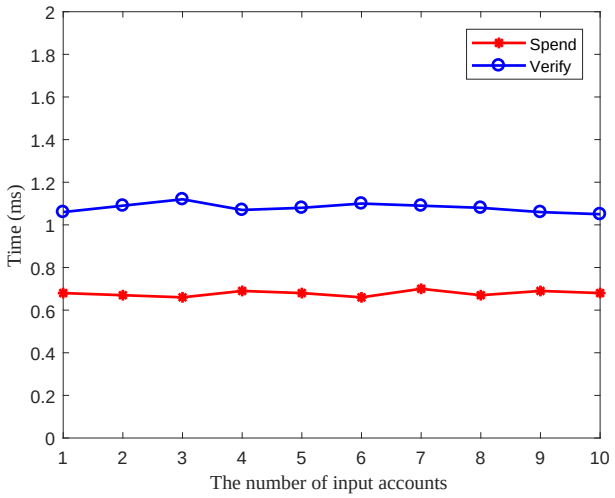


Fig. 4
Overhead of *Spend* and *Verify*

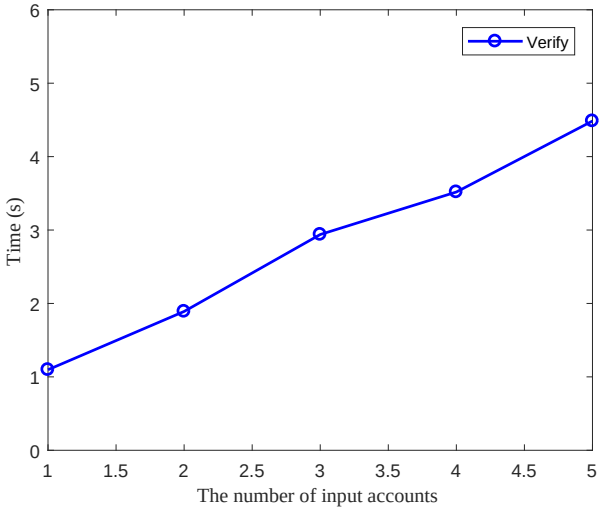


Fig. 5
Time consumption of *Verify*

since *Spend* protocol is executed only once in a transaction but *Verify* protocol can be run many times. The result is shown in Fig. 5. The time cost grows with the increase of the number of input accounts. This is in accordance with our scheme, as the effect of the mixins is eliminated by leveraging the accumulators with one-way domain, and the number of input accounts affect the number of zero-knowledge proof in *Verify*. For tracing authority to identify the long-term public key and one-time public key, it takes 6.331 ms.

8 Conclusion and Future Work

Balancing users' privacy and accountability remains a major challenge in decentralized cryptocurrencies. In this paper, we provided a positive answer towards resolving the conflict between two fundamental dichotomy of security requirements in cryptocurrency transactions. We introduced the concept of Traceable Monero and proposed a concrete construction of this new cryptocurrency. The proposed framework achieves the properties of correctness, balance, anonymity and traceability. Both the efficiency analysis and the implementation results show that the proposed system is comparable to the underlying Monero in efficiency.

We leave the following problems as future works. (1) Finding more methods to reduce the trust of the tracing authority except threshold mechanisms. (2) Designing new tracing approaches for other anonymous cryptocurrencies [13, 18, 39].

Acknowledgment

This work is supported by Data61, CSIRO and the National Key R&D Program of China (No.2017YFB0802000), the NSFC grant (61872229) and the Fundamental Research Funds for the Central Universities (GK201702004).

References

[1] Y. Li, Y. Yu, W. Susilo, G. Min, J. Ni, R. Choo. "Fuzzy Identity-Based Data Integrity Auditing for Reliable Cloud Storage Systems", *IEEE Trans. on Dependable and Secure Computing*, vol, 16, no. 1, pp. 72-83, 2019.

[2] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, G. Min. "Identity-Based Remote Data Integrity Checking with Perfect Data Privacy Preserving for Cloud Storage", *IEEE Trans. Information Forensics and Security*, vol. 12, no. 4, pp. 767-778, 2017.

[3] Y. Yu, Y. Li, B. Yang, W. Susilo, G. Yang, J. Bai. "Attribute-Based Cloud Data Integrity Auditing for Secure Outsourced Storage", *IEEE Trans. on Emerging Topics in Computing*. doi:10.1109/TETC.2017.2759329.

[4] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, B. Yang. "Efficient attribute-based encryption with attribute revocation for assured data deletion", *Information Science*, vol, 479, pp. 640-650, 2019.

[5] S. Nakamoto. "Bitcoin: A peer-to-peer electronic cash system". http://bitcoin.org/bitcoin.pdf, 2008.

[6] Y. Yu, Y. Li, J. Tian, J. Liu. "Blockchain-Based Solutions to Security and Privacy Issues in the Internet of Things", *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 12-18, 2018.

[7] N. van Saberhagen. "Cryptonote v 2.0". https://cryptonote.org/whitepaper. pdf, 2013.

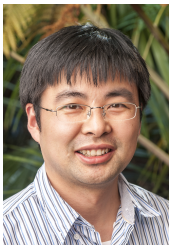
[8] J. K. Liu, V. K. Wei, D. S. Wong. "Linkable spontaneous anonymous group signature for ad hoc groups", *Australasian Conference on Information Security and Privacy*. Springer, Berlin, Heidelberg, pp. 325-335, 2004.

[9] S. Noether, A. Mackenzie. "Ring confidential transactions", *Ledger*, vol. 1, pp. 1-18, 2016.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

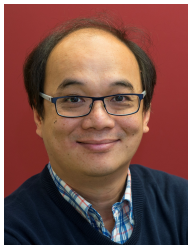
[10] P. P.Swire, "Financial privacy and the theory of high-tech government surveillance," *Wash. ULQ*, 77, 461, 1999.
[11] A. Narayanan, J. Bonneau, E. Felten, A. Miller, S. Goldfeder. "Bitcoin and cryptocurrency technologies: A comprehensive introduction", *Princeton University Press*, 2016.
[12] I. Miers, C. Garman, et al. "Zerocoin: Anonymous distributed e-cash from bitcoin", *Security and Privacy*, 2013 IEEE Symposium on. pp. 397-411, 2013.
[13] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, Madars Virza. "Zerocash: Decentralized anonymous payments from bitcoin", *Security and Privacy*, 2014 IEEE Symposium on. IEEE. pp. 459-474, 2014.
[14] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, M. Virza. "SNARKs for C: verifying program executions succinctly and in zero knowledge", *Advances in Cryptology-CRYPTO 2013*. Springer, Berlin, Heidelberg, pp. 90-108, 2013.
[15] S. F. Sun, M. H. Au, J. K. Liu, T. H. Yuen. "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero", *European Symposium on Research in Computer Security*. Springer, Cham, pp. 456-474, 2017.
[16] M. H. Au, P. P. Tsang, W. Susilo, Y. Mu. "Dynamic universal accumulators for DDH groups and their application to attribute-based anonymous credential systems", *Cryptographers Track at the RSA Conference*. Springer, Berlin, Heidelberg, pp. 295-308, 2009.
[17] M. Conti, S. Kumar, C. Lal, S. Ruj. "A survey on security and privacy issues of bitcoin", IEEE Communications Surveys & Tutorials, 2018. DOI: 10.1109/COMST.2018.2842460.
[18] Y. Li, W. Susilo, G. Yang, Y. Yu, X. Du, D. Liu, N. Guizani. "Toward Privacy and Regulation in Blockchain-based Cryptocurrencies", *IEEE Network*, DOI: 10.1109/MNET.2019.1800271
[19] M. Jakobsson, M. Yung. "Revokable and versatile electronic money", *In Proceedings of the 3rd ACM conference on Computer and communications security*, pp. 76-87, 1996.
[20] J. Camenisch, J.-M. Piveteau, M. Stadler. "An Efficient Fair Payment System", *ACM Conference on Computer and Communications Security*, pp. 88-94, 1966.
[21] J. Camenisch, U. Maurer, M. Stadler. "Digital payment systems with passive anonymity-revoking trustees". *Journal of Computer Security*, vol. 5, no. 1, pp. 69-89, 1997.
[22] D. Kulger, H. Vogt. "Off-line payment with auditable tracing", *Financial cryptography-FC' 2002*, Berlin: Springer-Verlag, pp.42-55, 2002.
[23] J. Barcelo. "User privacy in the public bitcoin blockchain". http://www. dtic. upf.edu/jbarcelo/papers/20140704_User_ Privacy_in_the_Public_Bitcoin_Blockc hain/paper. pdf, 2014.
[24] P. Koshy, D. Koshy, P. McDaniel. "An analysis of anonymity in bitcoin using p2p network traffic", *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, pp. 469-485, 2014.
[25] F. Reid, M. Harrigan, "An analysis of anonymity in the bitcoin system," *Security and Privacy in Social Networks*. Springer, New York, pp. 197-223, 2013.
[26] P. Moreno-Sanchez, M. B. Zafar, A. Kate. "Listening to whispers of ripple: Linking wallets and deanonymizing transactions in the ripple network", *Proceedings on Privacy Enhancing Technologies*, vol. 2016, no. 4, pp. 436-453, 2016.
[27] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, N. Christin. "An Empirical Analysis of Traceability in the Monero Blockchain", *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 143-163, 2018.
[28] G. Danezis, S. Meiklejohn. "Centrally banked cryptocurrencies", *arXiv preprint arXiv: 1505.06895*, 2015.
[29] E. Cecchetti, F. Zhang, Y. Ji, A. Kosba, A. Juels, E. Shi. "Solidus: Confidential distributed ledger transactions via PVORM", *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, pp. 701-717, 2017.
[30] C. Garman, M. Green and I. Miers. "Accountable privacy for decentralized anonymous payments", *International Conference on Financial Cryptography and Data Security*. Springer, Berlin, Heidelberg, pp. 81-98, 2016.
[31] J. Katz. "Digital signatures", *Springer Science and Business Media*, 2010.
[32] M. Chase and A. Lysyanskaya. "On signatures of knowledge", *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, pp. 78-96, 2006.
[33] T. P. Pedersen. "Non-interactive and information-theoretic secure verifiable secret sharing", *Annual International Cryptology Conference*. Springer, Berlin, Heidelberg, pp. 129-140, 1991.
[34] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*. vol. 3, no. 4, pp. 469-472, 1985.
[35] J. Camenisch. "Group Signature Schemes and Payment Systems Based on the Discrete Logarithm Problem", *Diss. Swiss Federal Institute of Technology Zurich*, 1998.
[36] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, B. Yang. "Assured Data Deletion with Fine-Grained Access Control for Fog-Based Industrial Applications", *IEEE Trans. on Industrial Informatics*, vol. 14, no. 10, pp. 4538-4547, 2018.
[37] A. Fiat, A. Shamir. "How to prove yourself: Practical solutions to identification and signature problems," *Conference on the Theory and Application of Cryptographic Techniques*. Springer, Berlin, Heidelberg, pp. 186-194, 1986.
[38] B. B. Brumley, "Efficient three-term simultaneous elliptic scalar multiplication with applications", *Proceedings of the 11th Nordic Workshop on Secure IT Systems*. vol. 6, pp. 105-116, 2006.
[39] Y. Yu, Y. Ding, Y. Zhao, Y. Li, X. Du, M. Guizani. "LRCoin: Leakage-resilient Cryptocurrency Based on Bitcoin for Data Trading in IoT", *IEEE Internet of Things Journal*, DOI 10.1109/JIOT.2018.2878406, https://ieeexplore.ieee.org/document/8513813.

Yannan Li is currently a Ph.D. candidate of School of Computing and Information Technology, University of Wollongong, Australia. Her research interests are applied cryptography and cryptocurrencies.



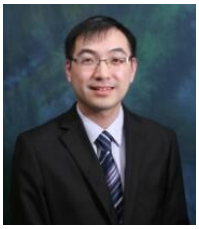
Guomin Yang is currently a Senior Lecturer and a DECRA Fellow with University of Wollongong, Australia. He received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2009. His current research interests include digital signature and blockchain.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2019.2910058, IEEE Transactions on Dependable and Secure Computing

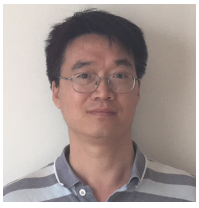


Willy Susilo received a Ph.D. degree in Computer Science from University of Wollongong, Australia. He is a Professor and the Head of School of Computing and Information Technology and the director of Institute of Cybersecurity and Cryptology (iC^2) at the University of Wollongong. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council (ARC). His main research interests is applied cryptography. He is a senior member of IEEE.

Yong Yu is currently a Professor of Shaanxi Normal University, China. He holds the prestigious one hundred talent Professorship of Shaanxi Province as well. He received his Ph.D. degree in cryptography from Xidian University in 2008. His research interests are blockchain and cloud security. He is an Associate Editor of Soft Computing.



Man Ho Au received the Ph.D. degree from the University of Wollongong, Australia, in 2009. He is currently an Assistant Professor with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include public key cryptography and cryptocurrency..



Dongxi Liu is a Senior Research Scientist in CSIRO since 2008. His research interests are applied cryptography, data processing and system security.