MProve: A Proof of Assets Protocol for Monero Exchanges

Arijit Dutta, Saravanan Vijayakumaran
Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai, India
arijit.dutta@iitb.ac.in, sarva@ee.iitb.ac.in

Abstract

Theft from cryptocurrency exchanges due to cyberattacks or internal fraud is a major problem. Exchanges can partially alleviate customer concerns by providing periodic proofs of solvency. We describe MProve, a proof of assets protocol for Monero exchanges which can be combined with a known proof of liabilities protocol to provide a proof of solvency. MProve also provides a simple proof of non-collusion between exchanges.

Index Terms

Cryptocurrency, Monero, proof of assets

I. Introduction

Cryptocurrency exchanges represent a tradeoff between convenience and security for their customers. An exchange provides its customers with a familiar login interface on a website where they can view, transfer, or trade their cryptocurrency balances. In case customers forget their password, they can recover it using multi-factor authentication or by contacting the exchange's customer support. The customer is free from the hassle of storing the private keys corresponding to their cryptocurrency funds, which are subject to loss and theft. But these conveniences for the customers come at the risk of their funds being stolen from the exchange due to hacking or internal fraud. The loss of bitcoins from the Mt. Gox exchange in 2014 is the most infamous embodiment of this risk [1]. Despite being acutely aware of the need to secure their systems, several cryptocurrency exchanges have failed to prevent theft of their assets. An estimated $927 million worth of cryptocurrencies were reported as stolen from exchanges in the first nine months of 2018 [2].

While ensuring security of the systems and protocols used by cryptocurrency exchanges is an important problem, we do not address it in this work. Instead, we consider protocols by which exchanges can provide *proofs of solvency* to their customers. These proofs give the customers assurance that an exchange continues to hold enough cryptocurrency assets to cover their cumulative liabilities. There are two main benefits of such proofs: (1) they prevent exchanges from concealing the loss of cryptocurrency funds due to cyberattacks, and (2) they prevent exchanges from selling cryptocurrency assets to customers without actually owning the required quantities of the assets.

Monero is a privacy-focused cryptocurrency which uses ring signatures, one-time addresses, and cryptographic commitments to obfuscate the sources, destinations, and amounts in a transaction [3]. In this paper, we describe MProve, a proof of assets protocol for Monero exchanges. This protocol can be combined with a known proof of liabilities protocol to provide a proof of solvency.

II. Related Work

The first known approach for generating a proof of solvency is attributed to Greg Maxwell and Peter Todd [4]. But this approach revealed the total amount of assets held by an exchange and the cryptocurrency addresses storing these assets. To address these issues, Dagher *et al.* proposed a scheme called Provisions which generates privacy-preserving proofs of solvency for Bitcoin exchanges [5]. It consists of three main protocols and an optional fourth protocol:

1) *Proof of assets:* The exchange chooses a set $\mathcal{P}$ of *unspent* addresses with known public keys from the Bitcoin blockchain such that it knows the private keys corresponding to a subset $\mathcal{P}_{\text{known}} \subset \mathcal{P}$. The exchange then creates a Pedersen commitment C_{assets} [6] to the total amount of bitcoins available in the $\mathcal{P}_{\text{known}}$ addresses. The commitment C_{assets} is accompanied by a proof that the exchange included the amount corresponding to an address in $\mathcal{P}$ only if it knows the corresponding private key. The proof preserves the exchange's privacy in the sense that the set $\mathcal{P}_{\text{known}}$ is not revealed.
2) *Proof of liabilities:* The exchange publishes a list of liabilities which contains an entry for each of its customers. Each entry contains Pedersen commitments to the bits in the binary representation of a particular customer's Bitcoin balance. These commitments can be combined (even by a third party) to generate a Pedersen commitment C_i to the balance of the ith customer. The ith customer can check the correctness of C_i by logging in to the exchange and obtaining the blinding factor used to generate C_i. Finally, a Pedersen commitment to the total liabilities of the exchange is computed as $C_{\text{liabilities}} = \sum_i C_i$. This scheme is not foolproof as customer omissions from the list of liabilities can be detected only if one of the omitted customers checks the list.
3) *Proof of solvency:* The exchange computes $C_{\text{difference}} = C_{\text{assets}} - C_{\text{liabilities}}$ and gives a zero-knowledge proof that $C_{\text{difference}}$ is a Pedersen commitment to the zero amount.
4) *Proof of non-collusion (optional):* To prevent exchanges from sharing unspent addresses while generating their individual proofs of assets, a proof of non-collusion is required. The proof of assets protocol in Provisions involves publishing a list L_1 containing Pedersen commitments to the private keys corresponding to the public keys in $\mathcal{P}_{\text{known}}$. Using the same private keys, a second list L_2 is computed containing public keys generated using a base point which is different from the default base point of the secp256k1 elliptic curve used in Bitcoin. Each exchange gives a zero-knowledge proof that L_2 is a permutation of L_1 after base change and removal of blinding factors. If two exchanges were to use the same unspent address, their respective L_2 lists would have an element in common, revealing their collusion. This proof of non-collusion reveals the number of Bitcoin addresses owned by an exchange. For this reason, it was presented as an optional protocol by the designers of Provisions.

The proofs of liabilities and solvency in Provisions can be used without modification for Monero exchanges. But the proof of assets protocol in Provisions cannot be used due to the privacy-focused design of Monero. In the next section, we give a brief overview of the Monero protocol. After a discussion of the challenges in using the Provisions proof of assets protocol for Monero exchanges, we describe the MProve protocol in Section IV. By design, the MProve protocol gives a proof of non-collusion between exchanges which does not reveal the number of addresses owned by an exchange. In Section V, we briefly discuss the application of MProve to Bitcoin exchanges.

III. Overview of Monero

The design of Monero is based on the CryptoNote protocol [7]. In a Monero transaction, *one-time addresses* (also called *stealth addresses*) are used to hide the recipient of the funds. One-time addresses are just public keys whose private keys are generated using the recipient's long-term public keys and a Diffie-Hellman protocol. The source of funds in a transaction will also be a one-time address which was the destination of a previous transaction. Ring signatures are used to conceal the sender address among a set of other one-time addresses randomly chosen from the blockchain. The ring signature in a transaction proves that the signer knows one of the private keys corresponding to the set of one-time addresses specified in the transaction. The precise one-time address which was the source of funds is not revealed. To prevent double spending, *linkable ring signatures* proposed in [8] are used in Monero with some modifications [9]. Two linkable ring signatures which spend from the same one-time address will have identical *key images* revealing the double spend.

The original Monero protocol implementation did not hide the amounts involved in a transaction. To spend from a one-time address storing a certain amount of Monero, the spender had to create the ring signature using other one-time addresses from the blockchain which were also storing the same amount. In 2017, the Monero protocol added *confidential transactions* [10] which hide the transaction amounts using Pedersen commitments.

In the following subsections, we describe those aspects of Monero addresses and transactions needed to present our proof of assets protocol.

A. Monero Public Keys and One-Time Addresses

Monero public keys are points on the elliptic curve used in EdDSA generated by the base point $G = (x, 4/5)$ with positive x [7], [11]. The order of G is a prime l which is larger than 2^{252}. Monero private keys are integers in the set $\mathbb{Z}_l^+ = \{1, 2, \ldots, l-1\}$. Let $\mathcal{E}$ denote the subgroup of the EdDSA curve generated by the base point G. We will use additive notation for the group operation on the curve. The public key $A \in \mathcal{E}$ corresponding to the private key $a \in \mathbb{Z}_l^+$ is given by

$$A = aG = \underbrace{G + \cdots + G}_{a \text{ times}}.$$

To receive funds in the Monero system, the recipient shares a pair of public keys $(A_{\text{vk}}, A_{\text{sk}})$ with the sender, where $A_{\text{vk}} = a_{\text{vk}}G, A_{\text{sk}} = a_{\text{sk}}G$. The subscripts vk and sk denote the *view key* and *spend key* respectively. The received funds cannot be spent without knowledge of the private spend key a_{sk}. On the other hand, the private view key a_{vk} can be shared with third parties allowing them to view (but not spend) the fund receipts.

Suppose Bob wants to send funds to Alice whose public key pair is $(A_{\text{vk}}, A_{\text{sk}})$. He chooses a random integer $r \in \mathbb{Z}_l^+$ and computes the points rA_{vk} and $H(rA_{\text{vk}})G$, where $H : \mathcal{E} \mapsto \mathbb{Z}_l^+$ is a hash function which maps curve points to integers. Bob creates and broadcasts a transaction which specifies the one-time address $P = H(rA_{\text{vk}})G + A_{\text{sk}}$ as the destination of the funds. The transaction also contains the point $R = rG$ which will enable Alice to recover the private key corresponding to P. The transaction will eventually be added to the blockchain as part of a block.

From every transaction which appears on the blockchain, Alice reads the one-time address P and random point R. Using her private view key a_{vk}, she computes the point $P' = H(a_{\text{vk}}R)G + A_{\text{sk}}$. For the transaction Bob created using Alice's public key pair, P' will be equal to P since $rA_{\text{vk}} = ra_{\text{vk}}G = a_{\text{vk}}R$. By checking for this equality, Alice can identify transactions which are sending funds to her. The private key corresponding to P is given by $H(a_{\text{vk}}R) + a_{\text{sk}}$. Hence the private view key a_{vk} can be safely revealed to third parties to outsource the work of scanning the blockchain for incoming transactions.

B. Linkable Ring Signatures

While digital signatures prove knowledge of the private key corresponding to a public key, ring signatures prove knowledge of *one of the private keys* corresponding to a set of public keys. Suppose Alice wants to spend the funds from a one-time address P for which she knows the private key. She creates a linkable ring signature on a message m as follows:

1. She assembles a list of one-time addresses $\mathcal{P} = (P_0, P_1, \ldots, P_{n-1})$ from the blockchain such that $P_j = P$ for exactly one $j \in \{0, 1, \ldots, n-1\}$.
2. Let $x_i \in \mathbb{Z}_l^+$ be the private key corresponding to P_i, i.e. $P_i = x_iG$. Using the private key corresponding to P_j, she computes the key image $I = x_jH_p(P_j)$ where $H_p : \mathcal{E} \mapsto \mathcal{E}$ is a hash function.
3. She picks α and $s_i, i = 0, 1, \ldots, n-1, i \neq j$, randomly from $\mathbb{Z}_l^+$. Note that s_j has not been chosen.
4. She computes points $L_j = \alpha G$, $R_j = \alpha H_p(P_j)$, and integer $c_{j+1} = H_s(\mathcal{P}, m, L_j, R_j)$ where $H_s : \{0,1\}^* \mapsto \mathbb{Z}_l^+$ is a hash function.
5. Increasing j modulo n, she computes points and integers

$$\begin{aligned}
L_{j+1} &= s_{j+1}G + c_{j+1}P_{j+1}, \\
R_{j+1} &= s_{j+1}H_p(P_{j+1}) + c_{j+1}I, \\
c_{j+2} &= H_s(\mathcal{P}, m, L_{j+1}, R_{j+1}), \\
&\vdots \\
L_{j-1} &= s_{j-1}G + c_{j-1}P_{j-1}, \\
R_{j-1} &= s_{j-1}H_p(P_{j-1}) + c_{j-1}I, \\
c_j &= H_s(\mathcal{P}, m, L_{j-1}, R_{j-1}).
\end{aligned}$$

6. Finally, she computes $s_j = \alpha - c_j x_j$. As L_j and R_j were computed using α in step 4, this implies that

$$\begin{aligned} L_j &= \alpha G = (s_j + c_j x_j)G = s_j G + c_j P_j, \\ R_j &= \alpha H_p(P_j) = (s_j + c_j x_j) H_p(P_j) \\ &= s_j H_p(P_j) + c_j I. \end{aligned}$$

7. The linkable ring signature on the message m is given by $\sigma = (I, c_0, s_0, s_1, \ldots, s_{n-1})$.

Alice includes the linkable ring signature in a transaction spending funds in the address P and broadcasts it onto the network for inclusion in the blockchain. The message m is the hash of the transaction prefix which consists all the transaction data except for the signatures. The verification of the linkable ring signature proceeds as follows:

1. The message m which was signed is recreated from the transaction prefix.
2. The one-time addresses $\mathcal{P} = \{P_0, P_1, \ldots, P_{n-1}\}$ used to create the linkable ring signature are read from the transaction.
3. Using σ, the integers $c_j, j = 1, 2, \ldots, n-1$, are calculated as

$$\begin{aligned} L_0 &= s_0 G + c_0 P_0, \\ R_0 &= s_0 H_p(P_0) + c_0 I, \\ c_1 &= H_s(\mathcal{P}, m, L_0, R_0), \\ &\vdots \\ L_{n-2} &= s_{n-2} G + c_{n-2} P_{n-2}, \\ R_{n-2} &= s_{n-2} H_p(P_{n-2}) + c_{n-2} I, \\ c_{n-1} &= H_s(\mathcal{P}, m, L_{n-2}, R_{n-2}). \end{aligned}$$

4. Finally, c_{n-1} and s_{n-1} are used to calculate c_0' as

$$\begin{aligned} L_{n-1} &= s_{n-1} G + c_{n-1} P_{n-1}, \\ R_{n-1} &= s_{n-1} H_p(P_{n-1}) + c_{n-1} I, \\ c_0' &= H_s(\mathcal{P}, m, L_{n-1}, R_{n-1}). \end{aligned}$$

5. The signature σ is accepted if c_0' equals the c_0 given in σ. Otherwise, it is rejected.

A valid linkable ring signature is a proof that the transaction was created by someone with knowledge of a private key among the n private keys corresponding to the addresses in $\mathcal{P}$. As the one-time address P_j which is spent in the transaction is not revealed, Alice can potentially try to double spend from it. But the second transaction's linkable ring signature σ will contain the same key image $I = x_j H_p(P_j)$ leading to its rejection by the network. This justifies the use of linkable ring signatures instead of regular ring signatures.

C. Pedersen Commitments and Range Proofs

In the current implementation of Monero, transaction amounts are hidden using Pedersen commitments [6]. A point $H \in \mathcal{E}$ was generated from the Keccak hash of the base point G to ensure that the discrete logarithm of H with respect to G is unknown. The Pedersen commitment to an amount $a \in \mathbb{Z}_l$ is given by

$$C(y, a) = yG + aH,$$

where $y \in \mathbb{Z}_l$ is a randomly chosen blinding factor. This commitment scheme is perfectly hiding as $C(y, a)$ is indistinguishable from a random element in $\mathcal{E}$ even to computationally unbounded adversaries. It is computationally binding as an adversary capable of computing the discrete logarithm of H with respect to G can generate a pair $(y', a') \neq (y, a)$ such that $C(y', a') = C(y, a)$. Pedersen commitments to amounts can be added (without knowing the blinding factors) to generate a commitment to the sum of the amounts, i.e.

$$C(y_1, a_1) + C(y_2, a_2) = C(y_1 + y_2, a_1 + a_2).$$

A key feature of Pedersen commitments is that digital signatures can be used to show that a commitment is hiding the zero amount without revealing the blinding factor. Note that a commitment to the zero amount $C(y, 0) = yG$

can be viewed as a public key whose corresponding private key is the blinding factor y. If an ECDSA signature is generated using the private key y, its validity can be verified using the public key $C(y, 0)$. If the commitment had been to a non-zero amount a, then $C(y, a)$ will contain a contribution from the point H. Since the discrete logarithm of H with respect to G is unknown, a computationally bounded adversary cannot compute the private key y' such that $y'G = C(y, a) = yG + aH$.

In a transaction, the sum of the input amounts should be equal to the sum of the output amounts and the transaction fees. This relation needs to be verifiable by miners without revealing the blinding factors used to generate the amount commitments. For simplicity, assume that a transaction has one input and two outputs. Let a_{in} be the input amount, $a^1_{\text{out}}, a^2_{\text{out}}$ be the output amounts, and f be the transaction fees. These amounts satisfy the relation

$$a_{\text{in}} = a^1_{\text{out}} + a^2_{\text{out}} + f.$$

The commitment to the input amount $C(y_{\text{in}}, a_{\text{in}})$ will be recorded in the blockchain with the blinding factor y_{in} known to the owner of the input. The input owner will randomly choose blinding factors $y^1_{\text{out}}, y^2_{\text{out}}$ and create the output commitments

$$C(y^1_{\text{out}}, a^1_{\text{out}}) = y^1_{\text{out}}G + a^1_{\text{out}}H,$$
$$C(y^2_{\text{out}}, a^2_{\text{out}}) = y^2_{\text{out}}G + a^2_{\text{out}}H.$$

The transaction will contain $C(y^1_{\text{out}}, a^1_{\text{out}})$, $C(y^2_{\text{out}}, a^2_{\text{out}})$, and the transaction fees f. It will also contain an ECDSA signature verifiable by the public key

$$\begin{aligned} &C(y_{\text{in}}, a_{\text{in}}) - C(y^1_{\text{out}}, a^1_{\text{out}}) - C(y^2_{\text{out}}, a^2_{\text{out}}) - fH \\ &= \left(y_{\text{in}} - y^1_{\text{out}} - y^2_{\text{out}}\right) G + \left(a_{\text{in}} - a^1_{\text{out}} - a^2_{\text{out}} - f\right) H \\ &= zG + 0H = C(z, 0) \end{aligned}$$

where the input owner knows the private key z. By calculating the public key $C(z, 0)$ and performing ECDSA signature verification, the miners are convinced that the difference between the commitments and the fees term is a commitment to zero.

The receiver of the outputs in a transaction needs to know the blinding factors and amounts in order to verify receipt and subsequently spend the received funds. These are securely communicated to the receiver using the same shared secret $rA_{\text{vk}} = a_{\text{vk}}R$ which was used to generate the one-time address. To communicate a blinding factor y_{out} and amount a_{out}, the sender stores $y_{\text{out}} \oplus H_K(rA_{\text{vk}})$ and $a_{\text{out}} \oplus H_K(H_K(rA_{\text{vk}}))$ in the transaction where $\oplus$ denote bitwise XOR and H_K is the Keccak hash function. As the point R is contained in the transaction, the receiver can use a_{vk} to recover the blinding factor and amount.

As lG is the identity of the group $\mathcal{E}$, we have $C(y, l + a) = C(y, a)$. To prevent this relation from being exploited to inflate the amount stored in an input commitment, *range proofs* are used to prove that the amounts in a commitment are in the range $\{0, 1, \ldots, 2^{64} - 1\}$. This particular range is of interest since the maximum number of piconeros (the smallest unit of currency in Monero) which can come into existence is limited to $2^{64} - 1$. Since $l > 2^{252}$, a range proof will exclude the possibility of a commitment being interpreted as hiding an amount $l + a$.

IV. MProve Proof of Assets Protocol

The main obstacle to using the proof of assets protocol proposed in Provisions [5] for Monero exchanges stems from a fundamental difference between unspent transaction outputs in Bitcoin and Monero. In Bitcoin, the unspent outputs can be identified by reading the blockchain. In Monero, ring signatures are used to hide the specific one-time address which is being spent in a transaction. Each Monero transaction specifies an anonymity set of one-time addresses which contains the address being spent. It only contains the key image of the address being spent to avoid double spending. So the set of unspent outputs cannot be identified by reading the Monero blockchain. This prevents us from using the Provisions proof of assets protocol which begins by assembling a set of unspent outputs which contain the exchange-owned outputs as a subset.

A fundamental requirement of any proof of assets protocol for Monero is that it should, explicitly or implicitly, reveal the key images of the exchange-owned one-time addresses which contribute to the total assets commitment C_{assets}. If this requirement is not met, then we cannot detect the usage of already spent one-time addresses to

generate C_{assets}. Protocols which explicitly reveal the key images have the benefit of providing an *automatic proof of non-collusion* as one-time addresses shared between exchanges will be revealed. This proof of non-collusion does not suffer the drawback of revealing the number of one-time addresses owned by an exchange, unlike the optional proof of non-collusion proposed in the Provisions paper. But a drawback on explicitly revealing the key images is that a future transaction spending from an exchange-owned address will reveal that it was owned by the exchange (see Section IV-F). MProve suffers from this drawback and removing it is an open problem for now.

A. Ring Signatures

Our proposed proof of assets protocol uses both linkable and regular ring signatures. While the regular ring signature creation and verification algorithms (as defined in [12]) are similar to their linkable counterparts, we present them here for clarity.

Suppose Alice wants to sign a message m using a ring signature involving the public keys $\mathcal{P} = (P_0, P_1, \ldots, P_{n-1})$ where she knows the private key x_j corresponding to P_j. She creates the ring signature as follows:

1. She picks α and $s_i, i = 0, 1, \ldots, n-1, i \neq j$, randomly from $\mathbb{Z}_l^+$.
2. She computes points $L_j = \alpha G$ and integer $c_{j+1} = H_s(\mathcal{P}, m, L_j)$ where $H_s : \{0,1\}^* \mapsto \mathbb{Z}_l^+$ is a hash function.
3. Increasing j modulo n, she computes points and integers

$$\begin{aligned} L_{j+1} &= s_{j+1}G + c_{j+1}P_{j+1}, \\ c_{j+2} &= H_s(\mathcal{P}, m, L_{j+1}), \\ &\vdots \\ L_{j-1} &= s_{j-1}G + c_{j-1}P_{j-1}, \\ c_j &= H_s(\mathcal{P}, m, L_{j-1}). \end{aligned}$$

4. Finally, she computes $s_j = \alpha - c_j x_j$. As L_j was computed using α in step 2, this implies that

$$L_j = \alpha G = (s_j + c_j x_j)G = s_j G + c_j P_j.$$

5. The ring signature on the message m is given by $\gamma = (c_0, s_0, s_1, \ldots, s_{n-1})$.

To check the validity of a ring signature, a verifier does the following:

1. Using the public key list $\mathcal{P} = (P_0, P_1, \ldots, P_{n-1})$, the message m, and the ring signature $\gamma = (c_0, s_0, s_1, \ldots, s_{n-1})$, the verifier calculates the integers $c_j, j = 1, 2, \ldots, n-1$, as

$$\begin{aligned} L_0 &= s_0 G + c_0 P_0, \\ c_1 &= H_s(\mathcal{P}, m, L_0), \\ &\vdots \\ L_{n-2} &= s_{n-2}G + c_{n-2}P_{n-2}, \\ c_{n-1} &= H_s(\mathcal{P}, m, L_{n-2}). \end{aligned}$$

2. Finally, c_{n-1} and s_{n-1} are used to calculate c_0' as

$$\begin{aligned} L_{n-1} &= s_{n-1}G + c_{n-1}P_{n-1}, \\ c_0' &= H_s(\mathcal{P}, m, L_{n-1}). \end{aligned}$$

3. The signature γ is accepted if c_0' equals the c_0 given in γ. Otherwise, it is rejected.

B. Proof Generation

The MProve proof of assets protocol proceeds as follows:

1. The exchange chooses a list of one-time addresses $\mathcal{P} = (P_1, P_2, \ldots, P_N)$ from the Monero blockchain such that it knows the private keys corresponding to a subset[1] $\mathcal{P}_{\text{known}}$ of $\mathcal{P}$. The list $\mathcal{P}$ is made public by the exchange.

[1]Even though $\mathcal{P}$ is a list, we will sometimes find it convenient to interpret it as a set.

2. For each $P_i \in \mathcal{P}$, the exchange can read the corresponding Pedersen commitment C_i from the blockchain. Let C_i be the commitment to an amount a_i with blinding factor y_i, i.e.

$$C_i = C(y_i, a_i) = y_i G + a_i H. \tag{1}$$

   For $P_i \in \mathcal{P}_{\text{known}}$, the exchange knows y_i and a_i. For $P_i \notin \mathcal{P}_{\text{known}}$, the exchange may know y_i and a_i if it was the party which sent funds to P_i. In general, the exchange will not know y_i and a_i for $P_i \notin \mathcal{P}_{\text{known}}$.
3. For each $P_i \in \mathcal{P}$, the exchange randomly picks $z_i \in \mathbb{Z}_l$ and generates C'_i as

$$C'_i = \begin{cases} z_i G & \text{if } P_i \in \mathcal{P}_{\text{known}}, \\ z_i G + C_i & \text{if } P_i \notin \mathcal{P}_{\text{known}}. \end{cases} \tag{2}$$

4. For each $i = 1, 2, \ldots, N$, the exchange publishes a regular ring signature γ_i on a message m verifiable by the pair of public keys $(C'_i, C'_i - C_i)$. The calculation of γ_i is described in Appendix A.
5. For each $i = 1, 2, \ldots, N$, the exchange publishes a linkable ring signature σ_i on a message m verifiable by the pair of public keys $(P_i, C'_i - C_i)$. The calculation of σ_i is described in Appendix B.
6. The exchange publishes a commitment C_{assets} which satisfies the equation

$$\sum_{i=1}^{N} C_i = C_{\text{assets}} + \sum_{i=1}^{N} C'_i. \tag{3}$$

   The exchange claims that C_{assets} is a Pedersen commitment to the amount of Monero it owns.

The intuition behind the protocol construction is as follows. Let $I_{\text{known}} = \{i \mid P_i \in \mathcal{P}_{\text{known}}\}$ be the set of indices i such that the exchange knows the private key corresponding to P_i. The left hand side of (3) is a commitment to the amount $\sum_{i=1}^{N} a_i$. Ideally, we want the C_{assets} term on the right hand side of (3) to be a commitment to $\sum_{i \in I_{\text{known}}} a_i$ and the $\sum_{i=1}^{N} C'_i$ term to be a commitment to the remaining amount.

If $P_i \notin \mathcal{P}_{\text{known}}$, then the exchange does not know the private key corresponding to P_i. To create the ring signature σ_i, the exchange has to then use the private key z_i where $C'_i - C_i = z_i G$. This implies that $C'_i - C_i$ is a commitment to zero whenever $P_i \notin \mathcal{P}_{\text{known}}$. So the commitments C_i and C'_i on both sides of (3) commit to the same amounts whenever $P_i \notin \mathcal{P}_{\text{known}}$ and there is no transfer of funds (in the form of the $a_i H$ terms) from the corresponding C_i terms to the C_{assets} term.

If there were no constraint on C'_i for $P_i \in \mathcal{P}_{\text{known}}$, then an exchange can inflate the amount committed in C_{assets} using a single such C'_i. For example, suppose $P_1 \in \mathcal{P}_{\text{known}}$. Then the exchange can set

$$\begin{aligned} C'_1 &= z_1 G + C_1 + (l - b)H, \\ C'_i &= z_i G + C_i \qquad \text{for } i = 2, 3, \ldots, N, \\ C_{\text{assets}} &= -\sum_{i=1}^{N} z_i G + bH, \end{aligned} \tag{4}$$

and still satisfy the equation in (3) for some arbitrary amount $b \in \mathbb{Z}_l$. We could force every C'_i for $i \in I_{\text{known}}$ to be commitment to the zero amount by requiring a signature verifiable by the public key C'_i. While this would ensure the transfer of funds (in the form of the $a_i H$ terms) from C_i to the C_{assets} term in (3), it would also reveal the set $\mathcal{P}_{\text{known}}$. To avoid this, we require a regular ring signature γ_i verifiable by the pair of public keys $(C'_i, C'_i - C_i)$ for all $i = 1, 2, \ldots, N$. For $i \notin I_{\text{known}}$, γ_i can be generated using the same private key (corresponding to $C'_i - C_i$) which was used to generate σ_i. For $i \in I_{\text{known}}$, if the private key corresponding to $C'_i - C_i$ is used to generate γ_i then C'_i and C_i commit to the same amount. Consequently, C'_i cannot contain any H term beyond that contributed by C_i and cannot be used to inflate the C_{assets} term as illustrated in (4). On the other hand, if for some $i \in I_{\text{known}}$ the private key corresponding to C'_i is used to generate γ_i then this implies that C'_i is a commitment to zero ensuring that the $a_i H$ terms from C_i are included in C_{assets}.

While the above discussion considers a potentially malicious exchange, the following theorem assures us that an honest exchange can correctly generate a commitment to its assets.

Theorem 1. *If an exchange follows the MProve protocol honestly, then C_{assets} will be a commitment to the amount*

$$a_{owned} = \sum_{i \in I_{known}} a_i. \tag{5}$$

Proof: Consider the definition of C'_i given in (2).

- If $P_i \in \mathcal{P}_{\text{known}}$, C'_i is a commitment to zero. Hence $\sum_{i \in I_{\text{known}}} C'_i$ is a commitment to the zero amount.
- If $P_i \notin \mathcal{P}_{\text{known}}$, $C'_i - C_i$ is a commitment to zero. Let $I_{\text{unknown}} = \{i \mid 1 \leq i \leq N, P_i \notin \mathcal{P}_{\text{known}}\}$ denote the set of indices i such that the exchange does not know the private key corresponding to P_i. Then $\sum_{i \in I_{\text{unknown}}} (C'_i - C_i)$ is a commitment to the zero amount.

Rearranging (3), we get

$$\sum_{i \in I_{\text{known}}} C_i = C_{\text{assets}} + \sum_{i \in I_{\text{known}}} C'_i + \sum_{i \in I_{\text{unknown}}} (C'_i - C_i). \tag{6}$$

As the last two sums on the right hand side are commitments to zero, C_{assets} and $\sum_{i \in I_{\text{known}}} C_i$ must be commitments to the same amount. Since

$$\begin{aligned} \sum_{i \in I_{\text{known}}} C_i &= \sum_{i \in I_{\text{known}}} (y_i G + a_i H) \\ &= a_{\text{owned}} H + \sum_{i \in I_{\text{known}}} y_i G, \end{aligned} \tag{7}$$

C_{assets} is a commitment to a_{owned}. ■

C. Proof Verification

The output of an exchange in the MProve protocol consists of the following:

- A list of one-time addresses $P_1, P_2, \ldots, P_N$.
- The commitments $C'_1, C'_2, \ldots, C'_N$ created by the exchange.
- The regular ring signatures $\gamma_i = (d_0^i, t_0^i, t_1^i)$ for $i = 1, 2, \ldots, N$.
- The linkable ring signatures $\sigma_i = (I_i, c_0^i, s_0^i, s_1^i)$ for $i = 1, 2, \ldots, N$.
- The message m used to create γ_i and σ_i.
- The commitment C_{assets} which the exchange claims to be a commitment to its total assets.

Verification involves the following operations:

1. The verifier checks that none of the key images I_i published by the exchange as part of the signatures σ_i appear in any of the transactions in the blockchain. If a key image appears in a transaction, the verifier rejects the proof of assets as it implies that the funds in the corresponding one-time address P_i have already been spent. If none of the key images have appeared on the blockchain, the verifier continues with proof verification.
2. The verifier reads the commitments C_i corresponding to the P_is from the blockchain.
3. The public key $C'_i - C_i$ is computed for each i.
4. The public key pair $(C'_i, C'_i - C_i)$ is used to verify the regular ring signatures γ_i.
5. The public key pair $(P_i, C'_i - C_i)$ is used to verify the linkable ring signatures σ_i.
6. Equality in (3) is verified using the C_is, C'_is, and C_{assets}.
7. The verifier also checks that none of the key images I_i published by the exchange appear in the signatures published by other exchanges. If a key image is common to the signatures published by two different exchanges, collusion is declared.

As observed in [5], the exchanges need to generate their proofs after the same block for collusion to be detectable.

D. Security Properties

We consider two security properties of the MProve proof of assets protocol: *inflation resistance* and *address privacy*. We will model computationally bounded entities as probabilistic polynomial-time (PPT) algorithms. The inflation resistance property prevents a PPT exchange from creating a commitment to an amount which is greater than its total assets. The address privacy property prevents a PPT adversary from identifying the addresses owned by the exchange from the information provided as output of the proof of assets.

Theorem 2. *Suppose an exchange can create a proof of assets with commitment $C_{assets} = C(y, a)$ such that*

(i) it knows the blinding factor $y \in \mathbb{Z}_l$ and amount $a \in \mathbb{Z}_l$,

(ii) the amount a is greater than a_{owned} defined in (5), and

(iii) the proof is accepted by the verification procedure in Section IV-C.
Then the exchange can calculate the discrete logarithm of H with respect to G with overwhelming probability.

Proof: As the exchange was successful in creating the linkable ring signatures σ_i, it knows $w_i \in \mathbb{Z}_l$ such that $w_i G = C'_i - C_i$ for $i \in I_{\text{unknown}}$, i.e. $P_i \notin \mathcal{P}_{\text{known}}$, with overwhelming probability. This is because creating σ_i without knowing either of the private keys corresponding to the public keys $\{P_i, C'_i - C_i\}$ amounts to a forgery of the linkable ring signature which is possible only with negligible probability.

As the exchange was successful in creating the regular ring signatures γ_i, it knows $w_i \in \mathbb{Z}_l$ such that $w_i G$ is equal to either C'_i or $C'_i - C_i$ for $i \in I_{\text{known}}$, i.e. $P_i \in \mathcal{P}_{\text{known}}$, with overwhelming probability. Let I_1, I_2 be a partition of the set I_{known} such that $w_i G = C'_i$ for $i \in I_1$ and $w_i G = C'_i - C_i$ for $i \in I_2$.

As $I_{\text{unknown}} \cup I_1 \cup I_2 = \{1, 2, \ldots, N\}$, we can rearrange the terms in (3) to get

$$\begin{aligned}\sum_{i \in I_1} C_i &= C_{\text{assets}} + \sum_{i \in I_1} C'_i + \sum_{i \in I_{\text{unknown}} \cup I_2} (C'_i - C_i) \\ &= C_{\text{assets}} + \sum_{i=1}^{N} w_i G. \end{aligned} \tag{8}$$

Since $I_1 \subseteq I_{\text{known}}$, the exchange knows the blinding factors y_i and amounts a_i such that $C_i = y_i G + a_i H$ for all $i \in I_1$. By the theorem hypothesis, it also knows y and a such that $C_{\text{assets}} = yG + aH$. Substituting these values in (8), we get

$$\sum_{i \in I_1} (y_i G + a_i H) = yG + aH + \sum_{i=1}^{N} w_i G \tag{9}$$

$$\implies \left(\sum_{i \in I_1} y_i - \sum_{i=1}^{N} w_i - y \right) G = \left(a - \sum_{i \in I_1} a_i \right) H \tag{10}$$

The exchange can then calculate the discrete logarithm of H with respect to G as

$$\left(a - \sum_{i \in I_1} a_i \right)^{-1} \left(\sum_{i \in I_1} y_i - \sum_{i=1}^{N} w_i - y \right). \tag{11}$$

The multiplicative inverse of $a - \sum_{i \in I_1} a_i$ exists because it is a non-zero element in the prime field $\mathbb{F}_l$. This follows from the assumption that $a > a_{\text{owned}} = \sum_{i \in I_{\text{known}}} a_i \geq \sum_{i \in I_1} a_i$. ■

If we assume that a PPT entity cannot calculate the discrete logarithm of H except with negligible probability, Theorem 2 assures us that a PPT exchange cannot use MProve to output a commitment to an amount which is greater than the total assets it owns.

We define the address privacy of the MProve protocol using the following experiment (which we call `AddrPriv`). Let $\mathbf{C}' = (C'_1, C'_2, \ldots, C'_N)$, $\mathbf{\Gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_N)$, $\mathbf{\Sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_N)$ be vectors containing the commitments and signatures output by the MProve protocol. Let $\mathbf{C} = (C_1, C_2, \ldots, C_N)$ be the commitments corresponding to the list of one-time addresses in $\mathcal{P}$. Let m be the message which is signed to create the signatures. The value of C_{assets} is determined by $\mathbf{C}$ and $\mathbf{C}'$ according to (3).

1. The exchange chooses an index j such that $P_j \in \mathcal{P}$ is a one-time address whose private key is known to it.
2. The exchange chooses a bit b uniformly from $\{0, 1\}$.
3. If $b = 0$, the exchange uses the MProve protocol to create a proof of assets where the linkable ring signature σ_j is created using the private key corresponding to P_j and the ring signature γ_j is created using the private key corresponding to C'_j.
4. If $b = 1$, the exchange uses the MProve protocol to create a proof of assets where both the linkable ring signature σ_j and ring signature γ_j are created using the private key corresponding to $C'_j - C_j$.
5. Given $\mathcal{P}, \mathbf{C}, \mathbf{C}', \mathbf{\Gamma}, \mathbf{\Sigma}, m$, and index j, an adversary $\mathcal{A}$ outputs a bit b', i.e.
$$b' = \mathcal{A}\left(\mathcal{P}, \mathbf{C}, \mathbf{C}', \mathbf{\Gamma}, \mathbf{\Sigma}, m, j\right). \tag{12}$$
6. The adversary succeeds if $b' = b$. Otherwise, it fails.

TABLE I
MPROVE PROOF GENERATION AND VERIFICATION PERFORMANCE

$\lvert\mathcal{P}\rvert$	$\lvert\mathcal{P}_{\text{known}}\rvert$	Proof Size	Gen. Time	Ver. Time
1000	100	0.32 MB	0.70 s	0.65 s
1000	500	0.32 MB	0.69 s	0.69 s
1000	900	0.32 MB	0.68 s	0.67 s
10000	1000	3.2 MB	7.01 s	6.76 s
10000	5000	3.2 MB	6.92 s	6.76 s
10000	9000	3.2 MB	6.87 s	6.75 s
100000	10000	32 MB	71.79 s	67.85 s
100000	50000	32 MB	71.13 s	67.83 s
100000	90000	32 MB	70.39 s	67.82 s

Definition 1. *The MProve protocol provides address privacy if every PPT adversary $\mathcal{A}$ succeeds in the* `AddrPriv` *experiment with a probability which is negligibly close to $\frac{1}{2}$, irrespective of the message m and index j.*

As the C'_is in the MProve protocol are chosen randomly and independently of each other (see definition in (2)), the signatures γ_i for $i \neq j$ do not aid the adversary in estimating b. Similarly, the signatures σ_i created by one of the private keys corresponding to the public key pair $(P_i, C'_i - C_i)$ do not aid in the estimation of b for $i \neq j$. Hence, we can restrict our attention to adversaries of the form $\mathcal{A}(P_j, C_j, C'_j, \gamma_j, \sigma_j, m)$.

A ring signature scheme is said to provide *signer ambiguity* if it does not reveal the identity of the signing key except with probability negligibly close to that achievable by random guessing [8]. The linkable ring signature scheme used to generate σ_j has been shown to be signer ambiguous in the random oracle model provided the decisional Diffie-Hellman (DDH) problem is hard [13, Appendix C]. A similar argument can be used to show that the MProve protocol provides address privacy in the random oracle model. We state the following theorem whose proof is given in Appendix C.

Theorem 3. *The MProve protocol provides address privacy in the random oracle model under the decisional Diffie-Hellman assumption.*

E. Performance

The performance (on a 3.6 GHz CPU/8 GB RAM desktop PC) of the MProve proof generation and verification algorithms is given in Table I for $\mathcal{P}$ having sizes 1000, 10000, and 100000. For each case, the percentage of known addresses is either 10%, 50%, or 90%. Both running times and proof sizes increase linearly with the size of $\mathcal{P}$ with the proof generation time having a small dependence on the size of $\mathcal{P}_{\text{known}}$. The simulation code can be found at [14].

F. Drawback

The main drawback of the MProve protocol is that it does not preserve sender address privacy when an exchange spends from an address $P_i \in \mathcal{P}_{\text{known}}$ which was used in the protocol. This is because the key image I_i is revealed in the MProve protocol which can be matched with the key image which appears in the transaction spending from P_i. This does not affect the privacy of the MProve protocol itself as the transaction appears on the blockchain after the proof of assets is generated. But it effectively makes the transaction a zero mix-in transaction which increases traceability [15], [16]. Alleviating this problem is an interesting direction for future research.

V. APPLICATION TO BITCOIN EXCHANGES

While MProve is intended for Monero exchanges, it can also be used for Bitcoin exchanges for taking advantage of the proof of non-collusion. As the unspent amount a_i associated with a public key P_i is known, the commitment C_i can be simply set to $a_i H$. Here H needs to be a group element whose discrete logarithm with respect to the base point of the secp256k1 curve is not known. The C'_i commitments will be generated as in (2). As the z_is are randomly chosen, knowledge of the opening of C_i does not reveal whether $P_i \in \mathcal{P}_{\text{known}}$ or not. The ring signatures

γ_i and linkable ring signatures σ_i are generated as in the MProve protocol. The commitment C_{assets} published by the exchange has to satisfy

$$\sum_{i=1}^{N} a_i H = C_{\text{assets}} + \sum_{i=1}^{N} C'_i. \tag{13}$$

Collusion between exchanges can be detected by checking if any of the key images I_i in the linkable ring signatues σ_i appear in the proofs provided by two different exchanges. As spending from a Bitcoin address does not involve the generation of a key image, the drawback mentioned in Section IV-F does not apply.

APPENDIX A
RING SIGNATURE GENERATION IN MPROVE

In this appendix, we describe the calculation of the regular ring signature γ_i corresponding to step 4 of the MProve proof generation procedure described in Section IV-B. The calculation is the same as the algorithm described in Section IV-A with the public key list $\mathcal{Q}^i = (C'_i, C'_i - C_i)$. We use different notation to differentiate the terms from those used in the linkable ring signature calculation of Appendix B.

(i) For i such that $P_i \in P_{\text{known}}$, the private key z_i corresponding to the public key $C'_i = z_i G$ is used to create the regular ring signature $\gamma_i = (d_0^i, t_0^i, t_1^i)$ where

- Using randomly chosen β_i from $\mathbb{Z}_l^+$, d_1^i is calculated as

$$\begin{aligned} S_0^i &= \beta_i G, \\ d_1^i &= H_s(\mathcal{Q}^i, m, S_0^i). \end{aligned} \tag{14}$$

- Using randomly chosen t_1^i from $\mathbb{Z}_l^+$, d_0^i is calculated as

$$\begin{aligned} S_1^i &= t_1^i G + d_1^i (C'_i - C_i), \\ d_0^i &= H_s(\mathcal{Q}^i, m, S_1^i). \end{aligned} \tag{15}$$

- The value t_0^i is set to $\beta_i - d_0^i z_i$.

(ii) For i such that $P_i \notin P_{\text{known}}$, the private key z_i corresponding to the public key $C'_i - C_i = z_i G$ is used to create the regular ring signature $\gamma_i = (d_0^i, t_0^i, t_1^i)$ where

- Using randomly chosen β_i from $\mathbb{Z}_l^+$, d_0^i is calculated as

$$\begin{aligned} S_1^i &= \beta_i G, \\ d_0^i &= H_s(\mathcal{Q}^i, m, S_1^i). \end{aligned} \tag{16}$$

- Using randomly chosen t_0^i from $\mathbb{Z}_l^+$, d_1^i is calculated as

$$\begin{aligned} S_0^i &= t_0^i G + d_0^i C'_i, \\ d_1^i &= H_s(\mathcal{Q}^i, m, S_0^i). \end{aligned} \tag{17}$$

- The value t_1^i is set to $\beta_i - d_1^i z_i$.

APPENDIX B
LINKABLE RING SIGNATURE GENERATION IN MPROVE

In this appendix, we describe the calculation of the linkable ring signature σ_i corresponding to step 5 of the MProve proof generation procedure described in Section IV-B. The calculation is the same as the algorithm described in Section III-B with the public key list $\mathcal{R}^i = (P_i, C'_i - C_i)$.

Let $x_i \in \mathbb{Z}_l^+$ be the private key corresponding to P_i, i.e. $P_i = x_i G$.

(i) For i such that $P_i \in P_{\text{known}}$, the private key x_i is used to create the linkable ring signature $\sigma_i = (I_i, c_0^i, s_0^i, s_1^i)$ where $I_i = x_i H_p(P_i)$ is the key image.

- Using randomly chosen α_i from $\mathbb{Z}_l^+$, c_1^i is calculated as

$$\begin{aligned} L_0^i &= \alpha_i G, \\ R_0^i &= \alpha_i H_p(P_i), \\ c_1^i &= H_s(\mathcal{R}^i, m, L_0^i, R_0^i). \end{aligned} \tag{18}$$

- Using randomly chosen s_1^i from $\mathbb{Z}_l^+$, c_0^i is calculated as

$$\begin{aligned} L_1^i &= s_1^i G + c_1^i (C_i' - C_i), \\ R_1^i &= s_1^i H_p(C_i' - C_i) + c_1^i I_i, \\ c_0^i &= H_s(\mathcal{R}^i, m, L_1^i, R_1^i). \end{aligned} \tag{19}$$

- The value s_0^i is set to $\alpha_i - c_0^i x_i$.

(ii) For i such that $P_i \notin P_{\text{known}}$, the private key z_i corresponding to the public key $C_i' - C_i = z_i G$ is used to create the linkable ring signature $\sigma_i = (I_i, c_0^i, s_0^i, s_1^i)$ where $I_i = z_i H_p(C_i' - C_i)$ is the key image.

- Using randomly chosen α_i from $\mathbb{Z}_l^+$, c_0^i is calculated as

$$\begin{aligned} L_1^i &= \alpha_i G, \\ R_1^i &= \alpha_i H_p(C_i' - C_i), \\ c_0^i &= H_s(\mathcal{R}^i, m, L_1^i, R_1^i). \end{aligned} \tag{20}$$

- Using randomly chosen s_0^i from $\mathbb{Z}_l^+$, c_1^i is calculated as

$$\begin{aligned} L_0^i &= s_0^i G + c_0^i P_i, \\ R_0^i &= s_0^i H_p(P_i) + c_0^i I_i, \\ c_1^i &= H_s(\mathcal{R}^i, m, L_0^i, R_0^i). \end{aligned} \tag{21}$$

- The value s_1^i is set to $\alpha_i - c_1^i x_i$.

APPENDIX C
PROOF OF THEOREM 3

The proof involves constructing a PPT adversary $\mathcal{M}$ who violates the decisional Diffie-Hellman (DDH) assumption using a PPT adversary $\mathcal{A}$ who violates address privacy in the MProve protocol. Furthermore, the hash functions H_s and H_p will be modeled as random oracles.

Suppose $\mathcal{A}$ is a PPT adversary who violates address privacy in the `AddrPriv` experiment. By the discussion following Definition 1, we can restrict our attention to adversaries of the form $\mathcal{A}(P_j, C_j, C_j', \gamma_j, \sigma_j, m)$. If $\mathcal{A}$ violates address privacy, then there exists a polynomial f such that

$$\Pr\left[\mathcal{A}(P_j, C_j, C_j', \gamma_j, \sigma_j, m) = b\right] > \frac{1}{2} + \frac{1}{f(\lambda)}, \tag{22}$$

where λ is a security parameter.

Suppose $\mathcal{M}$ is an adversary tasked with identifying Diffie-Hellman triples. Let G be the generator of a cyclic group of order l. An entity who wishes to test $\mathcal{M}$ chooses a bit d uniformly from $\{0, 1\}$ and gives $\mathcal{M}$ the triple $(G_1, G_2, G_3) = (aG, bG, c_d G)$ where a, b, c_0 are chosen uniformly from $\mathbb{Z}_l^+$ and $c_1 = ab$. To estimate d from (G_1, G_2, G_3), $\mathcal{M}$ will use $\mathcal{A}$ as a subroutine as follows.

1. $\mathcal{M}$ chooses a bit b uniformly from $\{0, 1\}$.
2. If $b = 0$, $\mathcal{M}$ does the following:
   (i) It sets $P_j = G_1$, $H_p(P_j) = G_2$ (using the random oracle assumption), and $I_j = G_3$.
   (ii) It chooses random group elements C_j and C_j' and sets the public key list $\mathcal{R} = (P_j, C_j' - C_j)$.

(iii) It chooses scalars c_0, s_0, s_1 uniformly from $\mathbb{Z}_l^+$ and calculates L_1, R_1 as follows.

$$\begin{aligned} L_0 &= s_0 G + c_0 P_j, \\ R_0 &= s_0 H_p(P_j) + c_0 I_j, \\ c_1 &= H_s(\mathcal{R}, m, L_0, R_0), \\ L_1 &= s_1 G + c_1 (C'_j - C_j), \\ R_1 &= s_1 H_p(C'_j - C_j) + c_1 I_j. \end{aligned} \tag{23}$$

(iv) It sets the random oracle output $H_s(\mathcal{R}, m, L_1, R_1)$ to be equal to c_0.

(v) It sets $\sigma_j = (I_j, c_0, s_0, s_1)$ which will pass the linkable ring signature verification algorithm for the public key list $\mathcal{R}$.

(vi) It chooses scalars d_0, t_0, t_1 uniformly from $\mathbb{Z}_l^+$ and calculates S_1 for public key list $\mathcal{Q} = (C'_j, C'_j - C_j)$ as follows.

$$\begin{aligned} S_0 &= t_0 G + d_0 C'_j, \\ d_1 &= H_s(\mathcal{Q}, m, S_0), \\ S_1 &= t_1 G + d_1 (C'_j - C_j), \end{aligned} \tag{24}$$

(vii) It sets the random oracle output $H_s(\mathcal{Q}, m, S_1)$ to be equal to d_0.

(viii) It sets $\gamma_j = (d_0, t_0, t_1)$ which will pass the ring signature verification algorithm for the public key list $\mathcal{Q}$.

3. If $b = 1$, $\mathcal{M}$ does the following:
   (i) It chooses a random group element C_j and sets $C'_j = C_j + G_1$.
   (ii) It sets $H_p(C'_j - C_j) = G_2$ (using the random oracle assumption) and $I_j = G_3$.
   (iii) It chooses a random group elements P_j and sets the public key list $\mathcal{R} = (P_j, C'_j - C_j)$.
   (iv) For public key list $\mathcal{R}$, it generates the linkable ring signature $\sigma_j = (I_j, c_0, s_0, s_1)$ using the same procedure as the $b = 0$ case.
   (v) For public key list $\mathcal{Q} = (C'_j, C'_j - C_j)$, it generates the ring signature $\gamma_j = (d_0, t_0, t_1)$ using the same procedure as the $b = 0$ case.
4. $\mathcal{M}$ obtains the output $b' = \mathcal{A}(P_j, C_j, C'_j, \gamma_j, \sigma_j, m)$. If $b' = b$, then $\mathcal{M}$ estimates the bit d as 1. Otherwise, $\mathcal{M}$ estimates d as 0.

When $d = 1$, the triple (G_1, G_2, G_3) is a DH triple and the I_j in the linkable ring signature σ_j constructed by $\mathcal{M}$ is the key image of one of the keys in $\mathcal{R}$. By (22), the adversary $\mathcal{A}$ can identify which key was used with a probability which is non-neglibly better than $\frac{1}{2}$. This in turn implies that $\mathcal{M}$ can correctly estimate d with the same probability. So we have

$$\begin{aligned} \Pr\left[\mathcal{M}(G_1, G_2, G_3) = 1 \mid d = 1\right] &= \Pr[b' = b \mid d = 1] \\ &> \frac{1}{2} + \frac{1}{f(\lambda)}. \end{aligned} \tag{25}$$

When $d = 0$, G_3 (and consequently I_j) is independent of (G_1, G_2). As the inputs $P_j, C_j, C'_j, \gamma_j, \sigma_j$ to $\mathcal{A}$ or their components are either uniformly random group elements/scalars or outputs of random oracles, the adversary can only estimate the bit b with probability $\frac{1}{2}$. So we have

$$\begin{aligned} \Pr\left[\mathcal{M}(G_1, G_2, G_3) = 0 \mid d = 0\right] &= \Pr[b' \neq b \mid d = 0] \\ &= \frac{1}{2}. \end{aligned} \tag{26}$$

As d was chosen uniformly from $\{0, 1\}$, averaging (25) and (26) we get

$$\Pr\left[\mathcal{M}(G_1, G_2, G_3) = d\right] > \frac{1}{2} + \frac{1}{2f(\lambda)}. \tag{27}$$

This contradicts the DDH assumption as $\mathcal{M}$ is a PPT adversary.

REFERENCES

[1] Wikipedia contributors. Mt. Gox — Wikipedia, the free encyclopedia. [Accessed 15-Nov-2018]. [Online]. Available: https://en.wikipedia.org/wiki/Mt._Gox

[2] "Cryptocurrency anti-money laundering report," 2018 Q3 Report, CipherTrace Inc, Oct. 2018. [Online]. Available: https://ciphertrace.com/crypto-aml-report-2018q3

[3] Monero website. [Online]. Available: https://getmonero.org/

[4] Z. Wilcox, "Proving your Bitcoin reserves," Bitcoin Talk Forum Post, May 2014. [Online]. Available: https://bitcointalk.org/index.php?topic=595180.0

[5] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, New York, NY, USA, 2015, pp. 720–731.

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

[7] N. v. Saberhagen, "CryptoNote v 2.0," White paper, 2013. [Online]. Available: https://cryptonote.org/whitepaper.pdf

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

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

[10] G. Maxwell, "Confidential transactions," 2015. [Online]. Available: https://people.xiph.org/~greg/confidential_values.txt

[11] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, pp. 77–89, Sep 2012.

[12] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," in *Advances in Cryptology — ASIACRYPT 2002*. Springer, 2002, pp. 415–432.

[13] J. K. Liu, V. K. Wei, and D. S. Wong, "Linkable spontaneous anonymous group signature for ad hoc groups," Cryptology ePrint Archive, Report 2004/027, 2004, https://eprint.iacr.org/2004/027.

[14] MProve simulation code. [Online]. Available: https://github.com/avras/monero/tree/v0.13.0.4-mprove/tests/mprove

[15] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of Monero's blockchain," in *European Symposium on Research in Computer Security – ESORICS 2017*, 2017, pp. 153–173.

[16] M. Möser, K. Soska, E. Heilman, K. Lee, H. Heffan, S. Srivastava, K. Hogan, J. Hennessey, A. Miller, A. Narayanan, and N. Christin, "An empirical analysis of traceability in the Monero blockchain," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 143–163, 2018.