Spring March 2, 2017

SecureCoin: A Robust Secure and Effi ient Protocol for Anonymous Bitcoin Ecosystem

Maged H Ibrahim, *Helwan University*

SecureCoin: A Robust Secure and Efficient Protocol for Anonymous Bitcoin Ecosystem

Maged Hamada Ibrahim

(Corresponding author: Maged Hamada Ibrahim)

Department of Electronics, Communications and Computers, Faculty of Engineering, Helwan University
No. 1, Sherif St., Helwan, P.O. 11792, Cairo, Egypt
(Email: mhii72@gmail.com, maged_ismail@h-eng.helwan.edu.eg)

(Received Jan. 19, 2016; revised and accepted Apr. 7 & Apr. 25, 2016)

Abstract

Bitcoin is the first decentralized peer-to-peer electronic virtual asset and payment cryptocurrency, through which, users can transact digital currency directly, without the need for an intermediary (or authority), using a hashed version of cryptographic public keys, as pseudonyms called addresses. The Bitcoin ecosystem was supposed to be anonymous and untraceable. However, transactions from input to output addresses of the Bitcoin users are observed to be linkable, therefore, missing unlinkability as an important requirement of anonymity. Several protocols appeared to enhance Bitcoin users' anonymity and to ensure unlinkability of input-output addresses, to make input and output addresses of transactions unlinkable to each other, and hence untraceable. In this paper, we spot several vulnerabilities in the most recently proposed protocols, then we propose SecureCoin as an efficient protocol for anonymous and unlinkable Bitcoin transactions that covers these vulnerabilities in a robust and secure way and in full compatibility with the standard Bitcoin ecosystem. Our protocol provides better protection for the participating peers against malicious behavior of minority of the peers and protection against the most serious sabotage attack attempted by any number of saboteur peers. We analyze the security properties of our protocol and evaluate its efficiency. Finally, we compare the performance of our protocol with the recently proposed protocols and show that our protocol is computationally efficient and requires less Bitcoin fees.

Keywords: Anonymity, bitcoin, cryptocurrency, ECDSA, oblivious shuffling, silk road, unlinkability

1 Introduction

Bitcoin is a decentralized electronic currency protocol, that realizes worldwide of fast peer-to-peer transactions, and consequently payments with low, or near zero, transaction processing charges. In order to avoid the dependency on a central trusted authority charged with the issuance and control of currency, Bitcoin functions through a peer-to-peer topology. In this ecosystem, it is not possible to manipulate the value of the digital coins (bitcoins) or produce inflation through the overproduction of currency. Transactions and the creation of bitcoins are managed by the network itself; the creation of digital coins in a controlled and decentralized manner through the process known as mining. Cryptographic primitives guarantee the security of transactions. Coins can only be spent by the owner of their pseudonyms, and they can only be used in a single transaction with no chance of duplication. The supervision and management institutions that operate in traditional trusted centralized systems do not exist for Bitcoin [2].

Of the participants in the Bitcoin system, two – not necessarily mutually exclusive – groups can be distinguished:

Normal users. Users of the Bitcoin system who buy or sell for goods and services with coins, producing transactions in the system.

Miners. Special users who dedicate computing power to verify new transactions, creating what are known as transaction blocks. The calculations required to do this are expensive in computing power, which is why these users are rewarded.

1.1 Bitcoin Components and Processes

Bitcoin addresses. This is a user's digital address, also called pseudonym, which contains coins and which is used to make and receive transactions, similar to a bank account. A given user may have as many addresses as he wishes, and they are identified by a public key. Bitcoin uses the ECDSA (Elliptic Curve Digital Signature Algorithm) to sign its transactions, using parameters recommended by the Standards for Efficient Cryptography Group (SECG), secp256k1 [5]. The signatures use DER encoding.

ECDSA offers many advantages over other digital signature systems (e.g. RSA and DSS) which make it ideal to be used for a distributed Internet protocol. ECDSA provides relatively short keys and signature lengths, and a faster generation and verification. On being identified by the ECDSA public key, all the operations carried out with this address have to be supported by the use of the corresponding private key. The holder of the private key is the owner of the coins associated with the corresponding address.

Wallets. Personal virtual storage, similar to a physical/pocket wallet, where users' coin addresses and the payments made with them are stored and managed. Wallets are thus a grouping pairs of public and private keys and are used to carry out other tasks, for example, preparing transactions.

Transactions. A transaction is the transfer of coins from Bitcoin input address I to another output/destination address D. To create a transaction, the owner of address I signs a transcription for address D (amongst other data) with the private key associated with address I, so that the whole network knows that the new legitimate owner is the owner of address D. There are three main types of Bitcoin transaction forms [2] that will be used in our SecureCoin protocol:

- One-to-one form. A commonly used form of transaction is a simple payment from one address to another, which often includes some change returned to the original owner. This type of transaction has one input and one output if no change is returned, or in general, one input and two outputs.
- Aggregation form. Another common form of transaction is a transaction that aggregates several inputs into a single output. This represents the real-world equivalent of exchanging a pile of coins and currency notes for a single larger note. Transactions like these are frequently generated by wallet applications to clean up lots of smaller amounts that were received as change for payments, by transferring them to a new fresh address owned by the same user.
- Distribution form. Another transaction form that is seen frequently on the Bitcoin ledger is a transaction that distributes one input to multiple outputs representing multiple recipients. This type of transaction is sometimes used by commercial entities to distribute funds, such as when processing payroll payments to multiple employees.

Blocks. This is a structure that aggregates transactions. Transactions whose confirmation is pending are aggregated together as a block in a process which is known as mining.

Blockchain (Ledger). This is the public record of verified Bitcoin transactions in chronological order. When a block has been checked and confirmed, through mining, it is included as part of the chain.

Cryptographic hashes. In the hash calculations carried out in Bitcoin, the SHA-256 standards are used, and when shorter hashes are required, RIPEMD-160. Normally the hash calculations are carried out in two phases: the first with SHA-256, and the second, depending on the length desired, with SHA-256 or RIPEMD-160.

Random numbers and nonces. In Bitcoin, random numbers and nonces are used directly in the formation and generation of blocks. To make a new block, a random number that satisfies certain requirements needs to be found. Random numbers are also indirectly used in Bitcoin as part of the digital signature algorithm (ECDSA).

Proofs-of-work. Proofs-of-work are the main component guaranteeing the legitimate behavior of Bitcoin network. The idea "in brief" is that, verifying/calculating new transaction blocks ensures a high computational cost, such that to control the network and to regulate the rate at which new coins are generated. This control of complexity in the calculation of new blocks is carried out by requiring that the hash for each new block starts with a given number of zeros. This required number of zeros controls the difficulty and hence the rate at which new coins are generated. Older block data and a nonce are combined to calculate this hash. Given that cryptographic hash functions are not invertible, in order to find a new valid block the only alternative would be to obtain different nonces until one which fulfills the pre-established requirement is found.

1.2 The Linkability/Traceability Problem

As the authors in [36] pointed out, it is possible to discover the identity of someone who makes a transaction in Bitcoin through traffic analysis and tracing of users IP's. Owing to Bitcoin's design, the first person to publicize a transfer will probably be the payer. Therefore, discovering who the first person to publicize it was, will permit in all probability to know who the payer in the transaction and who the owner of the input addresses used are.

Another type of analysis, which stands out, is that based on the relations which may be established between addresses. Various studies have developed heuristics to reduce the degree of anonymity of Bitcoin users [1, 30]. For example, the authors in [1] estimate that approximately 40 percent of Bitcoin users could be identified using the heuristics set out in the study. A striking result, obtained by applying this type of measure, is that published in [30], in which a relation between the founder of Silk Road and

someone who was probably one of the creators of Bitcoin was established. The tracing capability shown by the researchers remains completely valid.

The above mentioned traceability problems are due to the fact that the input and output addresses of the Bitcoin users are linkable by one way or another. Therefore, protocols to enhance Bitcoin users' anonymity, ensuring unlinkability of input-output addresses, must be provided. Based on what are known as mixing services [7], recently, some efforts have been made towards overcoming the above attacks and providing stronger privacy to the Bitcoin users by mixing multiple transactions to make input and output addresses of transactions unlinkable to each other. In this direction, some third-party Bitcoin mixing services were first to emerge, but they have been prone to thefts.

2 Previous Work

Among the early proposals, to provide unlinkability between individual Bitcoin transactions, without introducing a trusted party, is the ZeroCoin scheme of [31], which is an extension to Bitcoin. It employs a cryptographic accumulator of minted zero-coins and a zero-knowledge proof of inclusion of a certain zero-coin within the accumulator. ZeroCoin suffers from significant computation and communication overheads where, the size of the proof that has to be stored in the blockchain for each transaction is prohibitively large and far exceeds the size of the Bitcoin transaction itself. The scheme is considered incompatible with the standard Bitcoin ecosystem. Later, several contributions were aiming to reduce the ZeroCoin overheads [12, 17, 38]. However, all these schemes require a modification to the Bitcoin standards and hence, they are incompatible with Bitcoin and are unlikely to be accepted for implementation as mentioned in [40].

Anonymous Bitcoin payments are facilitated in the MixCoin scheme [6]. This scheme does not make any modifications to the standard Bitcoin protocol. Central accountable mixing server is employed, where Bitcoin users send their coins. The center in turn, replies with a guarantee of returning the funds to the user. This strategy ensures unlinkability between the user's input and output addresses, since, the mix sends the coins back to the user. However, using this strategy, the decentralized property (the main property of Bitcoin) is eliminated. Unlinkability is only guaranteed against external observers, because the mixing server learns which address belongs to which user.

A modification to the CoinJoin protocol using blind signatures in the work of [29] avoids the problem of a centralized mix learning the relation between input and output addresses. The anonymous communication network, "Tor" [13] is employed in [29] to avoid centralized mixing and to provide unlinkability.

The CoinShuffle protocol in [37], employs the group transactions service in Bitcoin to ensure correctness and thus shares almost the same disadvantages with CoinJoin, i.e., limited anonymity levels and potential transaction fees. CoinShuffle improves over CoinJoin by using decryption mixnets for address shuffling which achieves anonymity against insiders. However, in CoinShuffle the last peer is in the unique position to determine the outcome of the shuffling and might exploit this to select preferred input addresses to her own output addresses. Our scheme – although employs a shuffling subroutine – does not suffer this disadvantage, since all the coins are transfered and committed to an aggregation temporary address before the result of the shuffling is announced.

2.1 Recent Protocols

The most recent protocol (until the time this paper was written) is CoinParty [40], which invokes the threshold ECDSA digital signature protocol of [25] and employs mixing peers to provide unlinkability for the Bitcoin ecosystem. They introduced an elegant idea of committing each input peer to a temporary Bitcoin address where the signature key of this address is jointly shared among the mixing peers on a threshold bases. Each input peer P_i must first transfer the coins x from his input address I_i to a temporary address T_i, controlled by the mixing peers. This strategy commits the input peers to the transactions. Next, after all transactions are confirmed, the mixing peers are responsible for mixing (shuffling) the output addresses and broadcast the shuffled version. Finally, the peers join to sign a transaction from every temporary address T_i to every shuffled output address $O_{\pi(i)}$. The work in this paper is inspired by the CoinParty and the CoinShuffle protocols.

2.2 Existing Vulnerabilities

The work in this paper is motivated by the security vulnerabilities and efficiency drawbacks of recent protocols described next.

CoinParty. Considering CoinParty protocol [40], the employed mixing peers are different from the input peers, therefore, a major drawback is that, once the input peers transfer their coins to the temporary addresses owned by the mixing peers, saboteur mixing peers may runaway (due to halting or disconnection) leaving the coins of the input peers stuck in the temporary addresses with absolutely no way to return them back. The consequences are disastrous if the number of disconnected mixing peers exceeds a certain threshold required to jointly sign a transaction. Moreover, they can easily conspire to steal the coins. In CoinParty, the mixing peers have no financial shares (coins) in the mixing protocol, therefore, they will not loose anything if they attempt such sabotage attack. In other words, they do not care if the input peers loose their coins. Notice that the signing keys of the temporary addresses are shared among the mixing peers on threshold basis. Hence,

if enough number of mixing peers leave, the signing keys are lost. Moreover, they most probably will not tend to participate without incentives (return fees), similar to the Bitcoin miners in a standard Bitcoin ecosystem, which increases the fees paid by the input peers. A serious efficiency drawback in CoinParty is that, the protocol requires the mixing peers to invoke the threshold ECDSA key generation protocol and the threshold ECDSA digital signature protocol for every temporary address. Invoking the threshold ECDSA for every temporary address results in a relatively high computation complexity for each mixing peer, specially, if these protocols are supposed to run on mobile devices. Moreover, the threshold ECDSA they invoked is in the honest-but-curious scenario, giving a great chance for undetected malicious behavior by the peers due to a corruptive adversary. The authors stated that the protocol must be implemented to withstand malicious behavior.

CoinShuffle. CoinShuffle protocol [37] employs the idea of multiparty private shuffle [7, 10, 15, 32]. The problems with the CoinShuffle protocol are as follows: In order to prevent malicious peers from aborting the protocol after they have received their funds thereby leaving another peer unpaid, the protocol performs a single atomic bulk transaction from all input addresses to all shuffled output addresses at once. Although this strategy is better than CoinParty to protect against sabotage attacks, the resulting anonymity set of the mixing is limited to the number of users participating in a particular mixing operation. Also, this bulk transaction makes the peers easily identifiable on the Bitcoin ledger, since it is very rare and almost unused for other purposes. The transaction itself is of large size and hence, dramatically increases the return fees to the Bitcoin miners. Another problem is that, at the end of the shuffling protocol, one peer knows all the output addresses in the clear, including his own, and this knowledge is before the shuffled addresses are known to other peers. This gives a great chance to this peer to reorder the output addresses in a certain way that he can benefit from.

3 Our Contribution

We propose SecureCoin, as a protocol fully compatible with the Bitcoin ecosystem, to realize anonymity and unlinkability security services to Bitcoin peers. Unlike CoinParty, our protocol does not involve any separate mixing peers as helpers and hence, it is possible to avoid sabotage attacks that could be attempted by mixing peers. Only the input peers are the participants in the protocol. Our protocol uses the inherited "aggregation" and "distribution" transactions forms of the Bitcoin and hence, improves complexity of running multiple instants of a threshold digital signature protocol by reducing to only one instant.

SecureCoin protocol improves security and efficiency over the CoinShuffle protocol, such that, it avoides bulk transaction from multiple input addresses to multiple output addresses. Also, it solves the CoinSuffle problem of allowing a particular peer to control mapping of certain input addresses to certain output addresses. The proposed SecureCoin protocol provides protection against sabotage attacks, attempted by any number of participating saboteurs.

The proposed SecureCoin is robust against malicious behavior of minority (one third) of the participating peers. It does not allow any number of participating saboteurs to maliciously behave without either, being detected and revoked, or loosing their input coins. The behavior of the participating peers is indistinguishable from other normal transactions and hence, cannot be distinguished by miners and ledger observers.

We analize the security properties of SecureCoin, evaluate its performance and compare to recently proposed protocols. We show that, SecureCoin is more efficient and requires less fees by the Bitcoin ecosystem.

4 System Model

In this section, we describe the assumptions, model and goals of the protocol presented in this paper.

4.1 Assumptions and Model

We assume the existence of n peers, $P_1, \cdots, P_n$, where $n > 3t$ and $t \geq 1$ is a particular predefined threshold. There is at most t possible malicious peers. We assume a static adversary model, yet, security against adaptive adversary could be achieved by employing a suitable non-committing encryption scheme. Also, security against coercive adversaries could be realized by employing a suitable deniable public key encryption scheme (e.g. [22, 23]).

Each peer has his own public/private key pair. These keys allow the peers to realize authenticated and private channels among themselves. We emphasize that, these keys are different from the ephemeral public/private keys used in the shuffling stage of our protocol. Given a certain threshold t, we assume that there is at least $n = 3t + 1$ peers that survive till the end of the protocol with at most t of them may behave maliciously [11, 25]. We use n to refer to the number of peers remaining after the execution of any phase in our protocol. In our protocol, there are four different classes of public/private key pairs for a public key cryptosystem, each peer holds:

1) The personal public/private key pair to realize private and authenticated channels with other peers.

2) The ephemeral public/private key pair for the oblivious shuffling stage.

3) The partial signature key corresponding to the aggregation address A.

4) The key pairs for his bitcoin pseudonyms (input and destination addresses).

4.2 Problem and Goals

We solve the problem where n peers, $P_1, \cdots, P_n$ (each peer P_i has a certain amount of x coins available at input address I_i) want to transfer that amount from the set of input addresses $I_1, \cdots, I_n$, to a set of destination addresses, $D_1, \cdots, D_n$, such that,

Correctness/safety. Each peer P_i receives back x coins on his destination address D_i. Coins must not be lost, stolen, or double-spent by any peer even in the presence of a malicious adversary. Honest peers should receive their funds in a timely manner.

Anonymity. Input and destination addresses are unlinkable, i.e., only peer P_i can map his own input address I_i to his own destination address D_i.

Protection against saboteurs. Participating peers may be accidentally or deliberately halted (or disconnected) from the network at any stage of the protocol. We argue that, if such sabotage occurs, then all peers must be equally affected. This type of attack may occur easily due to a halting adversary and is easier than corruptive attacks that requires collaboration of peers in harmony in order –for example– to steal other peers coins. In sabotage attacks, we have two situations: The number of saboteurs do not exceed the threshold t. In this case according to secure multiparty computations (SMC) settings, the protocol execution continue normally. The other situation is when the threshold is exceeded and the disconnected peers do not reconnect. This case is really disastrous. As the threshold is exceeded, the SMC protocol will be terminated. Some peers will loose their coins for ever while others may abort without any loss. Our objective in this case is to ensure that, disconnected peers, at any phase of the protocol, will suffer the same amount of loss as the other participating peers. This motivates disconnected peers to rejoin the protocol execution to save their coins. CoinShuffle effectively withstands saboteurs, since it is not based on threshold cryptosystems and the shuffled transfer of funds is made in a one n-to-n bulk transaction. On the contrary, CoinParty is vulnerable to this attack, as stated earlier, due to the incorporation of mixing peers, that have no financial share in the protocol, other than mixing fees as incentives. We conclude that a protocol must be designed such that, any attempt of sabotage attack by any number of participating saboteur peers at any stage of the protocol must cost them the same amount of lose as other participating peers.

Robustness & Revocation. Any malicious behavior attempted by any peer must be detected as early as possible, this malicious peer must be identified and kicked out (revoked) from the protocol, without affecting the good peers progress in the protocol.

Indistinguishability. The transactions performed by the peers incorporated in the anonymous protocol, must not be distinguished from normal Bitcoin transactions, performed by individual peers.

Unforgeability. An adversary must not be able to create a pair consisting of a message (Bitcoin transaction), m and a signature that is valid for m, that has not been generated by a legitimate signer and that passes the verification of the Bitcoin ecosystem.

Fairness. A participating peer that deposit his coins in the aggregation/temporary address must receive his coins, either in his destination address, or returned back to his input address in case he was kicked out of the protocol due to a malicious behavior.

Performance. The protocol should scale to large numbers of peers without imposing prohibitive overheads upon the Bitcoin network.

Compatibility. The mixing protocol must be fully compatible with the current Bitcoin network and produce legitimate Bitcoin transactions.

Cost efficiency. The protocol must be cost-efficient in terms of involved transaction fees.

5 Preliminaries and Basic Tools

In this section, we give an overview of a bunch of basic tools, necessary to build our SecureCoin protocol. These tools are partitioned into two categories: threshold cryptography tools and shuffling tools. The reader needs to be familiar with these tools in order to follow the description of our SecureCoin scheme. First we give an overview on the elliptic curve used in the Bitcoin ecosystem and the standard ECDSA algorithm, then we describe the cryptographic tools employed by SecureCoin.

5.1 Elliptic Curves for the Bitcoin

Standardized elliptic curves that are used most commonly in real-world applications are mostly given in their short Weierstrass form, $E : y^2 = x^3 + ax + b$ and are defined over a finite field $\mathbb{F}_p$, where $p > 3$ is a prime and $a, b \in \mathbb{F}_p$. For non-singular curves, there is a requirement that, $4a^3 + 27b^2$, is non-zero. Given such a curve E, the cryptographic group that is employed in protocols is a large prime-order subgroup of the group $E(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points on E. The group of rational points consists of all solutions $(x, y) \in \mathbb{F}_p^2$ to the curve equation, together with a point at infinity, denoted by $\mathcal{O}$, the neutral element. The number of $\mathbb{F}_p$-rational points is denoted

by $\#E(\mathbb{F}_p)$ and the prime order of the subgroup by q. A fixed generator of the cyclic subgroup is usually called the base point and denoted by $G \in E(\mathbb{F}_p)$.

For 256-bit primes, in addition to the NIST curve defined over $\mathbb{F}_{p^{256}}$, SEC2 also proposes a curve named secp256k1 defined over $\mathbb{F}_p$ where $p = 2^{256} - 2^{32} - 977$. This curve is the one used in Bitcoin [5].

There are three basic point operations on elliptic curves.

1) Point addition: Let $P_1 \in E(\mathbb{F}_p)$ and $P_2 \in E(\mathbb{F}_p)$, then, $P_3 = P_1 + P_2 \mod p$ is also a point on the elliptic curve, that is, $P_3 \in E(\mathbb{F}_p)$.

2) Point doubling: Let $P \in E(\mathbb{F}_p)$, then $Q = 2P \mod p$ is a point doubling operation where $Q \in E(\mathbb{F}_p)$.

3) Scalar multiplication: Let $k \in Z_q$ and $P \in E(\mathbb{F}_p)$, then, $Q = kP \mod p$ is the process of adding P to itself k times, where $Q \in E(\mathbb{F}_p)$. This process is performed through the well-known double-and-add operations.

5.2 Elliptic Curve Digital Signature

The Elliptic Curve Digital Signature Algorithm (ECDSA) was standardized in FIPS 186-4[1]. The signer generates a key pair (s, Q) consisting of a private signing key $s \in_R Z_q^*$ and a public verification key, $Q = sG \mod p$, where G is the generator point. To sign m, the signer chooses a per-message random integer $k \in_R Z_q^*$, computes the point $(x, y) = kG$, and computes $r = x \mod q$. The signature of a message M, is the pair (r, w), of integers modulo q, where $w = k^{-1}(m + sr) \mod q$ and m is the hash of M.

It is important that the per-message secret k is not revealed, since otherwise the secret signing key s can be computed by $s = r^{-1}(kw - m) \mod q$, because r and w are given in the signature and m can be computed from M. Even if only several consecutive bits of the per-message secrets for a certain number of signatures are known, it is possible to compute the private key (see [26]). Also, if the same value for k is used to sign two different messages M_1 and M_2 using the same signing key s and producing signatures (r, w_1) and (r, w_2), then k can be easily computed as $k = (w_1 - w_2)^{-1}(m_1 - m_2) \mod q$, which then allows recovery of the secret key.

5.3 Threshold Cryptography Tools

In this subsection, we review the threshold cryptography tools that will be used in building our SecureCoin. All theses tools are based on the well-known polynomial/Shamir's secret sharing scheme. We describe the elliptic curve version of these tools, in order to be directly applied for the implementation of SecureCoin.

[1]PUB FIPS. 186–2. digital signature standard (dss). US department of commerce/national institute of standards and technology.

5.3.1 Polynomial Secret Sharing

Consider a secret value, $s \in Z_q$ which is held by a dealer, where Z_q is a prime field. To share this secret among a set $\mathcal{P} = \{P_1, \cdots, P_n\}$ of $n > t$ participants, where t is a certain threshold, the dealer constructs a polynomial $g(x) = \sum_{j=0}^{t} a_j x^j \mod q$, he sets $a_0 = s$ and each other coefficient $a_{j\neq 0} \in_R Z_q$. $\forall i = 1, \cdots, n$, the dealer secretly delivers $g(i)$ to participant P_i. To reconstruct the secret s, each participant $P_i \in \mathcal{P}$ broadcasts $g(i)$, the participants compute s from any $t + 1$ shares using Lagrange interpolation formula, $s = g(0) = \sum_{i\in\mathcal{B}} \lambda_i g(i) \mod q$ where $\mathcal{B} \subset \mathcal{P}$, $|\mathcal{B}| = t + 1$ and, $\lambda_i = \Pi_{j\in\mathcal{B}, j\neq i} \frac{j}{j-i}$, is participant P_i's Lagrange coefficient.

5.3.2 Elliptic Curve Verifiable Secret Sharing

Verifiable secret sharing (VSS) is an extension to polynomial/Shamirs secret sharing to allow the recipients of the secret shares to verify that the shares are consistent (i.e., that any subset of $t + 1$ shares interpolate to the same unique secret). Assuming $n \geq 2t + 1$, the scheme tolerates the malicious behavior of at most t of the n participants. Two different types of VSS are distinguished; the conditionally secure scheme due to Feldman [14] and the unconditionally secure scheme due to Pedersen [34]. For best security, both of them will be used in our SecureCoin protocol. We present an overview of these subroutines over elliptic curves.

EC Feldman-VSS. Let p and q be two large primes, such that $q|p - 1$. The two primes p and q and the base point (EC generator point) G of order q are published as the system public parameters. The dealer shares the secret s among the participants on a t-degree polynomial $g(x) = \sum_{j=0}^{t} a_j x^j \mod q$, the dealer also broadcasts the $t + 1$ commitments $C_j = a_j G \mod p \; \forall j = 0, \cdots, t$. These commitments allow each participant P_i to verify the consistency of his share $g(i)$ by checking that, $g(i)G = \sum_{j=0}^{t} i^j C_j \mod p$. If this check fails for any share $g(i)$, P_i broadcasts a complaint. If more than t participants broadcasted a complaint, then at least one of them is honest, therefore, the dealer is deemed corrupt and disqualified. Otherwise, the dealer broadcasts the share $g(i)$ of each complaining participant P_i, if the share is consistent, P_i is disqualified, otherwise, if the share is inconsistent with the commitments or the dealer does not respond, then the dealer is disqualified. In the reconstruction phase of the secret, all the participants are able to check the validity of the share broadcasted by any of the other participants by verifying with the published commitments to filter out inconsistent shares and safely perform the Lagrange interpolation. When it comes to the distributed generation of a secret key s and the joint computation of $sG \mod p$, Feldman-VSS alone is not secure due to the attacks attempted in [19, 24].

EC Pedersen-VSS. The trick is to perform double exponentiation to allow randomization of the broadcasted commitments. The public parameters in this VSS are p, q and G as in Feldman-VSS and another generator point H, subject to the condition that $\log_G H$ is unknown and assumed hard to compute. In addition to the polynomial $g(x) = \sum_{j=0}^{t} a_j x^j \mod q$ with the secret s as the free term, the dealer constructs another polynomial as a randomization t-degree polynomial $r(x) = \sum_{j=0}^{t} b_j x^j \mod q$. He secretly delivers $(g(i), r(i))$ to participant P_i $\forall i = 1, ..., n$. The dealer also publishes the commitments $C_j = a_j G + b_j H \mod p \; \forall j = 0, \cdots, t$. Each participant P_i is able to verify the consistency of his share $g(i)$ by checking that, $g(i)G + r(i)H = \sum_{j=0}^{t} i^j C_j \mod p$. If this check fails for any share $g(i)$, P_i broadcasts a complaint. If more than t participants broadcast a complaint, then at least one of these participants is honest about his complaint and the dealer is disqualified. Otherwise the dealer broadcasts the pair $(g(i), r(i))$ for each complaining participant P_i, if the pair is consistent, P_i is disqualified, otherwise, if the pair is inconsistent with the commitments or if the dealer does not respond, then the dealer is disqualified. During reconstruction, any participant can verify the validity of the share broadcasted by any other participant via the published commitments to reject invalid shares and correctly computes the interpolation.

5.3.3 Joint Secret Sharing

Joint secret sharing are schemes to allow the participants to jointly share a secret among themselves in the absence of a dealer.

Joint random secret sharing (JR-SS). JR-SS [26] allows a set of n participants to jointly share a random secret among themselves without the assistance of a dealer. Each participant $P_i \in \mathcal{P}$ chooses a random integer $k_i \in Z_q$ and plays the dealer's role to share k_i among the participants over a t-degree polynomial $g_i(x) = k_i + \sum_{j=1}^{t} a_j x^j \mod q$. Each participant $P_i \in \mathcal{P}$ simply sums the shares he receives from the other participants to compute a share $g(i) = \sum_{j=1}^{n} g_j(i)$ which is a point on a t-degree polynomial $g(x)$ with its free term equals a random secret $k = \sum_{i=1}^{n} k_i \mod q$.

Joint random verifiable secret sharing (JR-VSS). To withstand malicious behavior of at most $t < n/2$ participants during the JR-SS, JR-VSS combines JR-SS with Feldman or Pedersen-VSS. In this scheme, each participant $P_i \in \mathcal{P}$ chooses a random secret integer $k_i \in Z_q$ and plays the dealer's role in the VSS protocol to share this secret among the other participants. Complaints are solved as in the VSS scheme. Finally, each participant sums what he has to compute his share on a t-degree polynomial, $g(x)$ with its free term $g(0) = \sum_{i=1}^{n} k_i \mod q$.

Joint zero secret sharing (JZ-SS). This scheme is a special case of the JR-SS. As implied by the name of the scheme, the random secret shared by each participant is a zero. After execution of a JZ-SS scheme, each participant holds a share $g(i)$ on a t-degree polynomial $g(x)$ with its free term $g(0)$ equals zero. This scheme is always employed when we need to randomize the shares, without changing the value of the secret.

Joint zero verifiable secret sharing (JZ-VSS). Similar to JR-VSS, to with stand malicious behavior of at most $t < n/2$ participants in the JZ-SS, the JZ-VSS combines the JZ-SS with Feldman- VSS or Pedersen-VSS. In this scheme, each participant $P_i \in \mathcal{P}$ plays the dealer's role in the VSS protocol to share a zero among the other participants. Complaints are solved as in the JR-VSS protocol. The shares are computed by each participant sums what he has to compute his share on a t-degree polynomial, $g(x)$ with its free term equals zero.

5.3.4 Joint Verifiable Multiplication (JVM) of Shared Secrets

Consider two secret values a and b, respectively shared over t-degree polynomials $A(x)$ and $B(x)$, the joint verifiable multiplication subroutine [4] computes $\mu = ab \mod q$ in a robust and secure way with no information revealed about neither a nor b. Each participant P_i locally computes $C(i) = A(i)B(i) \mod q$ which is a share on a $2t$-degree polynomial $C(x) = A(x)B(x) \mod q$ with $C(0) = \mu$. There is still a security problem; publishing and interpolating the shares $C(1), \cdots, C(n)$ reveals information about $A(x)$ and $B(x)$, therefore, it is necessary to randomize the shares of $C(x)$. To randomize the shares without changing the secret value $C(0)$, the participants run JZ-VSS to share a zero over a $2t$-degree polynomial $R(x)$ with $R(0) = 0$. Each participant P_i finally computes and broadcasts $D(i) = C(i) + R(i)$. The result μ could be computed by interpolating the $2t$-degree polynomial $D(x)$ using the Berlekamp-Welch decoder[2] to filter out bad shares. Since we are interpolating a polynomial of degree $deg = 2t$ and we have a maximum of t malicious participants (i.e.there are at most t possible faults), the Berlekamp-Welch bound implies that, the number of shares needed in order to correctly interpolate the polynomial is at least $deg + 2faults + 1 = 4t + 1$. Consequently, we need $n > 4t$. This bound on n could be reduced to $3t + 1$ if the participants run a polynomial degree reduction subroutine just before applying the Berlekamp-Welch decoding.

[2] L. R. Welch and E. R. Berlekamp. Error correction for algebraic block codes. Google Patents, December 30 1986. US Patent 4,633,470.

5.3.5 Joint Verifiable Reciprocal (JVR) of a Shared Secret

In our SecureCoin, we are faced with the following problem. Given some secret value k, shared among the participants, compute shares of the reciprocal of $k \mod q$ while k is kept secret. Each participant P_i already holds a share $g(i)$ representing a point on a t-degree polynomial $g(x)$ with $g(0) = k$. To compute shares of $k^{-1} \mod q$, we need $n > 4t$ participants to run the reciprocal protocol [3] (or $n > 3t$ in case polynomial degree reduction is employed) as follows: (i) The participants run the JR-VSS, which results in each participant holds a share $v(i)$ of a random secret v over some polynomial of degree t. (ii) The participants run the JVM subroutine and reconstruct $\mu = kv \mod q$, with no information revealed about k or v. (iii) Each participant P_i computes his share of the reciprocal as $\mu^{-1}v(i) \mod q$, which is a share over a t-degree polynomial with its free term equals $k^{-1} \mod q$.

5.4 Mixnets to Multiparty Secret Shuffle

Since the introduction of Chaums mixing network, mixnet [9] and dinning cryptographers problem, DCnet [8] thirty years ago, a number of anonymous authentication protocols have been developed. The mixnet family schemes use a set of mix servers that mix the received messages to make the communication paths ambiguous. The security of mixnet is based on the trust relationship of the mixers, and cannot provide unconditional anonymity.

In [35], inspection of Chaum's scheme [9] showed that the scheme is linkable. In Chaums scheme, the encryption function was assumed to be a one-way trapdoor permutation, such as the textbook version of RSA scheme. As a result, anyone can take an output message, encrypt it again and check with the input messages they obtain. In this way, the mix can be reversed. To prevent this re-encryption and possible size matching of the incoming flow and output flow, all messages are resized through random string padding to be of the same size. The output messages from the mix will be indistinguishable to adversaries, and therefore we can prevent traffic analysis of network transmissions. This will also ensure that no item is processed more than once. By discarding the repeated input messages, replay attacks can also be prevented. Otherwise, an attacker can repeat the input message and observe which output message is repeated. In this way, the relation of input messages and output messages can be discovered and the claimed anonymity is lost.

Several mixnets have also been designed based on zero-knowledge proofs and stronger security assumptions to guarantee delivery or to detect and exclude misbehaving participants. These schemes include flash mixes [27], hybrid mixes [28, 33], and provable shuffles [7, 15, 32]. The Dissent scheme [10] for anonymous messaging allows a group of participants to communicate messages in a private and anonymous way through verifiable secret shuffling.

6 Our SecureCoin Protocol

In this section, we describe in details our SecureCoin Protocol. Our protocol runs in three stages, each consists of few phases:

Stage 1. Aggregated temporary deposit.

- **Phase 1.1.** Distributed generation of A.
- **Phase 1.2.** Joint deposit of the coins.

Stage 2. Shuffling of destination addresses.

- **Phase 2.1.** Destination addresses generation.
- **Phase 2.2.** Oblivious shuffling.
- **Phase 2.3.** Accusations resolution (if exist).

Stage 3. Coins distribution.

6.1 Aggregated Temporary Deposit

The goal of this stage is to allow the peers to aggregate and deposit the required coins in a temporary aggregation address (ECDSA public-key), A, as a form of commitment. The aggregated coins must not be spent by any individual peer that could behave maliciously to steal the aggregated coins. Hence, the private key corresponding to this address A must be protected against minority of malicious peers. To achieve this, the corresponding private key is generated on a verifiable threshold bases and the public-key A is jointly computed such that, none of the peers has any information about the private key. Yet, still a transaction from this address A can be performed through threshold computation of the signature.

6.1.1 Distributed Generation of A

The peers jointly generate the aggregation address, A, as follows:

Step 1. The peers execute JR-VSS with Pedersen-VSS commitments. The execution results in each peer P_i holds a share $S(i)$ of a secret $s \in_R Z_q^*$ over a t-degree polynomial $S(x)$ with $S(0) = s$.

Step 2. The peers that are not disqualified in the JR-VSS in the previous step publish Feldman-VSS commitments to their shared polynomial. I.e., if $S_i(x) = s_i + \sum_{j=1}^{t} a_j x^j$ is the polynomial of peer P_i, P_i publishes $s_i G$ and $a_j G \mod p \ \forall j = 1, \cdots, t$.

Step 3. For any peer P_i who receives at least one valid complaint, the other peers join together to reconstruct his polynomial $S_i(x)$ and the values $s_i G$ and $a_j G \mod p \ \forall j = 1, \cdots, t$ in the clear.

Step 4. Finally, the remaining good peers join to safely compute $A = sG = \sum_{i=1}^{n} s_i G \mod p$.

At this point, each peer P_i holds a share $S(i)$ of an ECDSA private-key s over a polynomial of degree t and have jointly computed the temporary aggregation address $A = sG \mod p$. The disqualified peers in the above subroutine are kicked out and prevented from further participating in the rest of the protocol. In Bitcoin, the peer's address is actually a hashed version of the public key. However, for simplicity and wlog, along the work in this paper, we assume that the peer's address is the ECDSA public key.

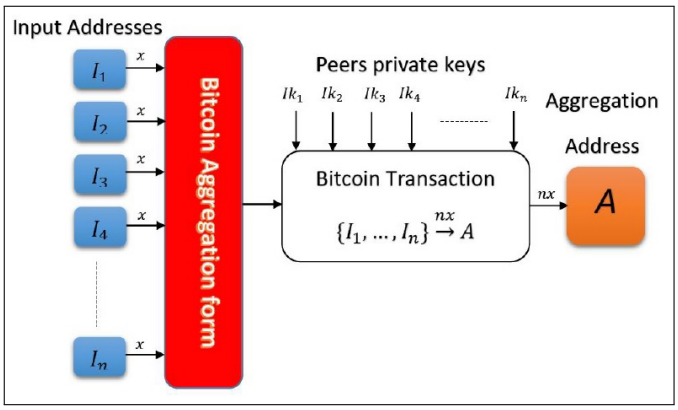

Figure 1: Deposit of the coins in the aggregation address

6.1.2 Joint Deposit of the Coins

The deposit of the coins in the aggregation address A is done in the standard way of Bitcoin transaction. Given that each peer P_i is holding the private key Ik_i corresponding to his input address I_i, the peers jointly generate one single transaction, as shown in Figure 1, containing the peers input addresses, $I_1, \cdots, I_n$, as inputs and the aggregation address A as output, $\{I_1, \cdots, I_n\} \xrightarrow{nx} A$. Notice that a transaction with several input addresses is only valid if it has been signed with all keys belonging to those input addresses [29, 37]. Thus each peer can verify whether the generated joint transaction sends the correct amount of money to the aggregation address, if this is not true, the peer just refuses to sign the transaction. If more than t peers refuse, the protocol aborts without transferring any coins. The peers do not proceed in the execution of the protocol until all the transactions are completed successfully and confirmed on the Bitcoin ledger. For n peers, there must be an amount of nx coins in the address A. The peers that do not contribute their signature are considered opted out of the protocol.

6.2 Shuffling of Destination Addresses

In this stage it is required that every peer P_i generates a fresh ephemeral personal encryption/decryption public key pair (pk_i, sk_i) of an IND-CCA secure public key cryptosystem and broadcasts the resulting public encryption key pk_i.

6.2.1 Destination Addresses Generation

Each peer P_i locally generates for himself a destination address D_i as in the standard Bitcoin address generation, as if P_i is going to make a transaction $I_i \rightarrow D_i$. This address D_i is kept secret for P_i at the moment. Each Participant P_i commits himself to his address D_i by broadcasting $e_i = \varepsilon_{pk_i}(D_i)$ as the public key encryption of his address D_i.

6.2.2 Oblivious Shuffling

The peers shuffle the freshly generated destination addresses, $D_1, \cdots, D_n$, in an oblivious manner [37], similar to the well known mix network of Chaum [30]. This is illustrated in Figure 2. First, the peers are lexicographically ordered according to their input addresses. Each peer P_i uses the encryption keys of each peer $P_{j>i}$ to create a layered encryption of his output address. Then, the peers perform a sequential shuffling, starting with peer P_1: Each peer P_i expects to receive $i-1$ ciphertexts from P_{i-1}. Upon reception, each peer strips one layer of encryption from the ciphertexts, adds her own ciphertext and randomly shuffles the resulting set, according to his picked permutation π. P_i sends the shuffled set of ciphertexts to the next peer P_{i+1}. If everybody acts according to the protocol, the decryption performed by the last peer results in a shuffled list of output addresses. The last peer broadcasts this list.

Each peer checks that his address exists in the broadcasted list and broadcasts a confirmation of existence. If a peer P_i does not find his address in the list he broadcasts a complaint (accusation). The peers enter an accusation resolution subroutine to solve this accusation. Notice that, a peer P_i that does not find his address in the list and keeps silent (i.e., does not broadcast neither a confirmation nor an accusation) is considered halted and is kicked out of the protocol.

6.2.3 Accusations Resolution (If Exist)

The peer P_i who broadcasted a complaint/accusation must be checked for his honesty. P_i is instructed to broadcast his ephemeral private key sk_i. Recall that each P_i has already broadcasted the commitment $e_i = \varepsilon_{pk_i}(D_i)$. The rest of the peers proceed to check the honesty of P_i. Each peer P_j performs as follows:

- Using sk_i, decrypts for D_i.
- Checks whether D_i is in the list of the shuffled destination addresses.
- If D_i is in the list, P_j broadcasts a complaint against P_i.

Finally, in case more than t peers broadcast a complaint against P_i, then P_i is deemed corrupt and kicked out of the protocol. The peers exit the accusation resolution subroutine. Otherwise, the peers declare P_i as honest about his complaint.

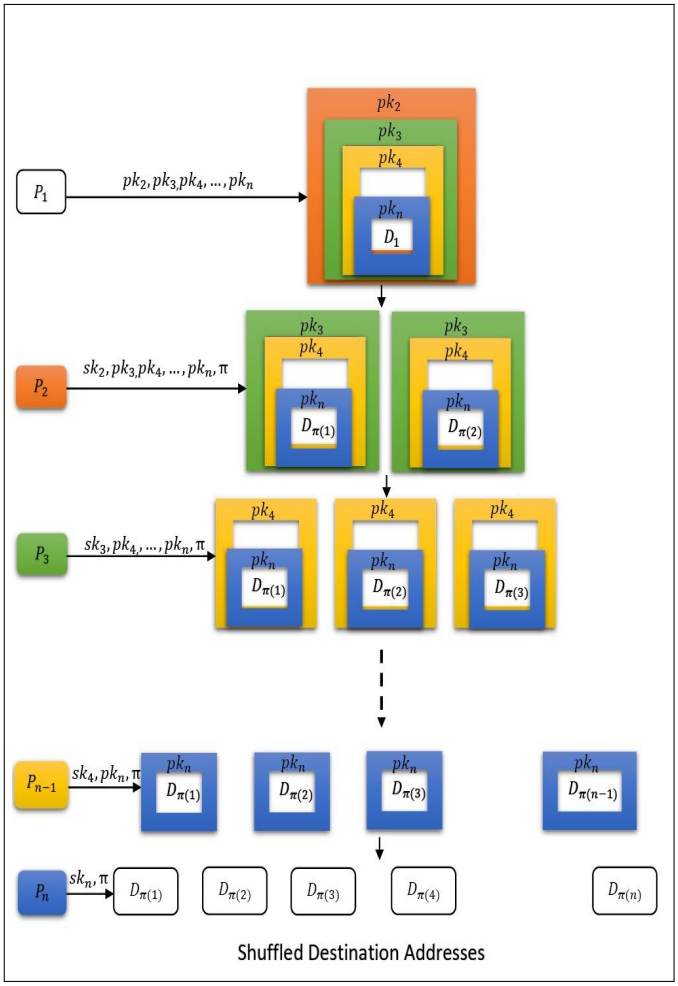

Figure 2: Oblivious shuffling of destination addresses

If the peer P_i passes the above checks, then P_i is declared honest about his complaint, but not necessarily honest about his behavior in the protocol. Still the accusation has not been solved. To proceed in solving the accusation, each peer P_i opens (broadcasts) everything; the destination address D_i, the ephemeral decryption key sk_i and the shuffled patterns. Each peer P_i performs as follows:

- Using sk_j, decrypts for D_j for all $j = 1, \cdots, n$.
- Checks whether D_j is in the list of shuffled destination addresses for all $j = 1, \cdots, n$.
- If the above check fails for any j, P_i broadcasts a complaint against P_j.

If any peer P_j receives complaints from more than t peers, P_j is deemed corrupt and is kicked out of the protocol.

Finally, all peers check the correctness of the performed shuffling and raise complaints against the misbehaved peer. The peers kick out any peer that receives more than t complaints. After all bad peers are kicked out of the protocol, the rest of the good peers repeat the address generation phase, then the shuffling phase, using new fresh pseudonyms and ephemeral keys.

6.3 Coins Distribution

The Coins distribution stage is shown in Figure 3. Let n^* be the number of kicked peers after the deposit of their x coins in the aggregation address. For the n remaining good peers, in this phase, the Bitcoin distribution form (one input to multiple recipients) is used to transfer an amount nx coins from the aggregation address A as a one input address to the output addresses, $D_{\pi(1)}, \cdots, D_{\pi(n)}$, and n^*x coins back to the input addresses, $I_{i_1}, \cdots, I_{i_{n^*}}$ of the peers that were kicked out after their deposit, such that, each address receives an exact amount of x coins. In this case the peers must join to sign the transaction, $T = A \xrightarrow{(n+n^*)x} \{D_{\pi(1)}, \cdots, D_{\pi(n)}, I_{i_1}, \cdots, I_{i_{n^*}}\}$. Let $m = H(T)$ and recall that each peer P_i holds a share $S(i)$ of the ECDSA private key s and that the corresponding public key is $A = sG$, the peers join to sign m and submit as follows.

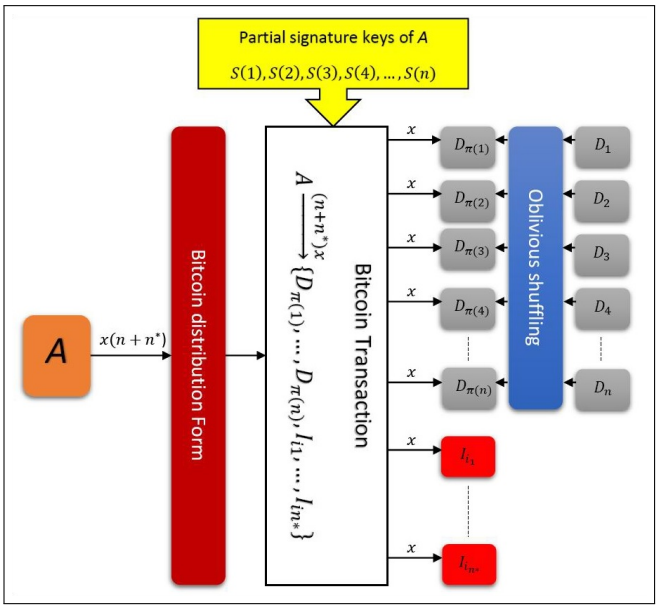

Figure 3: Coins distribution stage

Step 1. The peers execute a Pedersens JR-VSS. At the end, every peer P_i holds a share $K(i)$ of a secret $k \in_R Z_q^*$ over a polynomial $K(x)$ of degree t, with $K(0) = k$. Each peer broadcasts $G_i = K(i)G$ mod p.

Step 2. Each peer broadcasts Feldman-VSS commitments (i.e. to the base G) of all his picked random polynomials during JR-VSS in the previous step. These commitments allow the peers to validate the quantities G_i.

Step 3. Each peer is able to locally compute $(x, y) = kG$ by Lagrange interpolation and $r = x \mod q$.

Step 4. The peers that are not disqualified in the previous JR-VSS run the JVR subroutine, at the end, each peer P_i holds a share $K'(i)$ of the reciprocal k^{-1}

mod q over a t-degree polynomial (with polynomial degree reduction) $K'(x)$ with $K'(0) = k^{-1}$.

Step 5. The peers run an instant of the JZ-VSS to share a zero secret. At the end each peer holds a share $Z(i)$ over a t-degree polynomial $Z(x)$ with $Z(0) = 0$.

Step 6. Each peer P_i now holds a share $S(i)$ of s, a share $K'(i)$ of k^{-1} and a share $Z(i)$ of 0. All peers know the quantities r and m. Each peer P_i locally computes and broadcasts, $\omega_i = K'(i)[m+rS(i)]+Z(i) \mod q$.

Step 7. Using Berlekamp-Welch decoder, any peer is able to compute $\omega = k^{-1}(m + rs) \mod q$.

Step 8. All peers now know the Bitcoin ECDSA signature (r, ω) which is submitted to the Bitcoin system for verification and confirmation in the Bitcoin ledger.

Remark 1. *The above subprotocol requires a number of peers $n > 4t$. This bound on n could be reduced to $n > 3t$, if the peers run a polynomial degree reduction after each run of a JVM subroutine. Simply, assume a secret z shared on a t'-degree polynomial. Each peer shares his share of z over a t-degree polynomial. Finally, each peer sums the shares he receives from the other peers, to obtain a share on a t-degree polynomial for the same secret z.*

7 Security Analysis

In this section we give a rigor analysis of the security of SecureCoin.

7.1 Secrecy, Robustness & Revocation

In the core of the SecureCoin protocol, the joint generation of the private key of the aggregation address A (which is a uniformly distributed random value s) is shared on a threshold basis and the value $A = sG$ is publicly known. The protocol is t-secure, i.e., in the presence of at most t malicious peers:

Correctness.

- Any subset of $t + 1$ valid shares will always reconstruct to the same private key s.
- Any peer is able to locally compute the common public key A.
- The secret s is uniformly distributed in Z_q and hence, A is uniformly distributed in the subgroup generated by G.

Secrecy. A coalition of at most t peers learns no information about s except for what could be implied from the value A itself.

Robustness. From the security of robust threshold cryptography, at most t maliciously active peers will not disrupt the correctness of the protocol and will always be detected and disqualified. Executing the JR-VSS using Feldman-VSS alone has some security vulnerabilities. Malicious peers can deviate the uniform distribution of the result of Feldman's JR-VSS to a non-uniform distribution according to the attack described in [20]. More precisely, in case only Feldman-VSS is used, the attack works as follows: Assume that two traitors peers (say P_1 and P_2) want to bias the distribution towards values of A whose last bit is zero. P_1 gives members, $P_3, \cdots, P_{t+2}$, shares which are inconsistent with his broadcasted values, the rest of the members receive consistent shares. Thus, there will be t complaints against P_1, yet t complaints are not enough for disqualification. The traitors compute $\alpha = \sum_{i=1}^{n} s_i G$ and $\beta = \sum_{i=2}^{n} s_i G$. In case α ends with "0" then P_1 will do nothing and continue the protocol as written. If α ends with "1" then force the disqualification of P_1, this is achieved by asking P_2 to also broadcast a complaint against P_1, which brings the number of complaints to $t+1$. This action sets the public value A to β, which ends with "0" with probability 1/2. An illustrative example is as follows: Let "00", "01", "10", "11" be the possible two MSBs of A. Now, if A ends with zero ("00" or "01") then let α be, but if A ends with one ("10" or "11") then P_2 complains (let β be) and hence the probability that A ends with one (i.e. "11") is 1/2 (notice that "11" is the only case that makes the attack fails), hence, the attack fails with probability 1/4 and so we have. Thus effectively, the traitors have forced strings ending in "0" to appear with probability 3/4 rather than 1/2. One must notice that synchronous broadcast does not prevent such attack to take place. Hence, the third requirement for correctness and the secrecy requirement dramatically fail. Unlike Feldman-VSS, in Pedersen-VSS, the malicious peers view is independent of the value of the secret s, and therefore the secrecy of s is unconditional, which eliminates the possibility of this attack.

Revocation. Our protocol ensures that any malicious peer will be detected as soon as possible and will be revoked from the protocol. In Stage 1, the distributed generation of the aggregation address A employs JR-VSS for sharing the ECDSA private key, which allows every peer to verify the correctness of any quantity he receives from other peers. Any published malicious quantity will be detected by all good peers, if a good peer raises an accusation against a malicious peer, then all good peers will raise this accusation and since there are more good peers than bad peers, the malicious peer (through majority voting) will be revoked. In Phase 1.2, the signing of the transaction with several input addresses is valid if it has been signed with all keys belonging to those input addresses. The signing of each peer is visible to every other peer, therefore, the peer with an

incorrect signature will always be detectable by all other peers and revoked by good peers. In Stage 2, the accusation resolution (Phase 2.3), ensures that any malicious peer during the oblivious shuffling will be detected by other good peers and revoked. The Coins distribution (Stage 3), employs verifiable secret sharing where, again, ensures the detection and revocation of any malicious peer. Step 7 in the Coins distribution employs the Berlekamp-Welch decoder, which is capable of filtering out any faulty shares and ensures the correct computation of the final signature.

7.2 Anonymity and Unlinkability

Unlinkability and randomness depend on the shuffling of destination addresses stage. If there is a raised accusation in the shuffling phase of this stage, the protocol enters the accusations resolution phase where all accusations are solved, malicious peers are kicked out and destination addresses of malicious peers are burnt. Given the threshold t which is the maximum number of possible malicious peers, if at least $t+1$ peers raise an accusation against a certain peer, then at least one peer is honest about his accusation. The employed ephemeral public key encryption is IND-CCA secure, which means that an adversary cannot link destination addresses from the broadcasted ciphertexts $e_1, \cdots, e_n$, as they are randomized by the nature of a CCA secure cryptosystem. Based on observations of the blockchain an attacker can try to guess the mapping between a participant's input and output address. The set of addresses among which the attacker has to guess is the anonymity set and its size is the achieved anonymity level. A larger anonymity set leads to a smaller probability of a correct guess and hence more anonymity. In the following, we analyze peer's anonymity against blockchain observers (outsiders) and against participating peers.

7.2.1 Indistinguishability of Blockchains

We emphasize that, Blockchain observers are outsiders, i.e., they have absolutely no contact with any of the peers participated in the protocol. The SecureCoin protocol uses two types of transaction forms: Aggregation transaction and Distribution transaction. Both of these transaction forms are widely used individually by thousands of Bitcoin peers. Aggregation transactions are used by a peer –for example– to clean his wallet, i.e., when the peer has many pseudonyms in his wallet he uses the aggregation form to transfer his coins simultaneously from different pseudonyms to a single new fresh pseudonym and wipes out the old pseudonyms. Also, they are used when a peer is buying goods which cost a large amount of coins, so he spends the amount of coins from several input addresses of his own to the seller's address. On the other hand, distribution forms are used by an individual peer when he wants to distribute funds (e.g. salary) to multiple employees addresses. SecureCoin uses these two types of transactions in the same standard way of Bitcoin transactions, which make the transactions intended for anonymity by a group of peers in SecureCoin indistinguishable from transactions made by thousands of individual peers over the globe for different purposes.

One problem remains, that may threaten the above indistinguishability. The amount x transfered to/from the aggregation address is fixed. Blockchain observers may observe that (i) some fixed amount x is transfered to an address A' from multiple addresses and (ii) the same amount x leaves this address A' to multiple addresses. Hence, this address A' is more likely to be an aggregation address used for anonymity purpose. Although the observer cannot map an input address to the corresponding destination address, just knowing that these addresses are more likely to be involved in an anonymous transaction (i.e. belong to the same set of peers) is a security vulnerability that must be solved. To fix this problem, notice that it is unlikely that all peers have exactly the same amount x as an unspent transaction. A peer P_i may have some arbitrary x_i as an unspent transaction. In case $x_i < x$, then P_i does not qualify to participate from the very beginning. Now, each peer P_i prepares the transaction $I_i \xrightarrow{x_i} A$ and a return transaction $A \xrightarrow{x_i - x} D_i^*$, where D_i^* is a new fresh pseudonym (destination address) for P_i. The transactions $I_i \xrightarrow{x_i} A$ are signed in the aggregation phase as described while the return transactions $A \xrightarrow{x_i - x} D_i^*$ are signed as part of the transaction in the coins distribution stage to return the change back to their new addresses. In this way, the aggregation and distribution transactions are made with different amount of funds and so the problem is solved.

7.2.2 Indistinguishability of Participating Peers

The participating peers inevitably learn which input and output addresses are involved in the shuffling operation, as they have to sign the corresponding transactions and release them to the Bitcoin network. However, since participating peers do not learn which output belongs to which input address, the anonymity level against participating peers is equal to the remaining number of participants n after all accusations have been solved, which is as good as in the CoinShuffle protocol. Notice that, by nature, the shuffling is insecure if the number of participating peers are less than three. Since, for two peers, one peer will recognize his destination address and immediately maps the other address to the other peer.

7.3 Protection Against Sabotage Attacks

Peers that may withdraw trying – not only to disrupt the progress of the protocol – but to make other participating peers loose their funds with impossibility to return these funds back to their input addresses. It is Ok that the protocol is disrupted and terminated with the coins of each participating peer still in their wallets. In this case, the success of the protocol based on the existence of at

most t malicious peers is accepted. However, it is not accepted at all that the peers may loose their funds given this threshold, while the saboteur peers may run away without loosing the same amount of funds too.

We show that our protocol ensures that such an attempted attack will not succeed without the same loss from the attackers as the good peers. In fact, the sabotage attack becomes serious after the coins has been transfered from the peers accounts to another address. In SecureCoin, once the aggregation address A is established and the malicious peers are disqualified, the remaining peers jointly deposit their coins in A simultaneously. Since individual deposit of each peer allows possible saboteur peers (exceeding the threshold) to withdraw after few transactions have been made, leaving other peers unable to even jointly return their deposits back to their input addresses. Simultaneous deposit ensures that either, the protocol will be terminated without a loss or, the saboteurs will loose their coins too. Therefore, they are unlikely to misbehave after the deposit phase. This is actually a great improvement over the CoinParty protocol, where the mixing peers controlling the temporary address do not have any financial share in the addresses and hence, they may run away leaving the input peers stuck with their coins in the temporary addresses with no way to undo the transactions.

Finally, as has been shown, our protocol does not prevent a sabotage attack to take place, actually it is impossible to prevent it. However, the protocol ensures that if such attack is attempted, then either, no body will loose, or every body will loose.

7.4 Unforgeability

Our protocol employs the ECDSA in the same way used by the Bitcoin ecosystem. To a verifier, the generated signature by our protocol is completely indistinguishable from a signature generated by a Bitcoin wallet. Therefore, given that the Bitcoin signature is unforgeable, and that our threshold key generation is t-secure, our protocol is also unforgeable by an adversary that is able to corrupt at most t peers.

7.5 Deniability

Deniability against outsiders is achieved by the indistinguishability of the SecureCoin transactions and normal transactions. We argue that if there are at any point many more non-SecureCoin aggregated and distributed transactions than SecureCoin transactions in the Bitcoin network, a peer can plausibly deny having participated in SecureCoin. Our inspection of the public ledger shows that there are indeed many non-SecureCoin transactions of the same form as those issued by SecureCoin. Aslo, deniability holds against SecureCoin peers that were kicked out of the protocol prior to the establishment of the aggregation address A. Since, in this case, those kicked out peers do not know neither, the aggregation address nor the input addresses of other peers. However, SecureCoin does not provide deniability against participating peers as long as they knew the aggregation address A. Deniability in this case is an outstanding problem in SMC in general and not due to our protocol. Next we show that the claimed deniabilty against input peers of CoinParty is questionable. CoinParty authors stated that, *Deniability against mixing peers is not achieved because they learn which in-and-output addresses participated in the mixing during the shuffling phase.* Mixing peers also know the identities of the input peers. We argue that mixing peers are outsiders with malicious minority, and hence, the claimed deniability of CoinParty fails.

8 Evaluation and Comparisons

In this section, we evaluate the efficiency of SecureCoin and compare its performance to recent protocols.

8.1 Full Compatibility

The deviation of SecureCoin is transparent to standard Bitcoin clients since, Bitcoin ecosystem is not concerned how the ECDSA keys are generated as long as they are a valid Bitcoin ECDSA key pair. Also, Bitcoin ecosystem is not concerned how the signature is performed as long as the signature is a valid Bitcoin signature and the transaction is in the correct form. In other words, the Bitcoin ecosystem is the verifier while the peer's wallet is the generator of the transaction. Bitcoin has nothing to do with the peer's wallet and how it generates and signs transactions as long as they are in the correct form. In our threshold ECDSA, a verifier receiving a signature cannot distinguish whether this signature is generated by a single signer or by a group of signers on a threshold basis. Hence, SecureCoin is fully compatible with the Bitcoin ecosystem.

8.2 Cost Efficiency

It is known that the processing of a Bitcoin transaction of roughly less than 1KB will not be charged. The amount charged per 1KB defaults to 0.0001 XBT. Let n_i and n_o be the number of input addresses and output addresses of a transaction respectively. The size S of a transaction, assuming compressed public keys, could be roughly estimated based on the simple formula, $S = 148n_i + 34n_o + 10$ Bytes[3]. SecureCoin requires two transactions: An aggregation transaction with $n_i = n$ and $n_o = 1$ and a distribution transaction with $n_i = 1$ and $n_o = n$. For $n = 6$ participants, based on this formula, the first transaction is of size $S = 932$ Bytes while the second is of size $S = 362$ Bytes. Hence, no processing fees are due.

[3] http://bitcoinfees.com/

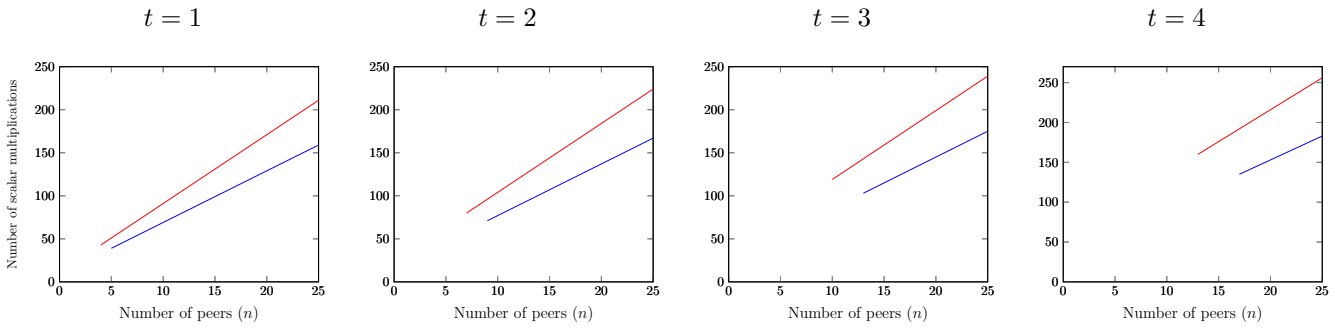

Figure 4: Number of scalar multiplications of SecureCoin for the threshold-ECDSA at different values of the threshold and the number of peers: blue ($n > 4t$), red ($n > 3t$).

8.3 Computation Complexity

Consider the joint generation of the address A. This requires one invocation of a JR-VSS which by its turn requires $2(t+1)+2(n-1)$ EC point multiplications.

Now Consider the threshold ECDSA signature in the Coins distribution stage. In step 1, the JR-VSS with Pedersen-VSS requires $2(t+1)+2(n-1)$ EC point multiplications. Step 2, requires no extra computations by each peer. Step 3, requires each peer to perform Lagrange interpolation on the broadcasted values in step 2, and hence, requires $t+1$ EC point multiplications. Step 4, requires the JR-VSS inside the JVR subroutine over $2t$-degree polynomial, which requires $2(2t+1)+2(n-1)$. In Step 5, the JZ-VSS is similar to the JR-VSS in step 1. Step 6, 7 and 8, requires no EC point multiplications. These total $8t+6n+1$ EC point multiplications that must be performed by each peer. The above computations assume $n > 4t$. In case $n > 3t$ the computation complexity increases due to the employment of polynomial degree reduction in step 4. In this case, it requires a total of $t^2+10t+8n$ EC point multiplications. These are illustrated graphically in Figure 4 for different values of the threshold and number of participating peers.

We notify the recent work of [18] that may provide a slightly improved efficiency for the implementation of threshold ECDSA. However, the protocol in [18], although it is unforgeable, it sacrifices robustness for the sake of efficiency and is not suitable for the goals of SecureCoin.

8.4 Comparisons

In this subsection, we compare the complexity and security of our protocol with previous protocols.

8.4.1 With CoinParty

Based on the transaction size, CoinParty is more efficient since all transactions are one-to-one. However, the employed mixing peers will not provide their services for free. In addition, they are many (at least 4). We cannot give any estimate of the cost because this has not been standardized yet. However, the cost will be relatively high to avoid sabotage.

CoinParty has the major problem of its vulnerability to sabotage attacks. Saboteur peers involved in the mixing and holding the private keys of the temporary addresses may abort the protocol after the input peers make their deposit, leaving them stuck with the impossibility to refund. The saboteurs did not loose any thing, since they have no share in the funds. They also may conspire to steal the coins. Our protocol ensures that if such sabotage attack is attempted, then either, the protocol terminates with all peers, including the saboteurs loosing their funds, or the protocol terminates with no loss at all. This is ensured by phase 1.2 of stage 1, joint deposit ensures that all peers will contribute in the deposit of the same amount x. A transaction with several input addresses is only valid if it has been signed with all keys belonging to those input addresses. Therefore, this phase ensures the simultaneous contribution of funds.

CoinParty runs multiple instants of the threshold key generation and the threshold digital signature (n instants). These protocols are complex by nature as we have shown, so running multiple instants is significantly complex specially if these protocols are supposed to run on smart devices. Figure 5, shows a comparison between our SecureCoin protocol and CoinParty for different values of the threshold and number of peers. It illustrates the dramatic increase in the number of EC point multiplications required in CoinParty over that required by SecureCoin. SecureCoin invokes the threshold protocols only once. This provides a significant complexity improvement over CoinParty.

8.4.2 With CoinShuffle

The cost-efficiency of our protocol proves efficiency over CoinShuffle. This is illustrated in Figure 6. It shows that, for $n = 6$, our protocol requires one aggregation transaction of size 932 Bytes and one distribution transaction of size 362 Bytes. On the other hand the CoinShuffle requires one multi-input multi-output transaction of size 1102 Bytes which is charged. Increasing n, Figure 6 shows that as long as $n < 14$ our protocol charges the participants with only 1KB, while this is limited to $n < 11$ in the CoinShuffle protocol. The CoinShuffle protocol has

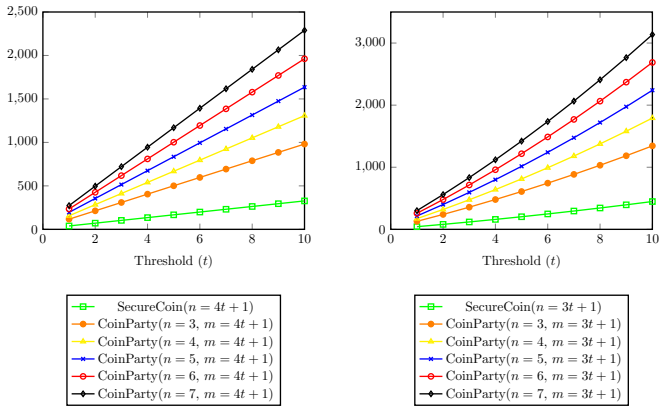

Figure 5: The number of scalar multiplications required by SecureCoin and CoinParty (m=number of mixing peers, n=number of input peers)

a security vulnerability that the last peer in the Shuffle protocol knows the set of shuffled destination addresses in the clear and the set of input addresses. This allows him to rearrange the shuffled version in a way to map certain input addresses to certain output addresses and benefit from this behavior specially if he collaborates with others in the protocol. SecureCoin eliminates this vulnerability since the set of input addresses and the set of destination addresses are isolated by the aggregation address A.

The bulk (exactly n-input to exactly n-output) transaction performed by CoinShuffle for a significant number of addresses is not a commonly used form by Bitcoin and makes it distinguishable by the ledger's observers as this type of transaction is rarely to be performed by an individual. Actually, there is no reason for an individual to perform such a costly transaction. Our protocol avoids this type of transaction. Instead, the aggregation transaction and the distribution transaction used in our protocol are performed frequently by many individuals in the Bitcoin network.

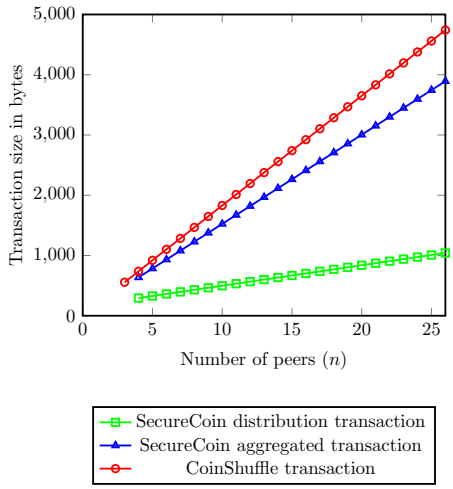

Figure 6: Comparison of transaction size between CoinShuffle and SecureCoin

8.5 Computation Time

Let MM denotes a modular multiplication operation while MA denotes a modular addition operation. The EC point addition (PA) requires 8MM+3MA operations. On the other hand, a point doubling requires 3MM+4MA operations [21]. The most basic technique for performing an EC scalar multiplication (SM), kG for an integer k, is the double-and-add method, which works in a similar way as the square-and-multiply method for exponentiation. Given a scalar k of a length of n bits, the double-and-add approach executes n point doubling and on the average of $n/2$ point additions; the exact number of point additions depends on the Hamming weight of k. Therefore, the overall cost of the double-and-add method to perform SM amounts to $3n + 8n/2 = 7n$ multiplications and $4n + 3n = 2 = 5.5n$ squarings over $\mathbb{F}_p$.

A better strategy for computing kG is to decompose the n-bit scalar k into two half-length integers k_1 and k_2 (often referred to as balanced length-two representation of k [21]). As a result, the overall cost amounts to $(0.5n)(3) + (0.375n)(8) = 4.3n$ multiplications and $(0.5n)(4) + (0.375n)(3) = 3.125n$ squarings in $\mathbb{F}_p$. However, the Hamming density can be reduced to 0.5 (on average) by representing k_1 and k_2 in Joint Sparse Form (JSF) [39], which, in turn, cuts the number of point additions by roughly one third to $0.5(n = 2) = 0.25n$. In this case, the total cost of computing kP is reduced to – on the average of – $(0.5n)(3) + (0.25n)(8) = 3.5n$ multiplications and $(0.5n)(4) + (0.25n)(3) = 2.75n$ squarings in $\mathbb{F}_p$.

To evaluate the computation time of our protocol, an implementation of the basic field arithmetic over prime field $\mathcal{F}_p$ for a 256-bit p secp256k1 on a mobile phone is shown in Table 1. This below-moderate specifications mobile device runs Android OS V2.3 on a CPU 1 GHz Scorpion and 768 MB RAM.

Table 1: Computation time of basic field arithmetic operations on a mobile phone (HTC-Desire)

Operation	Computation time
Multiplication	990.2 ns
Addition	121.2 ns
Subtraction	129.5 ns
Inverse	190.1 μs
Squaring	859.5 ns

Based on Table 1, we can determine the time taken by this device to perform EC operations. The SM operation requires, $(3.5)(990.2$ ns$)(256$ bits$) = 0.88$ ms, plus $(2.75)(859.5$ns$)(256$ bits$) = 0.6$ ms. Hence, a scalar multiplication takes about 1.5 ms. Let $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ and $P_3 = (x_3, y_3)$ be three points on the elliptic curve. Now let $P_3 = P_1 + P_2$. P_3 is computed as follows: compute $\lambda = (y_2 - y_1) = (x_2 - x_1)$, $v = (y_1x_2 - y_2x_1) = (x_2 - x_1), x_3 = \lambda^2 - x_1 - x_2$ and $y_3 = \lambda(x_1 - x_3) - y_1$. All computations are modulo p. Now we can concretely find what it takes to perform one

point addition on the EC. Computing λ requires two modular subtractions and one modular inversion. Computing v (given λ was computed) requires two modular multiplications and one modular subtraction. Computing x_3 requires one modular squaring and one modular subtraction while y_3 requires two modular subtractions and one modular multiplication. These totals six modular subtractions, one modular inversion, one modular squaring and three modular multiplications. From Table 1, we have for one point addition, $6(0.1295\mu s) + 190.1\mu s + 0.8595\mu s + 3(0.9902\mu s) = 0.195$ ms. These are summarized in Table 2.

The above implementation shows that scalar multiplication on elliptic curve is expensive compared to other modular operations. Based on Table 2, Figure 7, illustrates the threshold-ECDSA execution time required by SecureCoin compared to that required by CoinParty. Figure 7, shows that, for a threshold t up to the value of 10, which is a large value, the computations required by SecureCoin is less than one second.

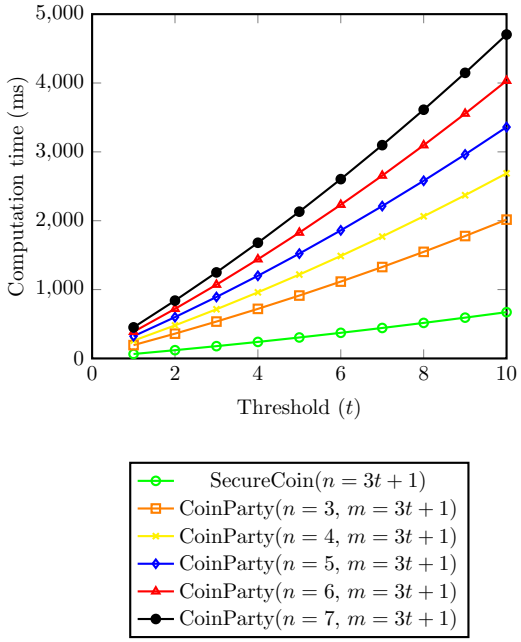

Figure 7: Comparison of Computation time of threshold-ECDSA between CoinParty and SecureCoin

Table 2: EC Computation time on a mobile phone. SM: Scalar Multiplication, PA: Point Addition

Operation	Computation time
SM	1.5 ms
PA	0.195 mss

9 Conclusions

Several contributions have been proposed recently to countermeasure the attacked anonymity of Bitcoin addresses. However, by analyzing these protocols, serious vulnerabilities have been revealed. CoinShuffle performs a bulk transaction of exactly n input addresses to n output addresses of the same amount which is easily observed on the blockchain. In CoinParty, the input peers are exposed to sabotage attacks by mixing peers, and in order to reduce the risk of such an attack, the return fees for the mixing peers are dramatically increased. In this paper, we proposed SecureCoin as a robust and secure protocol for achieving anonymity service in Bitcoin. Our protocol provides better protection for the participating peers against malicious behavior of minority of the peers and protection against the most serious sabotage attack attempted by any number of saboteur peers. We analyzed the security of our scheme and evaluated its efficiency. Finally, we compared our protocol to recently proposed protocols and showed that our protocol proves efficiency over these protocols and requires less fees by the Bitcoin ecosystem.

References

[1] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security*, pp. 34–51, Springer, 2013.

[2] A. M. Antonopoulos, *Mastering Bitcoin: Unlocking Digital Cryptocurrencies*, O'Reilly Media, Inc., 2014.

[3] J. Bar-Ilan and D. Beaver, "Non-cryptographic fault-tolerant computing in constant number of rounds of interaction," in *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, pp. 201–209, 1989.

[4] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for noncryptographic fault-tolerant distributed computation," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pp. 1–10, 1988.

[5] S. Blake-Wilson and M. Qu, *Standards for Efficient Cryptography (SEC) 2: Recommended Elliptic Curve Domain Parameters*, Certicom Research, Oct 1999.

[6] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for bitcoin with accountable mixes," in *Financial Cryptography and Data Security*, pp. 486–504, Springer, 2014.

[7] J. Camenisch and A. Mityagin, "Mix-network with stronger security," in *Privacy Enhancing Technologies*, pp. 128–146, Springer, 2006.

[8] D. Chaum, "The dining cryptographers problem: Unconditional sender and recipient untraceability," *Journal of Cryptology*, vol. 1, no. 1, pp. 65–75, 1988.

[9] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[10] H. Corrigan-Gibbs and B. Ford, "Dissent: accountable anonymous group messaging," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, pp. 340–350, 2010.

[11] I. Damgard, M. Geisler, M. Kroigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *Public Key Cryptography (PKC'09)*, pp. 160–179, Springer, 2009.

[12] G. Danezis, C. Fournet, M. Kohlweiss, and B. Parno, "Pinocchio coin: Building zerocoin from a succinct pairing-based proof system," in *Proceedings of the First ACM workshop on Language Support for Privacy-enhancing Technologies*, pp. 27–30, 2013.

[13] R. Dingledine, N. Mathewson, and P. Syverson, *Tor: The Second-generation Onion Router*, Technical report, DTIC Document, 2004.

[14] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th IEEE Annual Symposium on Foundations of Computer Science*, pp. 427–438, 1987.

[15] J. Furukawa and K. Sako, "An efficient scheme for proving a shuffle," in *Advances in Cryptology (CRYPTO'01)*, pp. 368–387, Springer, 2001.

[16] R. P. Gallant, R. J. Lambert, and S. A. Vanstone, "Faster point multiplication on elliptic curves with efficient endomorphisms," in *Advances in Cryptology (CRYPTO'01)*, pp. 190–200, Springer, 2001.

[17] C. Garman, M. Green, I. Miers, and A. D. Rubin, "Rational zero: Economic security for zerocoin with everlasting anonymity," in *Financial Cryptography and Data Security*, pp. 140–155, Springer, 2014.

[18] R. Gennaro, S. Goldfeder, A. Narayanan, "Threshold-optimal DSA/ECDSA signatures and an application to Bitcoin wallet security,", *IACR Cryptology ePrint Archive*, vol. 2016, pp. 13, 2016.

[19] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2007.

[20] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin, "Secure distributed key generation for discrete-log based cryptosystems," *Journal of Cryptology*, vol. 20, no. 1, pp. 51–83, 2007.

[21] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer Science & Business Media, 2006.

[22] M. H. Ibrahim, "A method for obtaining deniable public-key encryption," *International Journal of Network Security*, vol. 8, no. 1, pp. 1–9, 2009.

[23] M. H. Ibrahim, "Receiver-deniable public-key encryption," *International Journal of Network Security*, vol. 8, no. 2, pp. 159–165, 2009.

[24] M. H. Ibrahim, "Resisting traitors in linkable democratic group signatures," *International Journal of Network Security*, vol. 9, no. 1, pp. 51–60, 2009.

[25] M. H. Ibrahim, I. A. Ali, I. I. Ibrahim, and A.H. El-sawy, "A robust threshold elliptic curve digital signature providing a new verifiable secret sharing scheme," in *IEEE 46th Midwest Symposium on Circuits and Systems*, vol. 1, pp. 276–280, 2003.

[26] I. Ingemarsson and G. J. Simmons, "A protocol to set up shared secret schemes without the assistance of a mutually trusted party," in *Advances in Cryptology (EUROCRYPT'90)*, pp. 266–282, Springer, 1991.

[27] M. Jakobsson, "Flash mixing," in *Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing*, pp. 83–89, 1999.

[28] M. Jakobsson and A. Juels, "An optimally robust hybrid mix network," in *Proceedings of the Twentieth Annual ACM Symposium on Principles of Distributed Computing*, pp. 284–292, 2001.

[29] G. Maxwell, *Coinjoin: Bitcoin Privacy for the Real World*, Bitcoin Forum, aug. 2013. (`https://bitcointalk.org/index.php`)

[30] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 ACM Conference on Internet Measurement Conference*, pp. 127–140, 2013.

[31] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from bitcoin," in *IEEE Symposium on Security and Privacy (SP'13)*, pp. 397–411, 2013.

[32] C. A. Neff, "A verifiable secret shuffle and its application to e-voting," in *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pp. 116–125, 2001.

[33] M. Ohkubo and M. Abe, "A length-invariant hybrid mix," in *Advances in Cryptology (ASIACRYPT'00)*, pp. 178–191, Springer, 2000.

[34] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology (CRYPTO'91)*, pp. 129–140, Springer, 1992.

[35] B. Pfitzmann and A. Pfitzmann, "How to break the direct rsa-implementation of mixes," in *Advances in Cryptology (EUROCRYPT'89)*, pp. 373–381, Springer, 1990.

[36] F. Reid and M. Harrigan, *An Analysis of Anonymity in the Bitcoin System*, Springer, 2013.

[37] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for bitcoin," in *Computer Security (ESORICS'14)*, pp. 345–364, Springer, 2014.

[38] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. V. Zerocash, "Decentralized anonymous payments from bitcoin," in *IEEE Symposium on Security and Privacy (SP'14)*, pp. 459–474, 2014.

[39] J. A Solinas, *Low-weight Binary Representations for Pairs of Integers*, Technical Report, 2001. (`http://cacr.uwaterloo.ca/techreports/2001/corr2001-41.ps`)

[40] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "Coinparty: Secure multi-party mixing of bitcoins," in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pp. 75–86, 2015.

Maged Hamada Ibrahim received B.Sc. in Communications and Computers Engineering from Helwan University, Cairo, Egypt, with Distinction and Honor's Degree in 1995. He also obtained his M.Sc. from the same University in 2001. Then his Ph.D. from Helwan University in 2005. He is now an Associate Professor at Helwan University. He is joining several network security projects in Egypt. His main interest is engineering cryptography and communications security. More specifically, working on the design of efficient and secure cryptographic algorithms and protocols, in particular, secure distributed multiparty computations, public key infrastructures, digital signatures, digital rights management protocols and non-cryptographic solutions to telecommunication security problems. Other things that interest him are number theory and the inspection of mathematics for designing secure and efficient cryptographic schemes.