MProve+: Privacy Enhancing Proof of Reserves Protocol for Monero

Arijit Dutta, Suyash Bagad, and Saravanan Vijayakumaran

***Abstract*—Proof of reserves protocols enable cryptocurrency exchanges to prove solvency, i.e. prove that they have enough reserves to meet their liabilities towards their customers. MProve (EuroS&PW, 2019) was the first proof of reserves protocol for Monero which provided some privacy to the exchanges' addresses. As the key images and the addresses are inherently linked in the MProve proof, an observer could easily recognize the exchange-owned address when a transaction spending from it appears on the blockchain. This is detrimental for an exchange's privacy and becomes a natural reason for exchanges to not adopt MProve. To this end, we propose MProve+, a Bulletproofs-based (S&P, 2018) NIZK protocol, which unlinks the key images and the addresses, thus alleviating the drawback of MProve. Furthermore, MProve+ presents a promising alternative to MProve due to an order of magnitude smaller proof sizes along with practical proof generation and verification times.**

***Index Terms*—Cryptocurrency, Monero, Proof of Reserves**

I. Introduction

The business of a cryptocurrency exchange is primarily built on buying coins from the miners and selling them to non-miners. They also provide custodial wallets to their customers which has a two-fold advantage to the customers. Firstly, by means of these wallets, the customers can outsource to the exchange the cumbersome task of keeping the secret keys safe without them being stolen or forgotten. Using these wallets, the customers can also trade various cryptocurrencies among themselves. These trades are fast and efficient as they are handled internally by the exchange instead of having to publish them on the blockchain. In spite of the above advantages, exchanges are risky for customers as they are prone to hacking and exit scams [1]. This leads to a loss of customers' money and raises severe concerns. To alleviate customer concerns and regain their trust, proof of reserves protocols are proposed for cryptocurrency exchanges.

A proof of reserves protocol proves that an exchange is in possession of a certain amount of cryptocurrency. For example, in 2011, the Mt. Gox cryptocurrency exchange published a transaction on the Bitcoin blockchain transferring 424,242 bitcoins from its wallets to a previously revealed Bitcoin address [2]. This transaction might be considered as a proof of reserves proving that Mt. Gox indeed possessed a certain amount of bitcoins. In 2019, Blockstream released a tool for Bitcoin exchanges which generates a transaction including all unspent transaction outputs (UTXOs) of an exchange revealing the total reserves amount [3]. It also includes an invalid input to make the transaction invalid. This is to prevent the exchange reserves from being spent. However, these techniques reveal the total reserves amount of the exchange and all the owned Bitcoin addresses. This is crucial business information which an exchange might not want to reveal.

To address the privacy concerns of exchanges, several privacy preserving proof of reserves protocols have been proposed. Decker *et al.* [4] proposed a protocol for Bitcoin exchanges which only produces a binary output indicating whether the exchange has more reserves of bitcoins than the amount of bitcoins it has sold to its customers (also called the total *liabilities*) or not. This is known as a *proof of solvency*. Although the proposed protocol was privacy preserving, it is based on a trusted platform module.

Dagher *et al.* [5] proposed a proof of solvency protocol for Bitcoin exchanges called Provisions. It was the first scheme which required no trusted setup and was based only on cryptographic assumptions. The protocol has three stages. In the first stage, a privacy preserving proof of reserves protocol generates a Pedersen commitment C_{res} to the total reserves amount of bitcoins (say a_{res}). To generate C_{res}, an anonymity set is used which contains all the exchange-owned Bitcoin addresses and some cover addresses. Thus the protocol hides the total reserves amount in a commitment and blends all the exchange-owned Bitcoin addresses in an anonymity set. The associated zero-knowledge proof proves that C_{res} is indeed a commitment to the total reserves amount a_{res}.

In the second stage, a protocol called *proof of liabilities* generates a proof from which a Pedersen commitment C_{liab} is generated. The commitment C_{liab} commits to the total amount of bitcoins that the exchange has sold to its customers (say a_{liab}). Here for each customer, the exchange publishes a commitment for each bit of the customer's amount. It also gives a proof that it knows the corresponding blinding factor and that the committed value is either 0 or 1. These proofs along with the fact that the number of such bit commitments for a particular customer is no more than 51 verify that each customer's amount lies in the range $\{0, 1, \ldots, 2^{51} - 1\}$. Here 2^{51} is a bound on the maximum number of bitcoins that could exist. This range proof is necessary to show that the exchange is not reducing its liabilities by using a very large number which acts as a negative number in modular arithmetic.

By using some information provided by the exchange, a customer can (a) secretly find out her entry in the list of total liabilities, (b) verify that the commitment allotted to her commits to the amount she paid to the exchange. She can further verify the proofs associated with the bit commitments of other customers to verify the range proofs corresponding

Arijit Dutta and Saravanan Vijayakumaran are with the Department of Electrical Engineering, Indian Institute of Technology Bombay. Suyash Bagad is with Aztec protocol. Email ids: arijit.dutta@iitb.ac.in, suyash@aztecprotocol.com, sarva@ee.iitb.ac.in

to their amounts. Multiplying the amount commitments of all customers including hers, she computes C_{liab}. Computing C_{liab} after checking the range proofs for each customer can also be done by an auditor instead of any individual customer. Finally to prove that the exchange is solvent, the exchange provides a range proof to prove that $C_{\text{res}}C_{\text{liab}}^{-1}$ commits[1] to a non-negative number.

The proof of reserves protocol in Provisions [5] is specific to Bitcoin. Motivated by the general structure of Provisions, some proof of reserves protocols for other cryptocurrencies have been proposed e.g. MProve [7] for Monero [8], Revelio [9] for Mimblewimble [10] [11], and Nummatus [12] for Quisquis [13]. All these protocols generate C_{res} i.e. a Pedersen commitment to the total reserves amount without revealing the exchange-owned addresses/accounts. This Pedersen commitment C_{res} can be used with the proof of liabilities protocol proposed by Provisions [5]. In Provisions' proof of liabilities protocol, if a customer fails to check whether her amount is accounted in C_{liab}, the exchange could possibly omit that amount, effectively reducing its liabilities. Recently, Chalkias *et al.* [14] proposed a scheme called Distributed Auditing Proofs of Liabilities (DAPOL) which addresses this concern. The authors proposed to remove the limitation of Provisions' proof of liabilities by using private information retrieval to view the inclusion proof by the customers. All the above proof of reserves protocols including MProve+ can work along with DAPOL for proof of solvency. The commitment to the total reserves can also be used in a range proof to show that the total reserves of the exchange is more than a base amount which can be estimated from the trade volume data published by the exchange [15]. If exchanges publish proofs of reserves periodically, loss of assets by an exchange can be detected early.

Our contribution. MProve [7] is a proof of reserves protocol for Monero exchanges. When a Monero exchange spends from a one-time address which was used in MProve, it is revealed that the one-time address is the source of the transaction and it belongs to the exchange. In particular, the transaction becomes a *zero-mixin*[2] transaction. This is a significant privacy limitation since it not only affects the exchange privacy but also affects the privacy of other Monero transactions. If such one-time addresses are used as cover addresses in the rings of other transactions, it not only reduces the effective anonymity of the source address in those transactions but could also lead to traceability of other inputs via the cascade effect [16], [17]. The main contributions of this paper are as follows.

(i) We propose MProve+, which removes the above mentioned drawback of MProve using techniques from Bulletproofs [18] and Omniring [6].
(ii) We give a detailed analysis of the security properties of the MProve+ protocol and how it behaves with the privacy features of the Monero scheme.
(iii) We have implemented both MProve and MProve+ in Rust and compared their performance. The simulations show that MProve+ is practical to be adopted by Monero exchanges.

The organization of the paper is as follows. Section II discusses the preliminary concepts, the MProve protocol and its drawback, and those aspects of Bulletproofs [18] and Omniring [19] using which we constructed the MProve+ protocol. In Section III, we describe the construction of the MProve+ protocol. In Section IV, we present the security properties of the MProve+ protocol and discuss how it behaves with the privacy features of the Monero scheme. Section V discusses the major contributions of the paper. Section VI gives the performance comparison of the MProve+ and MProve protocol. We draw conclusions in Section VII.

II. Background

A. Notation and Preliminary Concepts

In this paper, we consider a cyclic group $\mathbb{G}$ of prime order q with generator G where the decisional Diffie Hellman (DDH) problem is assumed to be hard. They are represented by the tuple $\mathcal{G} = (\mathbb{G}, q, G)$. All group elements are denoted by upper case letters. All scalars in $\mathbb{Z}_q$ are denoted by lower case letters. As $\mathbb{G}$ is of prime order, every non-identity element of $\mathbb{G}$ is a generator. Let $H \in \mathbb{G}$ be another random generator of $\mathbb{G}$ such that the discrete logarithm relation between G and H is not known i.e. x is not known where $H = G^x$. A Pedersen commitment [20] C to an amount a is defined as $G^y H^a$, where $y \in \mathbb{Z}_q$ is a randomly sampled blinding factor.

Let $\mathbb{G}^n$ and $\mathbb{Z}_q^n$ be the n-ary Cartesian products of sets $\mathbb{G}$ and $\mathbb{Z}_q$ respectively. Bold fonts denote vectors. Inner product of two scalar vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle := \sum_{i=1}^n a_i \cdot b_i$ where $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n)$. Hadamard and Kronecker products are defined respectively as, $\mathbf{a} \circ \mathbf{b} := (a_1 \cdot b_1, \ldots, a_n \cdot b_n) \in \mathbb{Z}_q^n$, $\mathbf{a} \otimes \mathbf{c} := (a_1\mathbf{c}, \ldots, a_n\mathbf{c}) \in \mathbb{Z}_q^{nm}$ where $\mathbf{c} \in \mathbb{Z}_q^m$. The concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted as $\mathbf{a}||\mathbf{b} := (a_1, \ldots, a_n, b_1, \ldots, b_n)$. For a base vector $\mathbf{G} = (G_1, \ldots, G_n) \in \mathbb{G}^n$, vector exponentiation is defined as $\mathbf{G}^{\mathbf{a}} := \prod_{i=1}^n G_i^{a_i} \in \mathbb{G}$. For a scalar $u \in \mathbb{Z}_q \setminus \{0\}$, we denote its consecutive powers in the form of a vector $\boldsymbol{u}^n := (1, u, u^2, \ldots, u^{n-1})$. We represent exponentiation of all components of a vector $\mathbf{a}$ by the same scalar $k \in \mathbb{Z}_q$ by $\mathbf{a}^{\circ k} := (a_1^k, a_2^k, \ldots, a_n^k)$. Hadamard inverse of a vector is defined as $\mathbf{a}^{\circ -1} := (b_1, b_2, \ldots, b_n)$ where $b_i = a_i^{-1}$ if $a_i \neq 0$ and $b_i = 1$ otherwise. If an element a is chosen uniformly from a set A, such a choice is denoted by $a \xleftarrow{\$} A$. For a positive integer N, $[N]$ denotes the set $\{1, 2, \ldots, N\}$.

B. Monero

Monero [8], based on the CryptoNote protocol [21], is a privacy focused cryptocurrency. It preserves the privacy of the receiver, the sender, and the amount in a transaction by means of three techniques, namely, *one-time addresses*, *linkable ring signatures*, and *confidential transactions*. In Monero, a user who wishes to receive funds publishes a public key pair. For

[1] In this paper we follow multiplicative notation to be consistent with Omniring [6], which motivates our protocol.

[2] A zero-mixin transaction in Monero is a transaction which does not have any decoy addresses in the ring. When a one-time address used in MProve proofs is spent, it is explicitly revealed that the address is being spent and other decoy addresses in the ring of the transaction become useless. Hence the transaction is effectively a zero-mixin transaction.

example, let $(B_{\text{vk}}, B_{\text{sk}}) \in \mathbb{G}^2$ be the public key pair of Bob. The keys B_{vk} and B_{sk} are called the view public key and spend public key respectively. The corresponding secret keys, $b_{\text{vk}}, b_{\text{sk}} \in \mathbb{Z}_q$ such that $B_{\text{vk}} = G^{b_{\text{vk}}}$ and $B_{\text{sk}} = G^{b_{\text{sk}}}$, are called the secret view key and secret spend key. Bob can share his public view key B_{vk} and public spend key B_{sk} with anyone who wants to pay him. Multiple one-time addresses can be created using this public key pair.

Suppose in a Monero transaction *txn*, Alice wants to transfer the coins associated with one of her own one-time addresses to Bob. She creates a one-time address for Bob as follows. First, she chooses a random scalar r from $\mathbb{Z}_q$ and computes the destination one-time address $P' = G^{H_s(B_{\text{vk}}^r)} \cdot B_{\text{sk}}$, where $H_s : \mathbb{G} \mapsto \mathbb{Z}_q$ is a hash function which maps group elements to scalars[1]. Alice also computes the group element $R' = G^r$ and includes it in *txn*. Subsequently, *txn* containing (P', R') is added to the blockchain. For every (P, R) in every transaction in the blockchain, Bob computes a group element $P'' = G^{H_s(R^{b_{\text{vk}}})} \cdot B_{\text{sk}}$. To compute P'', only the knowledge of secret view key b_{vk} is required. For (P', R') in *txn*, P'' will be equal to P' as,

$$P'' = G^{H_s(R'^{b_{\text{vk}}})} \cdot B_{\text{sk}} = G^{H_s(G^{r b_{\text{vk}}})} \cdot B_{\text{sk}} = G^{H_s(B_{\text{vk}}^r)} \cdot B_{\text{sk}} = P'.$$

The above equality holds because $B_{\text{vk}}^r = G^{b_{\text{vk}} r} = R'^{b_{\text{vk}}}$. By verifying the above equality Bob can identify P' as his own one-time address. The secret key x' of the one-time address P' is $H_s(R'^{b_{\text{vk}}}) + b_{\text{sk}}$ because,

$$P' = G^{H_s(B_{\text{vk}}^r)} \cdot B_{\text{sk}} = G^{H_s(R'^{b_{\text{vk}}})} \cdot G^{b_{\text{sk}}} = G^{H_s(R'^{b_{\text{vk}}}) + b_{\text{sk}}}.$$

So the knowledge of b_{vk} is needed to identify that P' belongs to Bob. In this way, the fact that P' belongs to Bob is hidden.

Let the source one-time address from which Alice pays Bob in *txn* be P and x be its secret key i.e. $P = G^x$. Using a linkable ring signature [22], Alice hides the fact that P is the source of *txn*. Linkable ring signature is a cryptographic primitive which forms a ring (collection) of one-time addresses and proves that the signer knows the secret key of exactly one address in the ring. For example, Alice forms the ring as $\{P_1, P_2, \ldots, P, \ldots, P_n\}$ and generates a linkable ring signature. Here $P_1, P_2, \ldots$ are some one-time addresses taken from the Monero blockchain. They basically serve as cover addresses to hide the source of *txn* i.e. P. Linkable ring signatures further generate a group element which is called the *key image*. It is a deterministic function of the secret key (x in this case) corresponding to the one-time address which is owned by the signer of the linkable ring signature. The key image is defined as $I := H_p(P)^x$, where H_p is a hash function which generates a group element. In Monero, group elements are elliptic curve points, hence the subscript p is used. This key image I is used to determine whether the actual source of *txn* i.e. P is already spent or not. For this, the Monero blockchain maintains the set of already appeared key images, say, $\mathcal{I}$. If P is a spent address, then I would have already been in $\mathcal{I}$. This is because to spend from $P = G^x$, a linkable ring signature has to be signed with x which generates the same I. In such case, *txn* would be rejected by the Monero network. The name linkable ring signature comes from this linking feature of key images which helps detect double spending.

Finally, using confidential transactions, the amount ($a \in \{0, 1, 2, \ldots, 2^\beta - 1\}, \beta = 64$) associated with P' is hidden by the Pedersen commitment $C = G^y H^a$ where $y \xleftarrow{\$} \mathbb{Z}_q$ is called the blinding factor. Bob needs to know a, y to verify *txn* and spend from P' in future. To communicate a and y, Alice stores the following quantities in *txn*,

$$a' = a \oplus H_K(H_K(B_{\text{vk}}^r)) \tag{1}$$

$$y' = y \oplus H_K(B_{\text{vk}}^r), \tag{2}$$

where H_K is the Keccak hash function which maps group elements to scalars. Only Bob can recover a and y from a' and y' as follows.

$$a = a' \oplus H_K(H_K(R'^{b_{\text{vk}}})) \tag{3}$$

$$y = y' \oplus H_K(R'^{b_{\text{vk}}}). \tag{4}$$

The above equations hold again because $B_{\text{vk}}^r = G^{b_{\text{vk}} r} = R'^{b_{\text{vk}}}$. So knowledge of (R', b_{vk}) is needed to recover a, y from *txn* whereas we need the knowledge of $(b_{\text{vk}}, b_{\text{sk}})$ to generate the secret key x' of the one-time address P' (R' can be obtained from the Monero blockchain using b_{vk} as discussed above). To spend from P', Bob can sign a linkable ring signature with x'. The ability to generate x' implies the ability to generate a, y.

From the above discussion it is clear that proving knowledge of x such that $P = G^x$ is enough to prove the ownership of the one-time address P. More details on the above Monero technologies can be found in [7], [23].

C. MProve *and Its Drawback*

The first proof of reserves protocol for Monero was proposed and implemented by Stoffu Noether [24]. However, this scheme reveals the exchange-owned one-time addresses, their corresponding amounts, and the corresponding key images. MProve [7] is a proof of reserves protocol for Monero exchanges which provides some privacy by not revealing the exchange-owned addresses, their corresponding amounts and the total reserves amount. It generates a Pedersen commitment C_{res} to the total reserves amount of a Monero exchange. It also obfuscates all the exchange-owned one-time addresses by publishing a larger anonymity set. We give a brief summary of the MProve protocol below.

1) A summary of the MProve *Protocol:* In the MProve protocol, the exchange creates an anonymity set of one-time addresses i.e. $\mathcal{P}_{\text{anon}} = \{P_1, P_2, \ldots, P_n\}$ of which it knows the secret keys corresponding to some addresses. Let $\mathcal{P}_{\text{own}} \subset \mathcal{P}_{\text{anon}}$ be the set of exchange-owned one-time addresses. For each $P_i \in \mathcal{P}_{\text{anon}}$, the corresponding commitments to the amount i.e. $C_i = G^{y_i} H^{a_i}$ can be read from the blockchain. Apart from publishing a Pedersen commitment to the total reserves i.e. C_{res}, the exchange also publishes a group element C_i' for each $P_i \in \mathcal{P}_{\text{anon}}$ such that the following equation holds.

$$C_{\text{res}} = \prod_{i=1}^{n} C_i C_i'^{-1}. \tag{5}$$

[1] We ignore the concatenation of output index while generating one-time address as given in [7] for the ease of representation.

To satisfy equation (5), C_i's should be constructed as

$$C_i' = \begin{cases} G^{z_i} & \text{if } P_i \in \mathscr{P}_{\text{own}} \\ G^{z_i} C_i & \text{if } P_i \notin \mathscr{P}_{\text{own}}, \end{cases} \tag{6}$$

where z_is are some randomly chosen scalars from $\mathbb{Z}_q$. So the verifier of the proof will check the equality of equation (5) with $\{C_i, C_i'\}_{i=1}^n$, and C_{res} published by the exchange. Now one needs to ensure that the following statements hold.

S1. Set $\mathscr{P}_{\text{own}}$ does not contain any already spent one-time address.

S2. The exchange indeed followed the definition given in equation (6) while calculating C_i's.

To ensure that the above statements hold, the exchange publishes the following.

1) n linkable ring signatures $\{\sigma_i\}_{i=1}^n$, verifiable by a pair of group elements $(P_i, C_i' C_i^{-1})$.
2) n ring signatures $\{\gamma_i\}_{i=1}^n$, verifiable by a pair of group elements $(C_i', C_i' C_i^{-1})$.

A ring signature [25] scheme is a predecessor of linkable ring signature scheme which does not have the key image feature. For example, for some $i \in [n]$, ring signature γ_i proves that the exchange knows z_i such that either $C_i' = G^{z_i}$ or $C_i' C_i^{-1} = G^{z_i}$. Linkable ring signature σ_i proves that the exchange knows x_i or z_i such that $P_i = G^{x_i}$ or $C_i' C_i^{-1} = G^{z_i}$. It additionally reveals I_i which is equal to either $H_p(P_i)^{x_i}$ or $H_p(C_i' C_i^{-1})^{z_i}$. Notice that when σ_i is generated using the secret key corresponding to P_i, the key image of P_i is revealed. Any verifier can then check whether P_i is spent or not by checking if I_i is an element in the set of key images $\mathscr{I}$ from the Monero blockchain.

For $P_i \in \mathscr{P}_{\text{own}}$, the exchange can generate σ_i using either the secret key corresponding to P_i or $C_i' C_i^{-1}$. But if the exchange chooses to use the secret key corresponding to $C_i' C_i^{-1}$, $C_i C_i'^{-1}$ must be of the form G^{-z_i} for some z_i. So for this particular i, there will be zero contribution (no H term) of amount to C_{res} owing to the equation (5). Therefore the exchange has to generate σ_i using the secret key corresponding to P_i to include a_i in C_{res}. Then the key image of P_i i.e. $I_i = H_p(P_i)^{x_i}$ is revealed. Any verifier can now check if P_i is already spent or not i.e. I_i is an element in the set of key images $\mathscr{I}$ or not. In this way the validity of statement S1 given above can be verified. Both σ_is and γ_is are used to validate statement S2. Therefore, ring signatures γ_is and linkable ring signatures σ_is serve dual purposes. They validate both statements S1 and S2. Also for $i \in [n]$, they hide whether $P_i \in \mathscr{P}_{\text{own}}$ or $P_i \notin \mathscr{P}_{\text{own}}$.

Drawback of MProve[1]*:* Suppose a Monero exchange Ex uses an owned one-time address P_j to generate an MProve proof. Then Ex has to publish the key image of P_j i.e. I_j in the proof as a part of the linkable ring signature σ_j. Suppose at a later point of time, Ex creates a transaction txn to spend from P_j. In txn, Ex forms the ring of the linkable ring signature containing P_j and some other cover one-time addresses to obfuscate the source of txn. However in the linkable ring signature of txn, I_j appears again. When txn appears in the blockchain, an adversary can match I_j as a key image of txn and a part of the MProve proof published by Ex. Essentially she comes to know of the following statements.

1) As I_j appearing in txn has already appeared in an Ex generated MProve proof, Ex is spending in txn.
2) As I_j comes from σ_j which contains P_j as the only valid one-time address in the ring, P_j is owned by Ex.
3) In txn, P_j is the source of the transaction.

All the three statements affect the privacy of the exchange. However, the statements 2 and 3 are more crucial towards the privacy of the exchange as well as the entire Monero network because of the following reason. P_j is revealed as exchange-owned and txn effectively becomes a zero-mixin transaction. This increases traceability of transactions in the Monero blockchain [16], [17]. To avoid the cascade effect, P_j should be pruned from the set of UTXOs. The main reason for this drawback is the association of I_j with P_j through σ_j. MProve+ breaks this association using techniques from Bulletproofs [18] and Omniring [19] which are discussed next.

D. Bulletproofs and Omniring

The current Monero implementation suffers from the fact that the linkable ring signature size scales linearly with the size of the ring. This is crucial because these signatures are part of the transaction stored in the blockchain. As a consequence, it is expensive to use a large ring size (higher transaction size costs more transaction fees). Omniring [19] proposes a technique where the proof of validity of the transaction is logarithmic in the size of the ring. Omniring is motivated from Bulletproofs [18] and does not require any trusted setup. Currently, for Monero transactions with multiple sources, a separate ring is chosen for each source one-time address. Omniring proposes to use a single large ring for all source one-time addresses of a transaction, hence the name.

Bulletproofs [18] gives a state-of-the-art range proof system with logarithmic proof size. Here, given a Pedersen commitment[2] $C = G^v H^\gamma$, a prover can prove that $v \in \{0, 1, \ldots, N-1\}$ for some $N = 2^n \in \mathbb{Z}_q$ without revealing v. Currently, Bulletproofs are used in a Monero transaction to prove that all the output amounts in a transaction are in the right range. In the following, we discuss some aspects of Bulletproofs and Omniring that are relevant to us.

1) Range Proof Using Bulletproofs: In a range proof, a prover needs to prove that $v \in \{0, 1, \ldots, N-1\}$ for some $N = 2^n \in \mathbb{Z}_q$ where the verifier only knows C which is equal to $G^v H^\gamma$. To do so, v is represented in binary bits (say by binary vector $\mathbf{a}_L \in \mathbb{Z}_2^n$). The complement vector of $\mathbf{a}_L$, i.e. vector $\mathbf{1}^n - \mathbf{a}_L$, is denoted by $\mathbf{a}_R$. The condition $v \in \{0, 1, \ldots, N-1\}$ is then equivalently represented by following three constraint equations which use $\mathbf{a}_L$ and $\mathbf{a}_R$.

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \tag{7}$$

$$\langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle = 0 \tag{8}$$

$$\langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle = 0, \tag{9}$$

[1]The drawback of the MProve protocol is discussed in more detail in Section IV.C.1.

[2]In Monero, the amount is placed to the exponent of H and the blinding factor is placed to the exponent of G. In case of Bulletproofs [18], it is the opposite. However, this is just a difference in notation.

where the vector $\mathbf{y}^n = (1, y, y^2, \dots, y^{n-1})$ is constructed using the consecutive powers of $y \xleftarrow{\$} \mathbb{Z}_q$, a random challenge sent by the verifier. Here equation (7) ensures that $\mathbf{a}_L$ is the binary representation of v, equation (8) ensures that the component-wise product of $\mathbf{a}_L$ with $\mathbf{a}_R$ is always a zero vector, and equation (9) ensures that $\mathbf{a}_R$ is obtained by subtracting the elements of $\mathbf{a}_L$ from $\mathbf{1}^n$ vector. Both equations (8) and (9) ensure that the elements of $\mathbf{a}_L$ are either 0 or 1. Here the idea is that if a polynomial evaluates to zero at a random evaluation point chosen from a large set, then with high probability, the polynomial is a zero polynomial. These constraint equations are multiplied with powers of another random challenge $z \xleftarrow{\$} \mathbb{Z}_q$ sent by the verifier and added to form a single inner product as follows.

$$\langle \mathbf{a}_L - z\cdot\mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z\cdot\mathbf{1}^n) + z^2\cdot\mathbf{2}^n \rangle = z^2\cdot v + \delta(y, z), \quad (10)$$

where $\delta(y, z)$ is a function of y, z and can be calculated by the verifier. Bulletproofs proposes an optimized inner product proof with logarithmic proof size. However this inner product proof is not zero-knowledge. As $\mathbf{a}_L$, $\mathbf{a}_R$ are secret quantities, this inner product proof cannot be applied directly to prove equation (10). Thus the prover chooses two blinding vectors $\mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}_q^n$ and computes the following polynomials and their inner product.

$$\begin{aligned} l(X) &= \mathbf{a}_L - z\cdot\mathbf{1}^n + \mathbf{s}_L\cdot X && \in \mathbb{Z}_q^n[X] \\ r(X) &= \mathbf{y}^n \circ (\mathbf{a}_R + z\cdot\mathbf{1}^n + \mathbf{s}_R\cdot X) + z^2\cdot\mathbf{2}^n && \in \mathbb{Z}_q^n[X] \\ t(X) &= \langle l(X), r(X)\rangle = t_0 + t_1\cdot X + t_2\cdot X^2 && \in \mathbb{Z}_q[X], \end{aligned}$$

where $t_0 = z^2\cdot v + \delta(y, z)$. Then the prover and the verifier engage in a zero-knowledge protocol. The prover sends a commitment to $\mathbf{a}_L$, $\mathbf{a}_R$ as $A = H^\alpha \mathbf{G}^{\mathbf{a}_L}\mathbf{H}^{\mathbf{a}_R}$, a commitment to $\mathbf{s}_L$, $\mathbf{s}_R$ as $S = H^\rho \mathbf{G}^{\mathbf{s}_L}\mathbf{H}^{\mathbf{s}_R}$, and commitments to t_1 and t_2 as $T_1 = G^{t_1}H^{\tau_1}$, $T_2 = G^{t_2}H^{\tau_2}$ to the verifier where $\alpha, \rho, \tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$ are random scalars and $\mathbf{G}, \mathbf{H} \xleftarrow{\$} \mathbb{G}^n$ are random base vectors. The verifier sends a random evaluation point $x \xleftarrow{\$} \mathbb{Z}_q$ to the prover. Prover then evaluates $\mathbf{l} = l(x)$, $\mathbf{r} = r(x)$, and $\hat{t} = \langle \mathbf{l}, \mathbf{r}\rangle$. Because of blinding vectors $\mathbf{s}_L$ and $\mathbf{s}_R$, the prover can use $\mathbf{l}$, $\mathbf{r}$ in the inner product proof to prove that $\hat{t} = \langle \mathbf{l}, \mathbf{r}\rangle$, without revealing $\mathbf{a}_L$ and $\mathbf{a}_R$. Using C, A, S, T_1, T_2, $\mathbf{l}$, $\mathbf{r}$, $\hat{t}$, and other quantities sent by the prover, the verifier verifies the following conditions.

i. $\hat{t} \stackrel{?}{=} t_0 + t_1 x + t_2 x^2$.
ii. $\mathbf{l} \stackrel{?}{=} \mathbf{a}_L - z\cdot\mathbf{1}^n + \mathbf{s}_L\cdot x$ and $\mathbf{r} \stackrel{?}{=} \mathbf{y}^n \circ (\mathbf{a}_R + z\cdot\mathbf{1}^n + \mathbf{s}_R\cdot x) + z^2\cdot\mathbf{2}^n$.
iii. $\hat{t} \stackrel{?}{=} \langle \mathbf{l}, \mathbf{r}\rangle$.

As x is chosen randomly, this is equivalent to checking equation (10). However instead of sending $\mathbf{l}$, $\mathbf{r}$ (size $2n$) directly, the prover uses the optimized inner product proof of $\log_2 n$ size to prove that $\hat{t} = \langle \mathbf{l}, \mathbf{r}\rangle$. Hence the range proof is a logarithmic size range proof. Omniring and MProve+ follow a similar idea as discussed above.

2) Omniring: For a single source transaction in Omniring [19], we can prove the knowledge of the secret key corresponding to one element in the ring $\mathbf{P}$ (represented by a vector) by proving knowledge of a secret key ($x \in \mathbb{Z}_q$) and one secret unit vector $\mathbf{e}$ such that $\mathbf{P}^{\mathbf{e}} = G^x$. The unit vector $\mathbf{e}$ has zeros in $n-1$ places and 1 in the location corresponding to the source one-time address location in the ring. Therefore $\mathbf{e}$ selects only the source address in the ring vector $\mathbf{P}$. For a multiple source transaction, separate unit vectors are needed. The discrete logarithm relation can be alternatively represented as

$$1_g = G^{-x}\mathbf{P}^{\mathbf{e}}, \quad (11)$$

where 1_g is the identity element of the group $\mathbb{G}$. The Omniring authors called this equation the *main equality*. For a multiple source transaction in Omniring, the secret vector is formed by concatenating all the secret keys, unit vectors, output amounts, and blinding factors. The constraint equations are formed to ensure that the unit vectors contain zeros in all places except a single 1 in the source address location, the output amounts are in the right range, and the sum of input amounts is equal to the sum of output amounts and transaction fees for the transaction. The equations are added with blinding factors to form a single inner product like Bulletproofs. Then a technique similar to the Bulletproofs-based range proof is followed except with the following difference.

Let us define the secret vector as $\mathbf{a} = (-x \| \mathbf{e})$. Then the main equality (11) can be alternatively represented as

$$(G \| \mathbf{P})^{\mathbf{a}} = 1_g. \quad (12)$$

In the Bulletproofs-based range proof, to generate commitment A to the secret vectors $\mathbf{a}_L$ and $\mathbf{a}_R$, random base vectors $\mathbf{G}$ and $\mathbf{H}$ are chosen by the prover. As they are randomly generated, discrete logarithm relation between elements of the base vectors are not known. This is necessary and used in the extraction of the witnesses. In Omniring, from equation (12) we observe that the base vectors to generate A must include $\mathbf{P}$ to show that the main equality (11) holds. However, the prover might know the discrete logarithm relation between the elements of $\mathbf{P}$ especially when some of them are owned by the prover. The authors mitigate this issue by replacing the base vector $\mathbf{G}$ with $\mathbf{G}_w := ((G\|\mathbf{P})^w \circ \mathbf{Q})$ where w, $\mathbf{Q}$ are randomly chosen from $\mathbb{Z}_q$ and $\mathbb{G}^{n+1}$ respectively. They showed that even if the discrete logarithm relation between elements of $\mathbf{P}$ is known, it is computationally infeasible to compute a discrete logarithm relation between elements of $\mathbf{G}_w$. Further, for $w' \neq w$, it holds that $\mathbf{G}_w^{\mathbf{a}} = \mathbf{G}_{w'}^{\mathbf{a}}$ if the main equality (11) holds. Recall that the same base $\mathbf{G}$ is used to generate A and S in the Bulletproofs-based range proof. In Omniring, $\mathbf{G}_0$ is used to generate A and $\mathbf{G}_w$ is used to generate S, where $w \xleftarrow{\$} \mathbb{Z}_q$ is sent by the verifier after receiving A. The rest of the protocol will work only if $\mathbf{G}_0^{\mathbf{a}} = \mathbf{G}_w^{\mathbf{a}}$ holds. In this way the main equality (11) is implicitly verified. MProve+ follows this technique.

III. MProve+ : An Improvement over MProve

In this section, we describe MProve+ which helps to remove the drawback of MProve using the techniques of Bulletproofs and Omniring.

A. Intuition

In both MProve and MProve+ schemes, a Monero exchange Ex reveals a list of one-time addresses $\mathcal{P}_{\text{anon}} =$

$\{P_1, P_2, \ldots, P_n\}$ as the anonymity set. Suppose in the set $\mathcal{P}_{\text{anon}}$, *Ex* owns s one-time addresses which are to be used as source addresses. In both our schemes, the key images corresponding to the source addresses are published to show that the source addresses are not spent yet. In the MProve scheme, a key image (real or dummy) is published for each one-time address in $\mathcal{P}_{\text{anon}}$. This is to hide if a particular address in $\mathcal{P}_{\text{anon}}$ is a source or not. However, this creates the association of a key image with a unique address in the anonymity set $\mathcal{P}_{\text{anon}}$ and introduces the privacy issue discussed in Section II.C.1. In the MProve+ scheme, we publish the key images corresponding to only the source addresses in $\mathcal{P}_{\text{anon}}$, without revealing the association between the key images and their actual source addresses. An observer will be only able to infer that each key image can be the key image of any address in the set $\mathcal{P}_{\text{anon}}$. While this reveals the number of source addresses s, the association of a key image with multiple one-time addresses helps to remove the drawback of the MProve scheme (as discussed in Section IV.C). Below we give an overview of the MProve+ scheme.

- In the MProve+ scheme, *Ex* publishes a vector of one-time addresses $\mathbf{P} = (P_1, P_2, \ldots, P_n)$ which have Pedersen commitments $\mathbf{C} = (C_1, C_2, \ldots, C_n)$ associated with them. *Ex* also reveals a key image vector $\mathbf{I} = (I_1, I_2, \ldots, I_s)$ and a Pedersen commitment C_{res} to the total reserves.
- First, *Ex* wants to prove that it knows the s secret keys corresponding to some s of the n addresses in $\mathbf{P}$. In other words, it wants to prove that there are s distinct indices $\{i_1, i_2, \ldots, i_s\} \subset \{1, 2, \ldots, n\}$ such that it knows $\{x_1, x_2, \ldots, x_s\}$ where $P_{i_j} = G^{x_j}$ for all $j = 1, 2, \ldots, s$. *Ex* does not want to reveal the indices.
- Second, *Ex* wants to prove that the key images $\mathbf{I} = (I_1, I_2, \ldots, I_s)$ correspond to the same s indices. In other words, $I_j = \left(H_p(P_{i_j})\right)^{x_j}$ for $j = 1, 2, \ldots, s$.
- Third, *Ex* wants to prove that for the same s indices it knows[1] the blinding factor $r_j \in \mathbb{Z}_q$ and the amount $a_j \in \{0, 1, \ldots, 2^\beta - 1\}$ corresponding to the Pedersen commitments $C_{i_1}, C_{i_2}, \ldots, C_{i_s}$.
- Finally, *Ex* wants to prove that the amount in C_{res} is the same as the sum of the amounts in the Pedersen commitments at the same s indices. In other words, if $C_{\text{res}} = G^{r_{\text{res}}} H^{a_{\text{res}}}$ and $C_{i_j} = G^{r_j} H^{a_j}$, then *Ex* wants to prove that $a_{\text{res}} = \sum_{j=1}^{s} a_j$.

Note that the exchange is only trying to prove that C_{res} is a commitment to a sum of amounts from addresses it owns. To prove that it has enough reserves to meet its liabilities, it has to generate another commitment C_{liab} to its liabilities (using a protocol like DAPOL [14]) and show that $C_{\text{res}} C_{\text{liab}}^{-1}$ is a commitment to a non-negative amount (via a range proof). To prove the above statements, *Ex* proceeds as follows.

- *Ex* proves knowledge of s secret keys, amounts, and blinding factors by proving knowledge of s unit vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_s$, secret vectors $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$, $\mathbf{a} = (a_1, a_2, \ldots, a_s)$, and $\mathbf{r} = (r_1, r_2, \ldots, r_s)$ such that $\mathbf{P}^{\mathbf{e}_j} = G^{x_j} \wedge \mathbf{C}^{\mathbf{e}_j} = G^{r_j} H^{a_j}$ holds for $j = 1, 2, \ldots, s$. As discussed in Section II.D.2, the jth unit vector $\mathbf{e}_j$ is used to choose the jth source address and its corresponding Pedersen commitment in the vectors $\mathbf{P}$ and $\mathbf{C}$ respectively.
- As proving $I_j = \left(H_p(P_{i_j})\right)^{x_j}$ is the same as proving $I_j^{x_j^{-1}} = H_p(P_{i_j})$, *Ex* proves that the unit vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_s$ and secret keys in the vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_s)$ also satisfy $\mathbf{H}_p^{\mathbf{e}_j} = I_j^{x_j^{-1}}$ for all $j = 1, 2, \ldots, s$, where $\mathbf{H}_p = \left(H_p(P_1), H_p(P_2), \ldots, H_p(P_n)\right)$ is the vector of hashed one-time addresses.
- Finally, *Ex* shows that $a_{\text{res}} = \sum_{j=1}^{s} a_j$ by proving that there exists a binary[2] vector $\mathbf{b} = (b_0, b_1, \ldots, b_{\beta-1})$ such that $\sum_{i=0}^{\beta-1} b_i 2^i = a_{\text{res}} \wedge \sum_{i=0}^{\beta-1} b_i 2^i = \sum_{j=1}^{s} a_j$ holds.

All the above mentioned conditions for a proof of reserves are accumulated in a main equality similar to Omniring [19]. All the elements in the exponents of the main equality form the secret vector. To show that these elements of the secret vector satisfy all the necessary conditions, some constraint equations are formed. These equations collectively form a single inner product. We use Bulletproofs to prove that this inner product holds in zero-knowledge as discussed in Section II.D.1. The main equality is also implicitly verified during this inner product verification. This is similar to the technique that Omniring uses and is discussed in Section II.D.2. Below, we describe the MProve+ scheme in detail.

B. Construction of MProve+

The MProve+ protocol is constructed by modifying the scheme given in Appendix F of the Omniring [19] paper. Roughly speaking, a MProve+ proof is a giant Omniring transaction with a single output commitment, namely, a commitment to the total reserves. The differences between the MProve+ protocol and the protocol given in Appendix F of the Omniring paper [19] are as follows.

1) A one-time address is denoted as $P = H^x$ in the Omniring scheme. However it is denoted as $P = G^x$ in the MProve+ scheme.
2) A commitment is denoted as $C = G^a H^r$ in the Omniring scheme where a, r denote the amount and the blinding factor respectively. However a commitment in the MProve+ scheme is denoted by $C = G^r H^a$.
3) The number of outputs $|\tau|$ in the Omniring scheme is 1 in the MProve+ scheme. Hence the binary vector $\mathsf{vec}(\mathbf{B})$ of length $\beta|\tau|$ in $\mathbf{c}_L$ is replaced by a binary vector $\mathbf{b}$ of length β in the MProve+ scheme.

Below we give the language for the MProve+ protocol satisfying the requirements mentioned in Section III.A.

[1] To prove ownership, proving knowledge of the secret key is enough (Section II.B). However, if we show the knowledge of the source amounts and the blinding factors, then the commitment to the total reserves is more efficiently computed giving better performance.

[2] This binary representation basically gives a range proof on a_{res} and is motivated from Omniring. In our case, ranges of $a_1, a_2, \ldots, a_s$ are already verified in the blockchain. Therefore proving that a_{res} is the sum of them implicitly verifies its range. However it is observed that, using the binary representation $\mathbf{b}$ instead of a_{res} in the secret vector gives better performance.

$$\mathcal{L}^{\mathsf{crs}}_{\mathsf{MP+}} = \left\{ \left(\begin{matrix} \mathbf{P},\, \mathbf{C},\, \mathbf{H}_p, \\ \{I_j\}_{j=1}^{s},\, C_{\text{res}} \end{matrix} \right) \;\middle|\; \begin{array}{l} \exists(\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \mathbf{b}, \mathbf{a}, \mathbf{r}, a_{\text{res}}, r_{\text{res}}) \\ \text{such that each } \mathbf{e}_j \text{ is a unit vector,} \\ \mathbf{P}^{\mathbf{e}_j} = G^{x_j}, x_j \in \boldsymbol{x}, \\ \mathbf{C}^{\mathbf{e}_j} = G^{r_j} H^{a_j}, r_j \in \mathbf{r}, a_j \in \mathbf{a}, \\ \mathbf{H}_p^{\mathbf{e}_j} = I_j^{x_j^{-1}} \forall j \in [s], \\ \mathbf{b} \text{ is the binary representation} \\ \text{of } a_{\text{res}} \text{ of length } \beta, \\ C_{\text{res}} = G^{a_{\text{res}}} H^{r_{\text{res}}}, \\ \sum_{a_j \in \mathbf{a}} a_j = a_{res}. \end{array} \right\} \tag{13}$$

Here the common reference string crs specifies the necessary details like description of the group, its generators, the hash function $H_p(\cdot)$ to be used. We define $(\mathbf{P}, \mathbf{C}, \mathbf{H}_p, \{I_j\}_{j=1}^{s}, C_{\text{res}})$ as the statement stmt of the language. We also define $(\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \mathbf{b}, \mathbf{a}, \mathbf{r}, a_{\text{res}}, r_{\text{res}})$ as the witness wit of the language.

C. Forming the Main Equality

We define $\mathcal{E}$ as the $s \times n$ matrix containing $\mathbf{e}_1, \ldots, \mathbf{e}_s$ as the rows. We give the following definitions similar to the definitions in Appendix F of the Omniring paper [19]. Here u, v are public coin challenges sent by the verifier.

$$\hat{\mathbf{Y}} := \mathbf{P} \circ \mathbf{C}^{\circ u} \circ \mathbf{H}_p^{\circ u^2}, \tag{14}$$

$$\hat{\mathbf{I}} := \mathbf{I}^{\circ - u^2 \boldsymbol{v}^s}, \tag{15}$$

$$\hat{\mathbf{e}} := \boldsymbol{v}^s \mathcal{E}, \tag{16}$$

$$\xi := -\langle \boldsymbol{v}^s, u \cdot \mathbf{a} \rangle, \tag{17}$$

$$\eta := -\langle \boldsymbol{v}^s, \boldsymbol{x} + u \cdot \mathbf{r} \rangle. \tag{18}$$

The main equality is formed to verify the following representations from the language given in (13).

$$G^{-x_j} \mathbf{P}^{\mathbf{e}_j} = 1_g, \forall j \in [s], \tag{19}$$

$$G^{-r_j} H^{-a_j} \mathbf{C}^{\mathbf{e}_j} = 1_g, \forall j \in [s], \tag{20}$$

$$I_j^{-x_j^{-1}} \mathbf{H}_p^{\mathbf{e}_j} = 1_g, \forall j \in [s], \tag{21}$$

where 1_g is the identity element of $\mathbb{G}$. equations (19), (20), and (21) represent s equations each. We exponentiate the jth equation ($j \in [s]$) of (19), (20), and (21) with v^{j-1}, uv^{j-1}, and $u^2 v^{j-1}$ respectively. Multiplying all these modified equations together gives us the following equation.

$$H^{\xi} G^{\eta} \hat{\mathbf{Y}}^{\hat{\mathbf{e}}} \hat{\mathbf{I}}^{\boldsymbol{x}^{\circ -1}} = 1_g. \tag{22}$$

Equation (22) is called the main equality for $\mathsf{MProve+}$.

D. Defining Secret Vectors and the Constraint Equations

Now we construct the secret vectors given in Figure 1. Here, we essentially need to consider all the exponents in the main equality (22). Additionally we need to consider $\mathsf{vec}(\mathcal{E})$ which is a vector of length sn, formed by concatenating all the rows of matrix $\mathcal{E}$. Vector $\mathsf{vec}(\mathcal{E})$ is used to ensure that all the rows of $\mathcal{E}$ are indeed unit vectors i.e. they contain a single 1 and rest of the elements are 0. The vector $\mathbf{c}_L$ has 2 scalars (ξ, η), 3 vectors $(\boldsymbol{x}^{\circ -1}, \mathbf{a}, \mathbf{r})$ of length s, and 3 vectors $(\hat{\mathbf{e}}, \mathsf{vec}(\mathcal{E}), \mathbf{b})$ of length n, sn, and β respectively. Hence the length of $\mathbf{c}_L$ is $m = sn + n + 3s + 2 + \beta$. Vector $\mathbf{c}_R$ is an auxiliary vector of the same length m used to prove the constraints on the witnesses.

Figure 2 gives some constraint vectors which are used to select various parts of the secret vector and give a constraint in terms of an inner product. All these inner product constraint equations are given in Figure 4. Here equations (23) and (31) verify that all elements of $\mathsf{vec}(\mathcal{E})$ and $\mathbf{b}$ are either 0 or 1. equation (24) verifies that the $(n+2+1)$th to $(n+2+s)$th elements of $\mathbf{c}_L$ and $\mathbf{c}_R$ are inverses of each other. Equation (25) verifies that $\mathbf{b}$ is the binary representation of a_{res}. Equation (26) verifies that each block of n elements of the vector $\mathsf{vec}(\mathcal{E})$ contains a single 1 and the rest of the elements are 0 i.e. each such block is a unit vector. Equations (27) and (28) verify the definitions of ξ and η given in equations (17) and (18) respectively. Equation (29) verifies the definition of $\hat{\mathbf{e}}$ given in equation (16). Lastly, equation (30) verifies the equality $\sum_{i=0}^{\beta-1} b_i 2^i = \sum_{j=1}^{s} a_j$. Notice that a_{res} is not a member of the secret vector and the verification $C_{\text{res}} \stackrel{?}{=} G^{r_{\text{res}}} H^{a_{\text{res}}}$ is not done in the set of constraint equations. As we shall see, this verification is done in the final verification equation (V3) similar to the Omniring scheme.

$$\begin{aligned} \mathbf{c}_L &:= (\ \xi \parallel \eta \parallel \hat{\mathbf{e}} \parallel \boldsymbol{x}^{\circ -1} \parallel \mathsf{vec}(\mathcal{E}) \parallel \mathbf{b} \parallel \mathbf{a} \parallel \mathbf{r}\ ) \\ \mathbf{c}_R &:= (\ \mathbf{0}^{2+n} \parallel \boldsymbol{x} \parallel \mathbf{1}^{sn} - \mathsf{vec}(\mathcal{E}) \parallel \mathbf{1}^{\beta} - \mathbf{b} \parallel \mathbf{0}^{2s}\ ) \end{aligned}$$

Fig. 1: Honest encoding of witness.

$$\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \\ \mathbf{v}_4 \\ \mathbf{v}_5 \\ \mathbf{v}_6 \\ \mathbf{v}_7 \\ \mathbf{v}_8 \\ \mathbf{u}_5 \end{bmatrix} := \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \multicolumn{2}{c}{\boldsymbol{y}^{sn+\beta}} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \boldsymbol{y}^s & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{2}^{\beta} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \boldsymbol{y}^s \otimes \mathbf{1}^n & \cdot & \cdot & \cdot \\ 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\boldsymbol{v}^s & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & u\boldsymbol{v}^s \\ \cdot & \cdot & -\boldsymbol{y}^n & \cdot & \boldsymbol{v}^s \otimes \boldsymbol{y}^n & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \mathbf{2}^{\beta} & -\mathbf{1}^s & \cdot \\ \cdot & \cdot & \cdot & \cdot & \multicolumn{2}{c}{\boldsymbol{y}^{sn+\beta}} & \cdot & \cdot \\ \cdot & \cdot & \cdot & \boldsymbol{v}^s & \cdot & \cdot & \cdot & \cdot \end{bmatrix}$$

Fig. 2: Definitions of constraint vectors (Dots mean zero vectors).

$$\boldsymbol{\theta} := \mathbf{v}_0 + z \cdot \mathbf{v}_1,\ \boldsymbol{\mu} := \sum_{i=2}^{8} z^i \cdot \mathbf{v}_i,\ \boldsymbol{\nu} = z^8 \cdot \mathbf{v}_8,\ \boldsymbol{\omega} = z^5 \cdot \mathbf{u}_5,$$

$$\kappa = z \cdot \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle + z^3 \cdot \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle + \langle \mathbf{1}^m, \boldsymbol{\nu} \rangle,$$

$$\boldsymbol{\alpha} = \boldsymbol{\theta}^{\circ -1}(\boldsymbol{\omega} + \boldsymbol{\nu}), \quad \boldsymbol{\beta} = \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mu}, \quad \delta = \kappa + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle.$$

Fig. 3: Definitions of constraint vectors (continued).

$$\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0 \iff$$
$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_0 \rangle = 0 \tag{23}$$
$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_1 \rangle = \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle \tag{24}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_2 \rangle = a_{\text{res}} \tag{25}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_3 \rangle = \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle \tag{26}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_4 \rangle = 0 \tag{27}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_5 \rangle + \langle \boldsymbol{\gamma}_R, \mathbf{u}_5 \rangle = 0 \tag{28}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_6 \rangle = 0 \tag{29}$$
$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_7 \rangle = 0 \tag{30}$$
$$\langle \boldsymbol{\gamma}_L + \boldsymbol{\gamma}_R - \mathbf{1}^m, \mathbf{v}_8 \rangle = 0 \tag{31}$$

Fig. 4: A system of constraint equations guaranteeing integrity of encoding of witness.

E. Combining All Constraint Equations in a Single Inner Product

For a random scalar $z \in \mathbb{Z}_q$ sent by the verifier, multiplying equations (23) to (31) by consecutive powers of z namely $1, z, z^2, \ldots, z^8$ and adding them gives,

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\mu} \rangle + \langle \boldsymbol{\omega} + \boldsymbol{\nu}, \boldsymbol{\gamma}_R \rangle = \kappa + z^2 a_{\text{res}}, \tag{32}$$

where $\boldsymbol{\theta}, \boldsymbol{\nu}$, and κ are defined in Figure 3. To get a single inner product, we modify (32) as follows,

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\mu} \rangle + \langle (\boldsymbol{\omega} + \boldsymbol{\nu}) \circ \boldsymbol{\theta}^{\circ -1}, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\mu} \rangle \tag{33}$$
$$= \kappa + \langle (\boldsymbol{\omega} + \boldsymbol{\nu}) \circ \boldsymbol{\theta}^{\circ -1}, \boldsymbol{\mu} \rangle + z^2 a_{\text{res}}$$
$$\implies \langle \boldsymbol{\gamma}_L + \boldsymbol{\alpha}, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\mu} \rangle = \delta + z^2 a_{\text{res}}, \tag{34}$$

where $\boldsymbol{\alpha}$ and δ are defined in Figure 3.

In the following protocol, we prove the inner product given in equation (34) using the Bulletproofs technique as discussed in Section II.D.1. The main equality (22) is implicitly proved using the technique followed in Omniring as discussed in Section II.D.2.

MProve+ Protocol ($\Pi_{\mathsf{MProve+}}$): Argument of knowledge for $\mathcal{L}^{\mathsf{crs}}_{\mathsf{MP+}}$.

Setup $(\lambda, \mathcal{L})$:

$\mathsf{crs} = (\mathbb{G}, q, G, H, H_p(\cdot))$.
Generate: $r_{\text{res}} \xleftarrow{\$} \mathbb{Z}_q, a_{\text{res}} = \sum_{a_j \in \mathbf{a}} a_j, C_{\text{res}} = G^{r_{\text{res}}} H^{a_{\text{res}}}$, $\mathbf{b} = (b_0, b_1, b_{\beta-1})$, such that $\sum_{k=0}^{\beta-1} b_k 2^k = a_{\text{res}}, \mathbf{P} = \{P_j\}_{j=1}^n, \mathbf{C} = \{C_j\}_{j=1}^n, \mathbf{H}_p = \{H_P(P_j)\}_{j=1}^n, \mathbf{I} = \{I_j\}_{j=1}^s = \{H_p(P_j)^{x_j}\}_{j=1}^s$
Output: $\mathsf{stmt} = (\mathbf{P}, \mathbf{C}, \mathbf{H}_p, \mathbf{I}, C_{\text{res}})$, $\mathsf{wit} = (\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \mathbf{b}, \mathbf{a}, \mathbf{r}, a_{\text{res}}, r_{\text{res}})$

$\langle \mathcal{P}(\mathsf{crs}, \mathsf{stmt}, \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt}) \rangle$:

$\mathcal{V}$: $u, v \xleftarrow{\$} \mathbb{Z}, F \xleftarrow{\$} \mathbb{G}, \mathbf{Q} \xleftarrow{\$} \mathbb{G}^{2+n+s}, \mathbf{G}' \xleftarrow{\$} \mathbb{G}^{m-n-s-2}, \mathbf{H} \xleftarrow{\$} \mathbb{G}^m$

$\mathcal{V} \longrightarrow \mathcal{P}$: $u, v, F, \mathbf{Q}, \mathbf{G}', \mathbf{H}$

$\mathcal{P}$, $\mathcal{V}$:

1) Compute $\hat{\mathbf{Y}} = \mathbf{P} \circ \mathbf{C}^{\circ u} \circ \mathbf{H}_p^{\circ u^2}$ and $\hat{\mathbf{I}} = \mathbf{I}^{\circ -u^2 \boldsymbol{v}^s}$
2) For $w \in \mathbb{Z}_q$, denote
$$\mathbf{G}_w := \left[\left((H \| G \| \hat{\mathbf{Y}} \| \hat{\mathbf{I}})^{\circ w} \circ \mathbf{Q} \right) \| \mathbf{G}' \right] \tag{35}$$

$\mathcal{P}$:

1) $r_A \xleftarrow{\$} \mathbb{Z}_q$
2) $A := F^{r_A} \mathbf{G}_0^{\mathbf{c}_L} \mathbf{H}^{\mathbf{c}_R}$
Note: $\mathbf{G}_w^{\mathbf{c}_L} = \mathbf{G}_{w'}^{\mathbf{c}_L}$ $\forall w, w' \in \mathbb{Z}_q$ since $H^{\xi} G^{\eta} \hat{\mathbf{Y}}^{\hat{\mathbf{e}}} \hat{\mathbf{I}}^{\boldsymbol{x}^{\circ -1}} = 1_g$ by the main equality (22). Thus $A = F^{r_A} \mathbf{G}_w^{\mathbf{c}_L} \mathbf{H}^{\mathbf{c}_R}$ $\forall w \in \mathbb{Z}_q$

$\mathcal{P} \longrightarrow \mathcal{V}$: A

$\mathcal{V}$: $w \xleftarrow{\$} \mathbb{Z}_q$

$\mathcal{V} \longrightarrow \mathcal{P}$: w

$\mathcal{P}$:

1) $r_S \xleftarrow{\$} \mathbb{Z}_q, \mathbf{s}_L \xleftarrow{\$} \mathbb{Z}_q^m$, for $\mathbf{s}_R \in \mathbb{Z}_q^m$ s.t. for $j \in [m]$
$$\mathbf{s}_R[j] = \begin{cases} s_j \xleftarrow{\$} \mathbb{Z}_q, & \text{for } \mathbf{c}_R[j] \neq 0 \\ 0, & \text{for } \mathbf{c}_R[j] = 0 \end{cases}$$
2) $S = F^{r_S} \mathbf{G}_w^{\mathbf{s}_L} \mathbf{H}^{\mathbf{s}_R}$

$\mathcal{P} \longrightarrow \mathcal{V}$: S

$\mathcal{V}$: $y, z \xleftarrow{\$} \mathbb{Z}_q$

$\mathcal{V} \longrightarrow \mathcal{P}$: y, z

$\mathcal{P}$:

1) Define the following polynomials (in X):
$$l(X) := \mathbf{c}_L + \boldsymbol{\alpha} + \mathbf{s}_L \cdot X \qquad \in \mathbb{Z}_q^m[X]$$
$$r(X) := \boldsymbol{\theta} \circ (\mathbf{c}_R + \mathbf{s}_R \cdot X) + \boldsymbol{\mu} \qquad \in \mathbb{Z}_q^m[X]$$
$$t(X) := \langle l(X), r(X) \rangle = t_2 X^2 + t_1 X + t_0 \qquad \in \mathbb{Z}_q^N[X]$$
for some $t_2, t_1, t_0 \in \mathbb{Z}_q$. In particular,
$$t_0 = z^2 a_{\text{res}} + \delta$$
2) $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$
3) $T_1 = H^{t_1} G^{\tau_1}, T_2 = H^{t_2} G^{\tau_2}$

$\mathcal{P} \longrightarrow \mathcal{V}$: T_1, T_2

$\mathcal{V}$: $x \xleftarrow{\$} \mathbb{Z}_q$

$\mathcal{V} \longrightarrow \mathcal{P}$: x

$\mathcal{P}$:

1) $\boldsymbol{\ell} := l(x) = \mathbf{c}_L + \boldsymbol{\alpha} + \mathbf{s}_L \cdot x \ \in \mathbb{Z}_q^m$
2) $\boldsymbol{\mathfrak{r}} := r(x) = \boldsymbol{\theta} \circ (\mathbf{c}_R + \mathbf{s}_R \cdot x) + \boldsymbol{\mu} \ \in \mathbb{Z}_q^m$
3) $\hat{t} := \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle \ \in \mathbb{Z}_q$
4) $\tau := z^2 r_{\text{res}} + \tau_2 x^2 + \tau_1 x$
5) $r := r_A + r_S x$

$\mathcal{P} \longrightarrow \mathcal{V}$: $\boldsymbol{\ell}, \boldsymbol{\mathfrak{r}}, \hat{t}, \tau, r$

$\mathcal{V}$: Checks if the following relations hold:

(V1) $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle$

(V2) $F^r \mathbf{G}_w^{\ell} \mathbf{H}^{\theta^{\circ-1} \circ \mathfrak{r}} \stackrel{?}{=} A S^x \mathbf{G}_w^{\alpha} \mathbf{H}^{\beta}$

(V3) $H^{\hat{t}} G^{\tau} \stackrel{?}{=} H^{\delta} C_{\text{res}}^{z^2} T_1^x T_2^{x^2}$

Verification equations (V1) and (V2) need $\ell, \mathfrak{r} \in \mathbb{Z}_q^m$ which requires $\mathcal{O}(m)$ size communication from the prover. Instead, we can use the inner product protocol which is used in Bulletproofs [18] and Omniring [19]. The inner product argument is expressed by the following language.

$$\mathcal{L}_{IP} = \left\{ P \in \mathbb{G}, c \in \mathbb{Z}_q \;\middle|\; \begin{array}{c} \exists (\mathbf{a}, \mathbf{b}) \text{ such that} \\ P = U^c \mathbf{G}^{\mathbf{a}} \mathbf{H}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle . \end{array} \right\} \tag{36}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^{|\mathbf{a}|}$, $\mathbf{G}, \mathbf{H} \xleftarrow{\$} \mathbb{G}^{|\mathbf{a}|}$, $U \xleftarrow{\$} \mathbb{G}$. In our case, the verifier sets $c = \hat{t}$. Apart from the prover, the verifier can also compute the Pedersen commitment P to ℓ and $\mathfrak{r}$ without knowing ℓ and $\mathfrak{r}$ as

$$P = U^{\hat{t}} \mathbf{G}_w^{\ell} (\mathbf{H}')^{\mathfrak{r}} = U^{\hat{t}} (F)^{-r} A S^x \mathbf{G}_w^{\alpha} \mathbf{H}^{\beta}, \tag{37}$$

by verification equation (V2), where $\mathbf{H}' = \mathbf{H}^{\theta^{\circ-1}}$. With this, the prover and the verifier engage in the inner product argument to prove verification equations (V1) and (V2). So the prover does not send ℓ and $\mathfrak{r}$ in the previous step reducing the communication cost to $\mathcal{O}(\log_2(m))$. The inner product argument is public coin, so can be done by only one interaction between the prover and the verifier using the Fiat-Shamir heuristic. We have the following theorems which come directly from Theorem F.2 and F.3 of the Omniring paper [19], hence their proofs are omitted.

Theorem 1: The argument presented in $\Pi_{\mathsf{MProve+}}$ is public-coin, constant-round, perfectly complete and perfect special honest-verifier zero-knowledge.

Theorem 2: Assuming the discrete logarithm assumption holds over $\mathbb{G}$, $\Pi_{\mathsf{MProve+}}$ has computational witness-extended-emulation for extracting a valid witness wit.

F. Proof Generation and Verification

The exchange follows the $\Pi_{\mathsf{MProve+}}$ protocol and publishes $(\mathbf{P}, \mathbf{I}, C_{\text{res}})$ and a $\Pi_{\mathsf{MProve+}}$ proof. The verifier of $\Pi_{\mathsf{MProve+}}$ protocol does the following verification steps.

1) Computes $\mathbf{H}_p$ using the hash function $H_p(\cdot)$. Reads $\mathbf{C}$ by looking at the Monero blockchain and using $\mathbf{P}$.
2) Checks that no element in $\mathbf{I}$ appears in the set of key images $\mathcal{I}$. If this is not the case then double spending is detected.
3) Checks that all the elements in $\mathbf{I}$ are distinct. This is to ensure that no source amount is used more than once in calculating the total reserves.
4) Checks the proof of $\Pi_{\mathsf{MProve+}}$ as discussed above.
5) Checks that no element in $\mathbf{I}$ appears in the MProve+ proofs generated by another Monero exchange. If this is not the case then address sharing collusion is detected.

The verifier rejects the proof if any of the above steps fails. Otherwise she accepts the proof. For faster verification, we have done some optimization as discussed below.

Faster Verification: The cost of verifying an MProve+ proof is largely determined by the verification of the argument of knowledge $\Pi_{\mathsf{MProve+}}$. A verifier checks the validity of $\Pi_{\mathsf{MProve+}}$ by checking the verification equation (V3) and the inner product argument. As noted in [18], an inner product argument $\Pi_{\mathsf{IP}} = \left(\{L_j, R_j\}_{j=1}^{\log_2 m} \in \mathbb{G}, a, b \in \mathbb{Z}_q \right)$ for the language in (36) can be verified in a single multi-exponentiation check as

$$\mathbf{G}^{a \cdot \mathbf{s}} \cdot \mathbf{H}^{a \cdot \mathbf{s}^{\circ-1}} \cdot U^{a \cdot b} = P \cdot \prod_{j=1}^{\log_2 m} L_j^{x_j^2} \cdot R_j^{x_j^{-2}} . \tag{38}$$

where $\mathbf{s} = \{s_i\}_{i=1}^N, s_i = \prod_{j=1}^{\log_2 m} x_j^{b(i,j)}$ such that $b(i,j)$ is 1 if the j-th bit of $(i-1)$ is 1, and -1 otherwise. Note that $\mathbf{s}$ depends only on the challenges $\{x_j\}_{j=1}^{\log_2 m}$. For the inner product argument associated with $\Pi_{\mathsf{MProve+}}$, substituting the expression of P from (37), we get

$$\mathbf{G}_w^{a \cdot \mathbf{s}} \cdot \mathbf{H}^{b \cdot (\theta \circ \mathbf{s})^{\circ-1}} \cdot U^{a \cdot b} = \left(U^{\hat{t}} (F)^{-r} A S^x \mathbf{G}_w^{\alpha} \mathbf{H}^{\beta} \right) \cdot \prod_{j=1}^{\log_2 m} L_j^{x_j^2} \cdot R_j^{x_j^{-2}} .$$

Moving everything to the LHS, we get

$$\mathbf{G}_w^{a \cdot \mathbf{s} - \alpha} \cdot \mathbf{H}^{b \cdot (\theta \circ \mathbf{s})^{\circ-1} - \beta} \cdot U^{a \cdot b - \hat{t}} \cdot (F)^r \cdot A^{-1} \cdot S^{-x} \cdot \prod_{j=1}^{\log_2 m} L_j^{-x_j^2} \cdot R_j^{-x_j^{-2}} = 1. \tag{39}$$

Furthermore, we merge the verification equation (V3) in (40) using a random scalar $c \leftarrow \mathbb{Z}_q$.

$$\mathbf{G}_w^{a \cdot \mathbf{s} - \alpha} \cdot \mathbf{H}^{b \cdot (\theta \circ \mathbf{s})^{\circ-1} - \beta} \cdot U^{a \cdot b - \hat{t}} \cdot (F)^r \cdot A^{-1} \cdot S^{-x} \cdot \prod_{j=1}^{\log_2 m} L_j^{-x_j^2} \cdot R_j^{-x_j^{-2}} \cdot \left(H^{\hat{t} - \delta} G^{\tau} C_{\text{res}}^{-z^2} T_1^{-x} T_2^{-x^2} \right)^c = 1. \tag{40}$$

Effectively, the verification of an MProve+ boils down to a single multi-exponentiation check of size $\mathcal{O}(2m + 2\log_2 m + 9)$.

IV. Security Properties

The MProve+ protocol has the following security properties.

A. Inflation Resistance

We say that the MProve+ scheme is inflation resistant if no probabilistic polynomial time (PPT) exchange can generate an accepting MProve+ transcript committing to an amount $a'_{\text{res}} \neq \sum_{j=1}^{s} a_j$ as the reserves amount. This is similar to proving that in an Omniring transaction, the sum of inputs is equal to the sum of outputs. Hence the inflation resistant property for MProve+ follows directly from the balance property given in Theorem 4.2 of the Omniring [19] paper.

B. Collusion Resistance

In the MProve+ protocol, for each owned address P, the exchange has to publish a key image I such that the following relation holds,

$$P = G^x \wedge I = (H_p(P))^x, \tag{41}$$

where x is the secret key corresponding to P. Note that for a given P, the key image I in equation (41) is unique. Hence if two PPT exchanges use a common one-time address as a source address to generate reserves proofs, the key image corresponding to that one-time address will appear in both the reserves proofs. Thus a verifier can easily detect collusion between exchanges. If we assume the exchanges are PPT, then they can generate different key images for the same source address only with a negligible probability. This follows from the unforgeability of the argument of knowledge of the MProve+ protocol.

C. *Privacy*

A privacy focused proof of reserves protocol should preserve the privacy of the exchange which it enjoys in the underlying cryptocurrency. The protocol should not also violate the privacy of the entire cryptocurrency network. In the following, we describe how the publication of multiple MProve+ proofs affects the privacy of the exchange as well as the entire Monero network.

Explicit revelation of key images.

A fundamental requirement for a Monero proof of reserves protocol is to show that the source addresses that are used in the proof are not spent already. The simplest way to do it is to reveal the key images corresponding to the source addresses. Any verifier can then check whether the source addresses are unspent by checking if the key images have appeared in the set of already appeared key images $\mathcal{I}$. The reserves proof proposed by Stoffu Noether [24], MProve [7], and MProve+ follow this method. As we discuss below, the privacy of the entire Monero network including the exchange gets affected by this explicit revelation of key images of unspent source addresses. To address this issue with the proof of reserves protocols for Monero, a primitive called *UnspentProof* was proposed by Koe *et al.* [23, Section 8.1.5]. UnspentProof proves that a one-time address is not spent without revealing the corresponding key images. In Appendix A, we discuss the difficulties in using UnspentProof in a privacy focused proof of reserves protocol. We also discuss the other challenges in hiding the key images corresponding to the source addresses in Appendix A.

Privacy implications of publishing a polynomial number of MProve+ proofs.

Because of the challenges discussed in A, the MProve+ protocol publishes the key images of the source addresses explicitly. Let $f(\lambda)$ denote a polynomial of the security parameter λ. Suppose a Monero exchange has generated $f(\lambda)$ MProve+ proofs with the anonymity sets and the key image sets $\{\mathbf{P}^{(i)}\}_{i=1}^{f(\lambda)}$ and $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$ respectively. Let the corresponding cardinalities of those sets be $\{n_i\}_{i=1}^{f(\lambda)}$ and $\{s_i\}_{i=1}^{f(\lambda)}$ respectively.

Now let us consider the information revealed to a PPT adversary who observes $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$ together. First, consider only the ith MProve+ proof. When $(\mathbf{P}^{(i)}, \mathbf{I}^{(i)})$ is revealed together, then it is revealed is that any key image in $\mathbf{I}^{(i)}$ could have originated[1] from any one-time address in $\mathbf{P}^{(i)}$. This information can be represented by a complete bipartite graph[2] with the disjoint sets of vertices $(\mathbf{P}^{(i)}, \mathbf{I}^{(i)})$ and the edge set $\mathbf{P}^{(i)} \times \mathbf{I}^{(i)}$. Here an edge between a one-time address $P \in \mathbf{P}^{(i)}$ and a key image $I \in \mathbf{I}^{(i)}$ denotes that I could have originated from P.

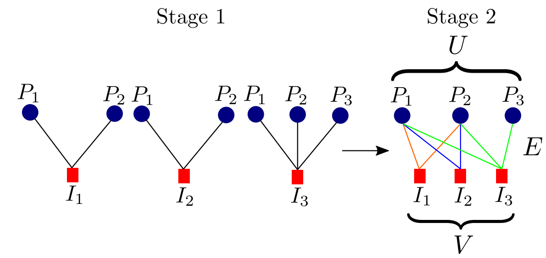

Fig. 5: Illustration of Example 1.

Now consider the case when $f(\lambda)$ MProve+ proofs are published and $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$ are revealed. The information in the individual bipartite graphs can be combined to identify the set of one-time addresses which could have originated a particular key image. For a key image $I \in \{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$, let $\mathcal{P}_{\text{orig}}(I)$ denote the set of one-time addresses of minimal cardinality which could have originated I, from the perspective of a PPT adversary which has access to the Monero blockchain and the MProve+ proofs. We call $\mathcal{P}_{\text{orig}}(I)$ the *originating set* for I. Suppose I has appeared in j_1th, j_2th,..., j_rth proofs among the overall $f(\lambda)$ MProve+ proofs. Then it is obvious that,

$$\mathcal{P}_{\text{orig}}(I) \subset \bigcap_{k=1}^{r} \mathbf{P}^{(j_k)}. \tag{42}$$

Consider the following example.

Example 1: Suppose the adversary observes three MProve+ proofs. The anonymity sets and the key image sets are as follows.

$$\begin{aligned} \mathbf{P}^{(1)} &= \{P_1, P_2\}, & \mathbf{I}^{(1)} &= \{I_1\}, \\ \mathbf{P}^{(2)} &= \{P_1, P_2\}, & \mathbf{I}^{(2)} &= \{I_2\}, \\ \mathbf{P}^{(3)} &= \{P_1, P_2, P_3\}, & \mathbf{I}^{(3)} &= \{I_3\}. \end{aligned}$$

As there are 3 MProve+ proofs, there are 3 corresponding complete bipartite graphs as shown in stage 1 of Figure 5. If we use the intersection formula for $\mathcal{P}_{\text{orig}}(\cdot)$ as given in equation (42), then we get

$$\mathcal{P}_{\text{orig}}(I_3) \subset \mathbf{P}^{(3)} = \{P_1, P_2, P_3\}.$$

But one can see that P_3 is the only possible one-time address which could have possibly originated I_3. This is because $\{P_1, P_2\}$ together have to originate $\{I_1, I_2\}$[3]. This makes P_3 as the only member of $\mathbf{P}^{(3)}$ which could possibly originate I_3.

[1] The statement that the key image I has originated from the one-time address P implies that there exists a scalar $x \in \mathbb{Z}_q$ such that $P = G^x \wedge I = (H_p(P))^x$ holds.

[2] This formulation was introduced in [26].

[3] The set $\{P_1, P_2\}$ is termed as *closed set* in [27]. This kind of structure makes some cover addresses useless in the anonymity sets/rings of transactions.

To get a precise definition of the originating set, we construct the simple[1] bipartite graph (U, V, E), using the $f(\lambda)$ anonymity sets and key image sets. Here U, V are the disjoint vertex sets given by

$$U = \bigcup_{i=1}^{f(\lambda)} \mathbf{P}^{(i)}, \quad V = \bigcup_{i=1}^{f(\lambda)} \mathbf{I}^{(i)},$$

and E is the edge set given by

$$E = \bigcup_{i=1}^{f(\lambda)} \left(\mathbf{P}^{(i)} \times \mathbf{I}^{(i)}\right).$$

Since we are requiring the graph to be simple, the edge set E will not have multiple edges. If an edge appears in both $\mathbf{P}^{(i)} \times \mathbf{I}^{(i)}$ and $\mathbf{P}^{(j)} \times \mathbf{I}^{(j)}$ for $i \neq j$, then we include it only once. The bipartite graph (U, V, E) corresponding to Example 1 is shown in stage 2 of Figure 5. Here the orange edges, blue edges, and green edges of E come from the first, second, and the third proof respectively.

A matching on a graph is a subset of the edge set such that the subset elements have no common vertices [28]. We give the following definition for $\mathscr{P}_{\text{orig}}(I)$.

Definition 1: Let $\mathcal{M}$ be the set of all maximum cardinality matchings on the bipartite graph (U, V, E) induced by the $f(\lambda)$ MProve+ proofs such that for each $M \in \mathcal{M}$ the set of edges $M \bigcap \left(\mathbf{P}^{(i)} \times \mathbf{I}^{(i)}\right)$ is a maximum cardinality matching in the bipartite graph $\left(\mathbf{P}^{(i)}, \mathbf{I}^{(i)}, \mathbf{P}^{(i)} \times \mathbf{I}^{(i)}\right)$ for all $i = 1, 2, \ldots, f(\lambda)$.

We define $\mathscr{P}_{\text{orig}}(I)$ for a key image I in $\bigcup_{i=1}^{f(\lambda)} \mathbf{I}^{(i)}$ as

$$\mathscr{P}_{\text{orig}}(I) = \left\{ P \in \bigcup_{i=1}^{f(\lambda)} \mathbf{P}^{(i)} \middle| (P, I) \text{ belongs to a matching in } \mathcal{M} \right\}.$$

The above definition gives $\mathscr{P}_{\text{orig}}(I_1) = \mathscr{P}_{\text{orig}}(I_2) = \{P_1, P_2\}$ and $\mathscr{P}_{\text{orig}}(I_3) = \{P_3\}$ as desired. Now we give the following theorem which is proved in Appendix B.

Theorem 3: The only information that a PPT adversary can obtain from the $f(\lambda)$ MProve+ proofs is the $f(\lambda)$ bipartite graphs $\left(\mathbf{P}^{(i)}, \mathbf{I}^{(i)}, \mathbf{P}^{(i)} \times \mathbf{I}^{(i)}\right)_{i=1}^{f(\lambda)}$.

Even if we were to disregard the edges in the graph, the key image sets $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$ can affect the privacy of the exchange. For example, when a PPT adversary observes only $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$, the following information is revealed to her.

1) The number of source addresses used in the proofs (cardinalities of $\mathbf{I}^{(i)}$s, i.e. s_is).
2) The number of new source addresses used in the $(i+k)$th proof ($k \geq 1$) which were not there in the ith proof (the number of new key images in $\mathbf{I}^{(i+k)}$ which were not there in $\mathbf{I}^{(i)}$).
3) The number of source addresses in the ith proof which were removed from the $(i+k)$th proof (the number of key image in $\mathbf{I}^{(i)}$ which are not there in $\mathbf{I}^{(i+k)}$).
4) The number of source addresses which are being used repeatedly. For example, consider a key image I which has appeared in $\mathbf{I}^{(i)}$, removed in $\mathbf{I}^{(i+1)}$ onwards, and appears again in $\mathbf{I}^{(i+k)}$. Then the appearance of I reveals that a source address was used in the ith proof, not in use from the $(i+1)$th proof to the $(i+k)$th proof, and was used again in the $(i+k)$th proof.

Further privacy concerns arise when source addresses are spent in future Monero transactions. We discuss this after discussing about the relation between the MProve+ proofs and Monero transactions.

MProve+ proofs and Monero transactions.

We can imagine a MProve+ proof to be a giant Monero transaction where the exchange is accumulating all its owned one-time addresses to generate a Pedersen commitment to the total reserves. However, followings are the major differences between a MProve+ proof and a Monero transaction.

1) A particular key image I can appear only once in a Monero transaction in the main chain[2]. Once I has appeared in the blockchain, the one-time address corresponding to I is considered spent and I is never going to appear in future Monero transactions. However, when I is published in a MProve+ proof, it might occur in future proofs. Here, the one-time address corresponding to I is playing the role of a source address and is not actually being spent.
2) A particular ring of a Monero transaction has a single key image. Any one-time address from the ring could be the originator of the key image. However in a MProve+ proofs, a single anonymity set can have multiple key images. Any of these key images could have been originated from any one-time address in the anonymity set.

In the rings of a Monero transaction, the choice of cover addresses (a.k.a mixins or decoy addresses) is important. Unless the decoy addresses are chosen properly, the source address of a Monero transaction might get de-anonymized. The de-anonymization occurs mainly because of the cascade effect due to zero-mixin transactions (transactions with rings having no decoy addresses). Various analyses have been done in this regard by Moser *et al.* [17] and Kumar *et al.* [16]. The Monero community has taken according countermeasures and further analyses have been carried out [26], [27], [29]. Yu *et al.* [26] propose an *inference attack*, where they propose an active adaptive adversary. This adversary, apart from having view on the blockchain, can generate new transactions, receive payments, and corrupt some decoy addresses in the rings of some transactions. The authors observed that the optimal anonymity (untraceability) cannot be achieved unless there is a centralized strategy for choosing decoy addresses for Monero transactions. In their analysis, the authors also have used matching of bipartite graphs and proposed a novel strategy for choosing decoy addresses for a Monero transaction.

In our analysis, we have shown that multiple MProve+ proofs can reveal the originating set $\mathscr{P}_{\text{orig}}(I)$ for a particular published key image I. While choosing the anonymity sets

[1]By a simple graph, we mean undirected graph with no loops or multiple edges.

[2]Hinteregger *et al.* [29] considered hardforks of Monero where a key image can appear more than once when same address is spent in the main chain and another hard forks.

across multiple proofs, the exchange needs to have a proper strategy similar to the above mentioned strategy. The goal of such a strategy is to make the cardinalities of the originating sets as large as possible. Proposing such a strategy is an interesting direction for future research.

Implication of MProve+ proofs when source addresses are spent in future Monero transactions.

Example 2: Consider a Monero transaction *txn* where a Monero exchange *Ex* is spending from a one-time address P. Before this transaction, the exchange has published some reserves proofs where P has been used as a source address. As a result, the corresponding key image (say I) of P has appeared in those reserves proofs. When P is being spent in *txn*, the same key image I will appear again in *txn*. As the same I has appeared in the reserves proofs published by *Ex* and in *txn*, the fact that *Ex* is spending in *txn* is revealed. This is a privacy drawback from which the exchange as well as the entire Monero network suffer.

The drawback shown in Example 2 exists in every proof of reserves protocol which has to reveal the key images of the source addresses explicitly to prove that they are not spent. The proof of reserves protocol proposed by Stoffu Noether [24], MProve [7], and MProve+ are some examples. Removing this drawback is an open problem because of the challenges discussed in Appendix A. We discuss the case when MProve is used in Example 2. Then we discuss about the improvement we gain when MProve+ is used in place of MProve in Example 2.

1) Effect of MProve on Monero transactions: Example 2 for the case of the MProve protocol has already been considered in Section II.C.1 and the implications have been discussed[1]. In Figure 6, we consider a single MProve proof where there are n key images i.e. $\{I_1, I_2, \ldots, I_n\}$ corresponding to the n linkable ring signatures, where n is the size of the anonymity set. Among these key images, some are real key images (originated from a one-time address) and some are dummy key images (originated from a group element which is not a one-time address). Consider the key image I_j, $j \in [n]$ which is generated from a one-time address P_j. For the PPT adversary $\mathcal{A}$ who observes this MProve proof, I_j could have originated either from P_j or from a group element $C_j' C_j^{-1}$. So in this case we have $\mathscr{P}_{\text{orig}}(I_j) = \{P_j, C_j' C_j^{-1}\}$. When the exchange spends from P_j in a future Monero transaction *txn*, I_j appears again. Let the ring of *txn* be $\mathbf{R}(txn)$. The set $\mathbf{R}(txn)$ must contain P_j as P_j is the source of *txn*. However the group element $C_j' C_j^{-1}$ cannot be an element in the set $\mathbf{R}(txn)$ as it is not a valid one-time address. The view of $\mathcal{A}$ in this situation is shown in terms of bipartite graphs in stage 1 of Figure 6. As any maximum matching on this graph has to match I_j to P_j, $\mathcal{A}$ successfully links I_j with P_j. This has been shown in stage 2 of Figure 6. So the probability that $\mathcal{A}$ successfully outputs P as the originating address for I is,

$$\Pr[\mathcal{A}(I_j, \mathscr{P}_{\text{orig}}(I_j), \mathbf{R}(txn)) = P_j] = 1. \tag{43}$$

[1] P and I in Example 2 are replaced by P_j and I_j respectively in Section II.C.1.

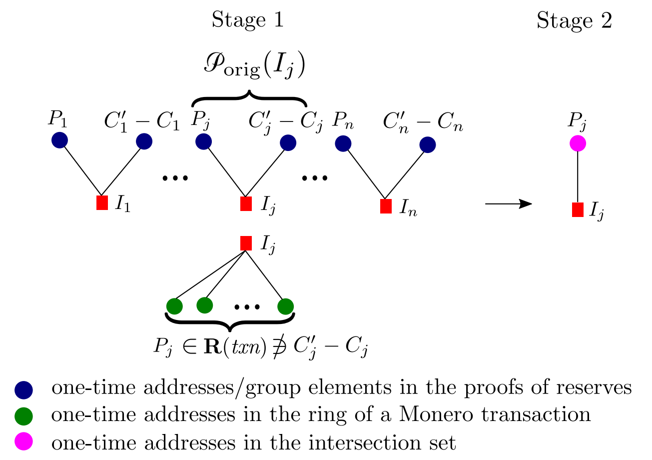

Fig. 6: Linking key image for MProve when a source address is spent.

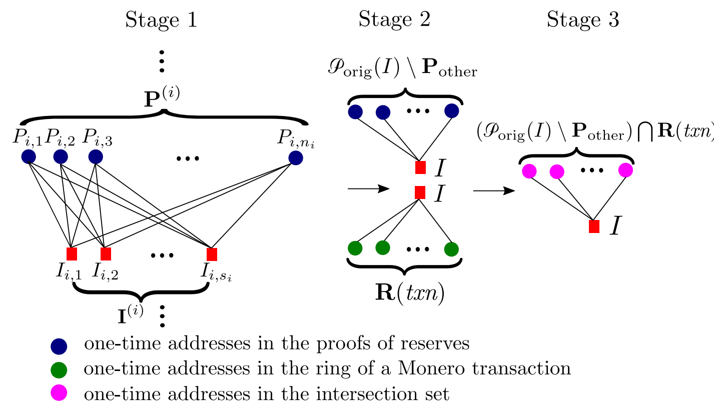

Fig. 7: Linking key image for MProve+ when a source address is spent.

2) Effect of MProve+ on Monero transactions: Now we consider the case when MProve+ is used in Example 2. Suppose *Ex* has used P as a source address in some of the $f(\lambda)$ published MProve+ proofs and I has appeared in some sets in $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$. As discussed above, each MProve+ proof induces a complete bipartite graph. This is shown in stage 1 of Figure 7. From this $f(\lambda)$ MProve+ proofs, the originating set for I i.e. $\mathscr{P}_{\text{orig}}(I)$ is revealed. Let $\mathcal{A}$ be a PPT adversary which wants to obtain the originating address of I (here P) from $\mathscr{P}_{\text{orig}}(I)$. If $\mathcal{A}$ is a participant in the Monero network, then it might have the side information that some addresses in $\bigcup_{i=1}^{f(\lambda)} \mathbf{P}^{(i)}$ do not belong to *Ex* and are definitely cover addresses. We model this side information by the set $\mathbf{P}_{\text{other}} \subset \bigcup_{i=1}^{f(\lambda)} \mathbf{P}^{(i)}$. $\mathcal{A}$ is given access to the set $\mathbf{P}_{\text{other}}$. Now consider the scenario just before a source address spending transaction *txn* appears on the Monero blockchain. $\mathcal{A}$ knows that any address in $\mathbf{P}_{\text{other}}$ cannot be the originating address for I. If we ignore negligible probabilities, the probability that $\mathcal{A}$ successfully outputs P as the originating address for I is given by,

$$\Pr[\mathcal{A}(I, \mathscr{P}_{\text{orig}}(I), \mathbf{P}_{\text{other}}) = P] = \frac{1}{\left|\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right|}. \tag{44}$$

Next, *txn* appears in the Monero blockchain with key image I and ring $\mathbf{R}(txn)$. The view of $\mathcal{A}$ in this situation is shown in stage 2 of Figure 7. With this additional information, $\mathcal{A}$ knows that any address in the set $\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$ could be the originating address corresponding to I. The intersection of the corresponding graphs is shown in stage 3 of Figure 7. Let InfoMPP denote the inputs to $\mathcal{A}$ in this case i.e.,

$$\text{InfoMPP} = (I, \mathcal{P}_{\text{orig}}(I), \mathbf{P}_{\text{other}}, \mathbf{R}(txn)). \tag{45}$$

The equation (44) is modified as follows to give the probability that $\mathcal{A}$ successfully links P with I.

$$\Pr[\mathcal{A}(\text{InfoMPP}) = P] = \frac{1}{\left|\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)\right|}. \tag{46}$$

It is desirable for Ex that the probability given in equation (46) is as low as possible. Assuming that Ex does not have the knowledge of $\mathbf{P}_{\text{other}}$, a strategy for Ex to reduce the probability in equation (46) is as follows.

For a given source address P with the associated key image I, Ex should choose anonymity sets in such a way that $\mathcal{P}_{orig}(I)$ becomes as large as possible. Later when Ex spends from P in txn, Ex should choose $\mathbf{R}(txn)$ as a proper subset of $\mathcal{P}_{orig}(I)$.

Now observe equation (43) and (46). For MProve, no matter how the anonymity set and the ring of the transaction are chosen, the linking probability is always 1. When P is linked with I, it cannot act as a decoy address for any Monero transaction. This is detrimental to the privacy of the entire Monero network because P is still a member of the Monero blockchain from which the cover addresses for a transaction are chosen [23]. When txn appears in the blockchain, the effective ring size of every transaction using P as a cover address reduces by one. However for MProve+, the exchange can choose the anonymity sets and the ring of the transaction carefully and keep the linking probability away from 1. Hence we conclude that MProve+ *is better than* MProve *when the privacy of the entire Monero network including the exchange is of concern.*

MProve+ proofs and untraceability property of Monero. One of the design goals for Monero is to achieve *untraceability*. Roughly speaking, untraceability means that given a transaction ring, no PPT adversary should be able to determine which address in the ring is actually being spent [21]. Now consider a key image I which has appeared across multiple MProve+ proofs. As discussed in Section **??**, when I appears in the source spending transaction txn, it is linked with the source one-time address P if the set $\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$ becomes a singleton set containing only P. The exchange can plausibly avoid this by choosing the ring $\mathbf{R}(txn)$ properly. Hence we conclude that the MProve+ protocol does not preserve the untraceability property of Monero in general. But unlike MProve, it does not always reveal the true source address in a source spending transaction.

MProve+ proofs and the amount confidentiality property of Monero. From Theorem 3, it is verified that the MProve+ protocol does not reveal the amounts corresponding to the addresses in the anonymity sets or the total reserves amount. However, the amount confidentiality is affected when a source spending transaction txn appears in the blockchain. As discussed above, in the extreme scenario when the set $\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$ becomes a singleton set, then the transaction txn becomes traceable. Then the amount confidentiality of the other transactions containing P (the source of txn) in their rings are affected. To make the discussion concrete, suppose the transaction txn' spends from a single input using the ring of m one-time addresses $(P, P_1, P_2, \ldots, P_{m-1})$ with corresponding Pedersen commitments $(C, C_1, C_2, \ldots, C_{m-1})$. For simplicity, suppose txn' has a single output represented by a one-time address P' and Pedersen commitment C'. Before it was revealed that P was spent in transaction txn, the upper bound on the amount in the commitment C' is given by the maximum of the amounts in the commitments $(C, C_1, C_2, \ldots, C_{m-1})$ minus the transaction fees. This is because any of the ring members could have been the true source of funds in the transaction. Once P has been revealed as spent in txn, it cannot be the source of funds in txn'. Thus the upper bound on the amount in the commitment C' is given by the maximum of the amounts in the commitments $(C_1, C_2, \ldots, C_{m-1})$. As this list of commitments is smaller, the upper bound can only be more restrictive.

One might argue that the amounts in the Pedersen commitments are not known. But the amounts in coinbase commitments in Monero are revealed to ensure that miners are creating valid blocks. While the amounts in non-coinbase commitments are not known exactly, they can be upper bounded by identifying the coinbase commitments which could have potentially contributed funds to them. Such an analysis has been demonstrated for the Pedersen commitments in Grin [30].

Even if the set $\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$ is not a singleton set, its size might be less than that of the ring $\mathbf{R}(txn)$. Then it is known that any of the one-time addresses in the set $\mathbf{R}(txn) \setminus \left(\left(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)\right)$ cannot be the source of the transaction txn. Then the amount confidentiality of the transaction txn is affected in the way as described above. Hence we conclude that the MProve+ protocol does not preserve the amount confidentiality property of Monero in general.

MProve+ proofs and the unlinkability property of Monero. Another design goal for Monero is to achieve *unlinkability*. The unlinkability property of Monero implies that a PPT adversary can link a one-time address with its corresponding public key pair only with a negligible probability [21]. Let $\{\mathcal{M}_\lambda\}$ denote a sequence of Monero-like systems whose group sizes (q) and the hash functions ($H_s(\cdot)$) depend on the security parameter λ. We consider one such particular system $\mathcal{M}_\lambda$ from the sequence and define the following `MoneroLink` experiment to precisely characterize the unlinkability property of Monero.

1) An experimenter chooses some scalars $x_0, y_0, x_1, y_1, r \xleftarrow{\$} \mathbb{Z}_q$. She sets two public key pairs $(X_0 = G^{x_0}, Y_0 = G^{y_0}), (X_1 = G^{x_1}, Y_1 = G^{y_1})$, and a random point $R = G^r$.
2) The experimenter selects a bit $b \xleftarrow{\$} \{0, 1\}$. Then she generates a one-time address $P = G^{H_s(X_b^r)} \cdot Y_b$.
3) The experimenter sends $(X_0, Y_0, X_1, Y_1, R, P)$ to a PPT adversary $\mathcal{A}$. The adversary $\mathcal{A}$ outputs $\hat{b}$ as a prediction of b. $\mathcal{A}$ wins if $\hat{b} = b$.

Owing to the unlinkability property of Monero, we have the following lemma.

Lemma 1: For every PPT adversary $\mathcal{A}$ in the `MoneroLink` experiment, there exists a negligible function $\mathsf{negl}(\lambda)$ of the security parameter λ such that the following inequality holds.

$$\left| \Pr[\mathcal{A}(X_0, Y_0, X_1, Y_1, R, P) = b] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda). \quad (47)$$

Next, we propose the following `MPPLink` experiment for the MProve+ protocol.

1) An experimenter chooses some scalars $x_0, y_0, x_1, y_1, r \xleftarrow{\$} \mathbb{Z}_q$. She sets two public key pairs $(X_0 = G^{x_0}, Y_0 = G^{y_0}), (X_1 = G^{x_1}, Y_1 = G^{y_1})$ and a random point $R = G^r$.
2) The experimenter selects a bit $\mathtt{b} \xleftarrow{\$} \{0, 1\}$. Then she generates a one-time address $P = G^{H_s(X_b^r)}.Y_b$. The secret key is $x = H_s(X_b^r) + y_b$.
3) The experimenter produces $f(\lambda)$ MProve+ proofs $\mathsf{MPP}_{\mathsf{act}}$ using the singleton set $\{P\}$ as the anonymity set in all of them. The proofs contain the key image $I = H_p(P)^x$.
4) The experimenter sends $(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{act}})$ to a PPT adversary $\mathcal{B}$. $\mathcal{B}$ outputs $\hat{\mathtt{b}}$ as a prediction of $\mathtt{b}$. $\mathcal{B}$ wins if $\hat{\mathtt{b}} = \mathtt{b}$.

Now we give the following definition.

Definition 2: The MProve+ protocol is said to preserve the unlinkability property of Monero, if for every PPT adversary $\mathcal{B}$ in the `MPPLink` experiment, there exists a negligible function $\mathsf{negl}_1(\lambda)$ of the security parameter λ such that the following inequality holds.

$$\left| \Pr[\mathcal{B}(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{act}}) = b] - \frac{1}{2} \right| \leq \mathsf{negl}_1(\lambda). \quad (48)$$

We give the following theorem which is proved in Appendix C.

Theorem 4: The MProve+ protocol preserves the unlinkability property of Monero in the random oracle model under the DDH assumption and given that Lemma 1 holds.

V. Summary of Contributions

Omniring proposes a transaction scheme for Monero with improved proof size based on the technique from Bulletproofs. The key insight of our paper is that the same technique can remove the privacy drawback of the MProve protocol to some extent. Since it may appear that we have constructed the MProve+ protocol by simply modifying the Omniring protocol, we briefly describe our attempts to construct an even simpler protocol. Instead of the language in equation (13), our first attempt for MProve+ was based on the following language where G_1 is a group element with unknown discrete logarithms with respect to G and H.

$$\mathcal{L} = \left\{ \begin{pmatrix} \mathbf{P}, \mathbf{C}, \mathbf{H}_p, \\ \{I_j\}_{j=1}^s, C_{\text{res}}, \\ G_1 \end{pmatrix} \middle| \begin{array}{l} \exists(\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \gamma) \text{ such} \\ \text{that each } \mathbf{e}_j \text{ is a unit vector,} \\ \mathbf{P}^{\mathbf{e}_j} = G^{x_j}, \\ \mathbf{H}_p^{\mathbf{e}_j} = I_j^{x_j^{-1}} \ \forall j \in [s], \\ G_1^\gamma \prod_{j=1}^{s} \mathbf{C}^{\mathbf{e}_j} = C_{\text{res}}. \end{array} \right\}$$

In $\mathcal{L}$, C_{res} is of the form $G_1^\gamma G^{a_{\text{res}}} H^{r_{\text{res}}}$. Unlike our current proposal, an argument of knowledge for $\mathcal{L}$ does not prove knowledge of the binary representation of a_{res}. It also does not prove the knowledge of the source amounts and blinding factors. Instead, it proves that if a commitment in $\mathbf{C}$ contributes to a_{res} then the prover knows the private key of the corresponding address in $\mathbf{P}$ (through the use of the unit vectors $\mathbf{e}_j$). The G_1^γ term in C_{res} is introduced to prevent observers from estimating the unit vectors. However, the proof generation and verification times for this simpler language turned out to be worse than our present proposal. So we chose the language in equation (13) as the basis for MProve+. We count this exploration of the space of possible languages for MProve+ as our primary contribution.

Our other contribution is a precise characterization of the information revealed by the MProve+ protocol, as stated in Theorem 3. While the notion of the originating set of a key image has appeared in previous work, it had not been explicitly defined as we have done in Definition 1.

Finally, we explain the effect of the MProve+ protocol on the privacy of Monero. We highlight that an exchange using MProve+ must choose the rings of the source spending transactions carefully to avoid affecting transaction untraceability. The interaction between untraceability and amount confidentiality is not well known. Even though Pedersen commitments are perfectly hiding, we explain how a reduction in the effective ring size also reduces the maximum amount of coins which can be stored in the transaction outputs.

VI. Performance

We compare our proof of reserves protocol MProve+ with MProve [7] which is the first and only proof of reserves protocol for Monero that attempts to provide some privacy to the exchange. In both MProve and MProve+, the anonymity set $\mathbf{P}$ is to be revealed as a part of the proof. Suppose the anonymity set size is n and the number of owned addresses is s. The proof sizes of MProve+ and MProve are respectively $(n+s+2\lceil\log_2 m\rceil+4)$ group elements, 5 scalars and $3n+2$ group elements, $6n$ scalars. Here m denotes the length of the witness vectors defined in Figure 1. Figure 8(a) shows the growth of proof sizes with anonymity set size for $s = 100$. Although the proof sizes of both MProve+ and MProve grow linearly, proof size of MProve+ is typically an order of magnitude smaller. For anonymity set size $n = 10^5$ and the number of owned addresses $s = 10^3$, an MProve+ proof size is 3MB as against 29MB for MProve. The difference in proof sizes increases as n grows. If exchanges are required to publish frequent proofs of reserves on a blockchain, protocols with smaller proof sizes will be preferred.

We have implemented MProve+ in Rust over the Ristretto elliptic curve. We demonstrate how Ristretto encoding of existing addresses in Monero could be computed, ensuring adaptability to the existing Monero framework [31]. For fair comparison, we have also implemented MProve over Ristretto. All experiments were run on a 2.6 GHz Intel Core i7 desktop with 8GB RAM. Our code is open-sourced on GitHub [32], [33].

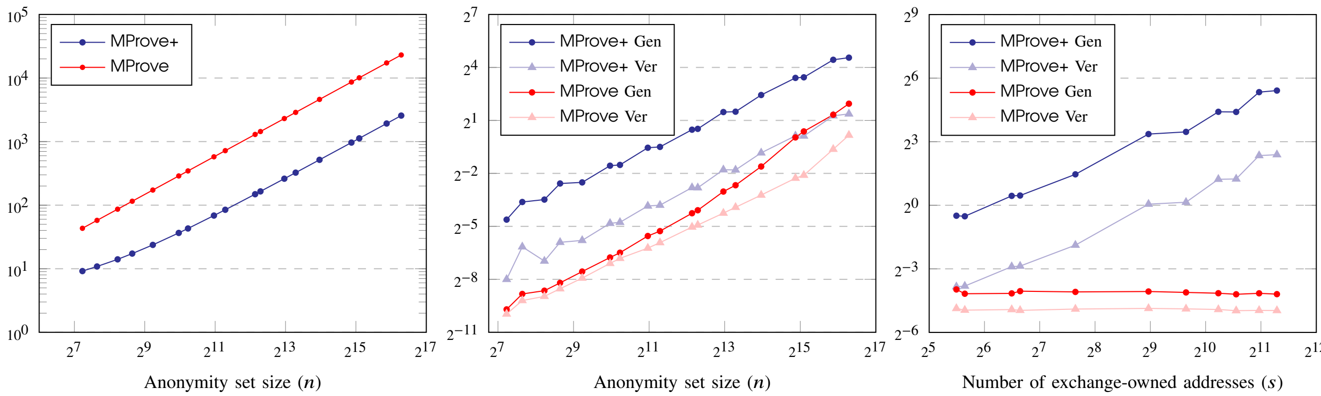

Fig. 8: Performance comparison of MProve+ and MProve for $\mathbb{G}$ = Ristretto elliptic curve.

Figure 8(b) shows the proof generation and verification times of MProve+ and MProve. For a constant s, we see a linear growth of proof generation and verification times of MProve as well as MProve+ with the anonymity set size n. Since the inner product protocol requires witness sizes to be a power of 2, the witness vectors of MProve+ in Figure 1 are appended with 0's to convert their size to the next power of 2. For witness sizes $m_1 \neq m_2$ such that $\lceil \log_2 m_1 \rceil = \lceil \log_2 m_2 \rceil$, the timings in the two cases will not be much different. Therefore, we observe a step-wise increment in the generation and verification timings of MProve+. An exchange owning 1000 addresses and wishing to have 49000 cover addresses would spend about 150 minutes in a MProve+ proof generation and the proof verification would take 20 minutes. An MProve proof of same configuration would take a minute for generation and verification each. Although the proof generation time for MProve+ is significantly higher than that of MProve owing to the greater number of group operations, the timings are not unreasonable for practical deployment. The verification of an MProve+ proof is around 8X faster than its generation because the inner product protocol can be verified using a single multi-exponentiation of size $\mathcal{O}(2s \cdot n + 2\log(s \cdot n))$ as explained in Section III.F. Faster verification enables customers of an exchange to verify the proofs without much computational cost and specialized hardware. From the perspective of an exchange, the privacy benefits combined with the smaller proof sizes of MProve+ overshadow the higher computational cost in using it.

A notable difference between the MProve+ and MProve protocol is that in MProve+, we reveal the number of addresses an exchange owns. While this may seem like a privacy concern, an exchange can create some addresses which have zero amount in them for the purpose of *padding* the number of owned addresses. An implication of revealing the number of owned addresses s is that the proof size as well as generation and verification times depend on the number of exchange-owned addresses. Figure 8(c) shows the dependence of generation and verification timings of MProve+ and MProve with respect to the number of owned addresses and for a constant anonymity set size. While timings for MProve remain constant, MProve+ timings grow linearly with s.

VII. Conclusion

We present the MProve+ protocol which gives better privacy than the MProve protocol using techniques of Bulletproofs [18] and Omniring [19]. The MProve+ protocol provides a significant improvement in terms of proof size over the MProve protocol. The performance of MProve+ protocol is also practical in terms of the proof generation time and verification time. Like the MProve protocol, when an exchange spends from a source address used in MProve+ proofs, it is revealed that the exchange is spending in the transaction. This is because of the explicit revelation of the key images of the source addresses. Removing this drawback remains as a open problem because of the challenges discussed in Appendix A. However, unlike the MProve protocol, the MProve+ protocol does not let a source spending transaction become a zero-mixin transaction given that the exchange chooses the ring of the transaction carefully. Hence the MProve+ protocol does a better job than the MProve protocol in preserving the privacy of the exchange as well as the entire Monero network.

VIII. Acknowledgments

We acknowledge the support of the Bharti Centre for Communication at IIT Bombay.

Appendix A
Difficulties in Hiding the Key Images of Source Addresses

UnspentProof. Consider a Monero user Bob who owns a one-time address P which is not spent. Bob wants to show that P is not spent without revealing the corresponding key image $I = H_p(P)^x$, where x is the secret key of P i.e. $P = G^x$. Suppose the public key pair of Bob is $(B_{\text{vk}}, B_{\text{sk}}) = (G^{b_{\text{vk}}}, G^{b_{\text{sk}}})$, where b_{vk} and b_{sk} are the secret view key and the secret spend key respectively. Let the Diffie-Hellman shared secret corresponding to P be B_{vk}^r, where $r \in \mathbb{Z}_q$. Then the secret key corresponding to P is $x = H(B_{\text{vk}}^r) + b_{\text{sk}}$ and we have,

$$I = H_p(P)^{H(B_{\text{vk}}^r)+b_{\text{sk}}}. \tag{49}$$

We say that a key image I is *originated* from a one-time address P if $P = G^x \wedge I = H_p(P)^x$ for the same private key x. In UnspentProof, the key images for all transactions where P has appeared as a ring member are tested. Suppose the set $\mathrm{TX}(P) = \{txn_1, txn_2, \ldots, txn_n\}$ represents the set of all transactions where P has appeared as a ring member. Each transaction in $\mathrm{TX}(P)$ has one or more key images. UnspentProof proves that each such key image in $\mathrm{TX}(P)$ is not originated from P. In this way, P is proved to be unspent without revealing I. However, to execute the proof, the verifier must know the Diffie-Hellman shared secret B^r_{vk}. Let $I_?$ be a key image which has appeared in one of the transactions in $\mathrm{TX}(P)$. Using $I_?$, B^r_{vk}, and P, the verifier computes the following quantity, termed as the *partial spend image*.

$$I_{?,s} = I_? H_p(P)^{-H(B^r_{\mathrm{vk}})}. \tag{50}$$

From Equation (49) and (50), if $I = I_?$ i.e. P is spent in the transaction being tested, then we have,

$$I_{?,s} = H_p(P)^{b_{\mathrm{sk}}}. \tag{51}$$

UnspentProof consists of two multi-base proof of knowledge signatures which are non-interactive Schnorr-like proofs [34], [35]. They involve sets of base elements and public keys. First, there is a three-base signature σ_3 with a base set $\{G, B_{\mathrm{sk}}, I_{?,s}\}$. Let the set of public keys for σ_3 be $\{Q, R, S\}$. The signature σ_3 proves the knowledge of a secret scalar k such that the following holds,

$$Q = G^k \wedge R = B^k_{\mathrm{sk}} \wedge S = I^k_{?,s}. \tag{52}$$

Here, σ_3 is signed with b_{sk} i.e. $k = b_{\mathrm{sk}}$. So we have the set of public keys $\{Q, R, S\} = \{B_{\mathrm{sk}}, B^{b_{\mathrm{sk}}}_{\mathrm{sk}}, I^{b_{\mathrm{sk}}}_{?,s}\}$. There is also a two-base signature σ_2 with a base set $\{G, H_p(P)\}$. Signature σ_2 proves knowledge of a secret scalar k' such that,

$$X = G^{k'} \wedge Y = H_p(P)^{k'}, \tag{53}$$

where $\{X, Y\}$ is the set of public keys for σ_2. Here, σ_2 is signed with $b_{\mathrm{sk}} * b_{\mathrm{sk}}$ i.e. $k' = b_{\mathrm{sk}} * b_{\mathrm{sk}}$. So the set of public keys for σ_2 is $\{X, Y\} = \{B^{b_{\mathrm{sk}}}_{\mathrm{sk}}, H_p(P)^{b_{\mathrm{sk}} * b_{\mathrm{sk}}}\}$. When σ_3 is signed with b_{sk} and σ_2 is signed with $b_{\mathrm{sk}} * b_{\mathrm{sk}}$, we have $S = I^{b_{\mathrm{sk}}}_{?,s}$ and $Y = H_p(P)^{b_{\mathrm{sk}} * b_{\mathrm{sk}}}$. From Equation (51), it follows $S = Y$ only if $I = I_?$. The UnspentProof protocol proceeds as follows.

1) Both the prover and the verifier compute the base sets of σ_3 and σ_2.
2) The prover generates the signatures σ_3 and σ_2 and sends them to the verifier along with the associated sets of public keys i.e. $\{Q, R, S\}$ and $\{X, Y\}$.
3) The verifier checks if (a) σ_3 and σ_2 are correct, and (b) $R = X$. The conditions (a) and (b) ensure that σ_3 is signed with b_{sk} and σ_2 is signed with $b_{\mathrm{sk}} * b_{\mathrm{sk}}$. If any of them does not hold, the verifier rejects the proof. Otherwise she proceeds to the next step.
4) If $S \neq Y$, the verifier declares that P is not spent in the transaction under test i.e. $I_? \neq I$. If $S = Y$, P is declared to be spent.

The same procedure is followed for every key image of every transaction in $\mathrm{TX}(P)$. In each case, the same σ_2 can be used whereas σ_3 changes every time. If the verification passes for every transaction in $\mathrm{TX}(P)$, then P is declared to be unspent. As any PPT adversary can forge a proof of knowledge signature only with a negligible probability, P is an unspent address with a probability overwhelmingly close to 1.

Hence by UnspentProof, we can show that a particular one-time address is not spent without revealing its key image. However, to execute UnspentProof, the prover must reveal the Diffie-Hellman shared secret (B^r_{vk} in the above example) to let the verifier calculate the partial spend image ($I_{?,s}$ in the above example). The verifier needs to calculate the partial spend image i.e. base of σ_3 herself. Suppose in the above example the verifier does not compute the partial spend image herself and the prover sends it to her. This allows the prover to cheat by sending any element other than $I_? H_p(P)^{-H(B^r_{\mathrm{vk}})}$ as $I_{?,s}$. Then even if P is spent in the transaction, σ_3 and σ_2 would be correct and $R = X$, $S \neq Y$ would hold. Hence the verifier will declare P is unspent in spite of P being spent in the transaction.

In spite of the novelty and the simplicity, UnspentProof has the following privacy concerns.

- To execute UnspentProof, revealing the Diffie-Hellman shared secret B^r_{vk} is unavoidable. However an entity which gets to know the Diffie-Hellman shared secret corresponding to a one-time address can easily obtain the amount associated with the one-time address using the method discussed in Section II.B. If UnspentProof is to be used in the MProve+ protocol to prove that the source addresses are not spent, then the corresponding Diffie-Hellman shared secrets have to be revealed. Using those secrets, any adversary can calculate the total reserves amount.
- The base set of the signature σ_3 contains B_{sk}. Also to prove that the Diffie-Hellman shared secret B^r_{vk} is authentic, the prover Bob needs to produce a two-base signature with the secret key b_{vk}, the base set $\{G, R\}$ and the public key set $\{G^{b_{\mathrm{vk}}}, R^{b_{\mathrm{vk}}}\} = \{B_{\mathrm{vk}}, B^r_{\mathrm{vk}}\}$. This signature reveals B_{vk}. So to prove that Bob owns a unspent one-time address P using UnspentProof, Bob needs to reveal the public key pair $\{B_{\mathrm{vk}}, B_{\mathrm{sk}}\}$ corresponding to the one-time address P. This leads to the violation of the unlinkability that the exchange enjoys in Monero network.

The above mentioned reasons forbid the MProve+ protocol to use UnspentProof as a primitive.

Zero-knowledge set non-membership proof. Another possible way to avoid revealing key images in proof of reserves protocols is to propose a zero-knowledge set non-membership proof. Let the set of key images which have appeared on the Monero blockchain be $\mathcal{I}$. Recall that the set of key images of the exchange-owned source addresses is defined as $\mathbf{I}$ and the anonymity set of one-time addresses is defined as $\mathbf{P}$. We need a proof of reserves protocol which satisfies the following requirements.

1) The protocol needs a scheme that makes the set $\mathbf{I}$, a *zero-knowledge* set so that any information regarding its elements or size is not revealed. The verifier should be able to verify that no element in $\mathcal{I}$ is a member of this set $\mathbf{I}$.
2) For each key image $I_i \in \mathbf{I}$ and some $P_i \in \mathbf{P}$, the prover should be able to show the knowledge of the secret key

$x_i \in \mathbb{Z}_q$ such that $P_i = G^{x_i} \wedge I_i = H_p(P_i)^{x_i}$ hold. The verifier also needs to ensure that all key images in $\mathbf{I}$ are distinct. This is to ensure that no source amount is used more than once in calculating the total reserves amount. So we need a zero-knowledge set non-membership proof and also a proof that the elements of the zero-knowledge set satisfy certain algebraic properties.

3) Since each transaction input in Monero has a corresponding key image, the set $\mathcal{I}$ is large and increases monotonically. As the non-membership proof has to be given for all elements in $\mathcal{I}$, the proof should be practical in terms of the size, the generation time, and the verification time.[1]
4) The security of Monero is based on the discrete logarithm assumption, the decisional Diffie-Hellman assumption, and the random oracle assumption for hash functions. Monero does not need any trusted setup. So it is desirable that a proof of reserves protocol does not use any primitive based on some other assumptions or a trusted setup.

We are not aware of any scheme which meets all the criteria mentioned above.

Appendix B
Proof of Theorem 3

Proof: Let the $f(\lambda)$ MProve+ proofs $\{\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}, \mathbf{I}^{(i)}, C_{\text{res}}^{(i)}, \Pi_{\mathsf{MPP}}^{(i)}\}_{i=1}^{f(\lambda)}$ be denoted by $\mathsf{MPP}_{\mathsf{act}}$. We can extract the $f(\lambda)$ bipartite graphs $\left(\mathbf{P}^{(i)}, \mathbf{I}^{(i)}, \mathbf{P}^{(i)} \times \mathbf{I}^{(i)}\right)_{i=1}^{f(\lambda)}$ from the $f(\lambda)$ anonymity sets and the key image sets i.e. $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$. To prove that this is the sole information that can be extracted from $\mathsf{MPP}_{\mathsf{act}}$ by a PPT adversary, we construct a simulator $\mathcal{S}_{\mathsf{MPP}}$ as follows. $\mathcal{S}_{\mathsf{MPP}}$ is given only $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$. $\mathcal{S}_{\mathsf{MPP}}$ publishes $f(\lambda)$ simulated MProve+ proofs (say $\mathsf{MPP}_{\mathsf{sim}}$) keeping $\left\{\mathbf{P}^{(i)}\right\}_{i=1}^{f(\lambda)}$ as the anonymity sets. $\mathcal{S}_{\mathsf{MPP}}$ replaces the elements in $\left\{\mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$ with uniform group elements $\left\{\mathbf{I}'^{(i)}\right\}_{i=1}^{f(\lambda)}$ keeping the structures of the bipartite graphs induced by $\mathsf{MPP}_{\mathsf{act}}$ as it is. We construct $\mathcal{S}_{\mathsf{MPP}}$ as follows.

1) From the given input $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$, $\mathcal{S}_{\mathsf{MPP}}$ calculates $\left(\{\mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}\}_{i=1}^{f(\lambda)}\right)$ using the Monero blockchain and the hash functions used in Monero.
2) $\mathcal{S}_{\mathsf{MPP}}$ generates the simulated key images i.e. $\{\mathbf{I}'^{(i)}\}_{i=1}^{f(\lambda)}$ from $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$ as follows. It chooses $\mathbf{I}'^{(1)}$ by sampling s_1 uniform group elements. Suppose in $\mathbf{I}'^{(1)}$, $I'_{1,k}$ is located in the position of $I_{1,k}$ in $\mathbf{I}^{(1)}$ ($k \in [s_1]$). If $I_{1,k}$ is repeated in $\mathbf{I}^{(j_1)}, \mathbf{I}^{(j_2)}, \ldots, \mathbf{I}^{(j_r)}$, $I'_{1,k}$ is placed in $\mathbf{I}'^{(j_1)}, \mathbf{I}'^{(j_2)}, \ldots, \mathbf{I}'^{(j_r)}$ in the same position where $I_{1,k}$ is located in those vectors. This is followed for all such k for which $I_{1,k}$ is repeated in the subsequent proofs. Similar procedure is followed sequentially from $i = 2$ to $f(\lambda)$. For example, $\mathbf{I}^{(j)}$ is filled only after filling $\mathbf{I}^{(1)}, \mathbf{I}^{(2)}, \ldots, \mathbf{I}^{(j-1)}$. Uniform group elements are placed in the locations of $\mathbf{I}^{(j)}$ which are not filled up already due to repetition. If any $I_{j,k} \in \mathbf{I}^{(j)}(k \in [s_j])$ repeats in subsequent proofs, $I'_{j,k}$ is placed in those positions.
3) $\mathcal{S}_{\mathsf{MPP}}$ sets $C'^{(i)}_{\text{res}} \xleftarrow{\$} \mathbb{G}$ for each $i \in [f(\lambda)]$.
4) For each $i \in [f(\lambda)]$, $\mathcal{S}_{\mathsf{MPP}}$ computes $(u_i, v_i, w_i, y_i, z_i, x_i) \xleftarrow{\$} \mathbb{Z}_q$ and sets $\mathsf{stmt}^{(i)} = \left(\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}, \mathbf{I}'^{(i)}, C'^{(i)}_{\text{res}}\right)$ and the challenges as $(u_i, v_i, w_i, y_i, z_i, x_i)$. $\mathcal{S}_{\mathsf{MPP}}$ samples $A_{\mathsf{S}}^{(i)}, T_{2,\mathsf{S}}^{(i)} \xleftarrow{\$} \mathbb{G}, \boldsymbol{\ell}_{\mathsf{S}}^{(i)}, \boldsymbol{r}_{\mathsf{S}}^{(i)} \xleftarrow{\$} \mathbb{Z}_q^m, \tau_{\mathsf{S}}^{(i)}, r_{\mathsf{S}}^{(i)} \xleftarrow{\$} \mathbb{Z}_q$. It computes $\hat{t}_{\mathsf{S}}^{(i)} = \langle \boldsymbol{\ell}_{\mathsf{S}}^{(i)}, \boldsymbol{r}_{\mathsf{S}}^{(i)} \rangle$. It then computes $S_{\mathsf{S}}^{(i)}$ and $T_{1,\mathsf{S}}^{(i)}$ as follows.
$$S_{\mathsf{S}}^{(i)} = \left((F)^{-r} A_{\mathsf{S}}^{(i)} \mathbf{G}_{w_i}^{\boldsymbol{\alpha}^{(i)} - \boldsymbol{\ell}_{\mathsf{S}}^{(i)}} \mathbf{H}^{\boldsymbol{\beta}^{(i)} - \boldsymbol{\theta}^{(i)\circ -1} \circ \boldsymbol{r}_{\mathsf{S}}^{(i)}}\right)^{-\frac{1}{x_i}}, \tag{54}$$
$$T_{1,\mathsf{S}}^{(i)} = \left(H^{\delta_i - \hat{t}_{\mathsf{S}}^{(i)}} G^{-\tau_{\mathsf{S}}^{(i)}} {C'^{(i)}_{\text{res}}}^{-z_i^2} {T_{2,\mathsf{S}}^{(i)}}^{x_i^2}\right)^{-\frac{1}{x_i}}, \tag{55}$$
where the values of $\boldsymbol{\alpha}^{(i)}, \boldsymbol{\beta}^{(i)}, \boldsymbol{\theta}^{(i)}$, and δ_i are calculated following the definitions given in Figure 3.
$\mathcal{S}_{\mathsf{MPP}}$ obtains the ith transcript as $\Pi_{\mathsf{MPP,S}}^{(i)} = (A_{\mathsf{S}}^{(i)}, S_{\mathsf{S}}^{(i)}, T_{1,\mathsf{S}}^{(i)}, T_{2,\mathsf{S}}^{(i)}, \boldsymbol{\ell}_{\mathsf{S}}^{(i)}, \boldsymbol{r}_{\mathsf{S}}^{(i)}, \hat{t}_{\mathsf{S}}^{(i)}, \tau_{\mathsf{S}}^{(i)}, r_{\mathsf{S}}^{(i)})$.
5) $\mathcal{S}_{\mathsf{MPP}}$ outputs the simulated proof as $\mathsf{MPP}_{\mathsf{sim}} = \{\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}, \mathbf{I}'^{(i)}, C'^{(i)}_{\text{res}}, \Pi_{\mathsf{MPP,S}}^{(i)}\}_{i=1}^{f(\lambda)}$.

Because of step 2 of the construction of $\mathcal{S}_{\mathsf{MPP}}$, the $f(\lambda)$ bipartite graphs which can be extracted from $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$ and $\left\{\mathbf{P}^{(i)}, \mathbf{I}'^{(i)}\right\}_{i=1}^{f(\lambda)}$ are the same except having one set of different disjoint vertices (key images). To show that this is the sole information that can be extracted from $\mathsf{MPP}_{\mathsf{act}}$, we propose the following privacy experiment `MPPPriv` for the MProve+ scheme as follows.

1) $\mathcal{S}_{\mathsf{MPP}}$ sets $\mathsf{MPP}_0 = \mathsf{MPP}_{\mathsf{sim}}$ and $\mathsf{MPP}_1 = \mathsf{MPP}_{\mathsf{act}}$.
2) $\mathcal{S}_{\mathsf{MPP}}$ chooses a bit $b \xleftarrow{\$} \{0,1\}$ randomly.
3) $\mathcal{S}_{\mathsf{MPP}}$ sends MPP_b to $\mathcal{D}_{\mathsf{MPP}}$.
4) $\mathcal{D}_{\mathsf{MPP}}$ outputs a bit $\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_b)$ as a prediction of b.

Note that MPP_0 is computed using no other information in MPP_1 except the $f(\lambda)$ bipartite graphs extracted from $\left\{\mathbf{P}^{(i)}, \mathbf{I}^{(i)}\right\}_{i=1}^{f(\lambda)}$. If no PPT adversary in the above experiment can successfully predict b with a probability non-negligibly better than $\frac{1}{2}$, then Theorem 3 holds. Now we make the following claim and prove it.

Claim 1: For every PPT $\mathcal{D}_{\mathsf{MPP}}$ in the `MPPPriv` experiment, there exists a negligible function $\mathsf{negl}(\lambda)$ of the security parameter λ such that,
$$\left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_b) = b] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda). \tag{56}$$

The LHS of the inequality (56) can be alternatively repre-

[1] As of July 24, 2020, there are about 3.58×10^7 key images in the Monero blockchain [36]. Also, the average number of transactions per day (in the last one year) in Monero is ≈ 9000 [37]. Since each transaction contains 2 inputs on an average, the set $\mathcal{I}$ grows approximately by 18,000 every day.

sented as,

$$
\begin{aligned}
&\left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_b) = b] - \frac{1}{2} \right| \\
=&\Big| \Pr[b = 0] \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_b) = b | b = 0] + \\
&\quad \Pr[b = 1] \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_b) = b | b = 1] - \frac{1}{2} \Big|, \\
=&\left| \frac{1}{2} \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_0) = 0] + \frac{1}{2} \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_1) = 1] - \frac{1}{2} \right|, \\
=&\left| \frac{1}{2} (1 - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_0) = 1]) + \frac{1}{2} \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_1) = 1] - \frac{1}{2} \right|, \\
=& \frac{1}{2} \left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_0) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_1) = 1] \right|, \\
=& \frac{1}{2} \left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{act}}) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{sim}}) = 1] \right|.
\end{aligned} \tag{57}
$$

By equation (57), Claim 1 can be alternatively represented as,

Claim 2: For any PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a negligible function $\mathsf{negl}_1(\lambda)$, such that

$$
\left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{act}}) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{sim}}) = 1] \right| \le \mathsf{negl}_1(\lambda). \tag{58}
$$

Let us consider the elements of $\mathsf{MPP}_{\mathsf{act}}$ and $\mathsf{MPP}_{\mathsf{sim}}$. For each $i \in f(\lambda)$, $\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}$ are common in both of them. $C_{\text{res}}^{(i)}$ and $C'^{(i)}_{\text{res}}$ are distributed uniformly in $\mathbb{G}$. By observation, the elements of the transcript $\Pi^{(i)}_{\mathsf{MPP,S}}$ and $\Pi^{(i)}_{\mathsf{MPP}}$ are identically distributed[1].

Let us consolidate all the distinct elements of $\bigcup_{i=1}^{f(\lambda)} \mathbf{I}'^{(i)}$ and $\bigcup_{i=1}^{f(\lambda)} \mathbf{I}^{(i)}$ in the vectors $\mathbf{I}_c$ and $\mathbf{I}'_c$ respectively in a lexicographic order. Let

$$
\mathbf{I}_c = \{I_1, I_2, \dots, I_N\}, \tag{59}
$$

$$
\mathbf{I}'_c = \{I'_1, I'_2, \dots, I'_N\}, \tag{60}
$$

where $N = \left| \bigcup_{i=1}^{f(\lambda)} \mathbf{I}^{(i)} \right|$. Let us define the following sets,

$$
\mathcal{P}_{\text{orig}} = \{\mathcal{P}_{\text{orig}}(I_1), \mathcal{P}_{\text{orig}}(I_2), \dots, \mathcal{P}_{\text{orig}}(I_N)\}, \tag{61}
$$

$$
\mathbf{H}_p(\mathcal{P}_{\text{orig}}) = \{\mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_1)), \mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_2)), \dots, \mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_N))\}, \tag{62}
$$

where the vector $\mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_k))$, $k \in [N]$ contains the hashes of the elements of the set $\mathcal{P}_{\text{orig}}(I_k)$. Because of the way $\mathbf{I}'^{(i)}$s are populated (discussed in step 2 of the construction of $\mathsf{MPP}_{\mathsf{sim}}$), $\big(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}})\big)$ are the same for both $\mathbf{I}_c$ and $\mathbf{I}'_c$. We also have,

$$
\mathcal{P}_{\text{orig}}(I_k) = \mathcal{P}_{\text{orig}}(I'_k), \quad \forall I_k \in \mathbf{I}_c, I'_k \in \mathbf{I}'_c, k \in [N]. \tag{63}
$$

From the above discussion, it is clear that to prove Claim 2, it is enough[2] to prove that for every PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a negligible function $\mathsf{negl}'(\lambda)$ such that,

$$
\begin{aligned}
\Big| &\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c) = 1] \\
&- \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}'_c) = 1] \Big| \le \mathsf{negl}'(\lambda).
\end{aligned} \tag{64}
$$

Consider the set $\mathcal{P}_{\text{orig}}(I_j), \mathcal{P}_{\text{orig}}(I'_j) \in \mathcal{P}_{\text{orig}}$, where $I_j \in \mathbf{I}_c$ and $I'_j \in \mathbf{I}'_c$ ($\mathcal{P}_{\text{orig}}(I_j) = \mathcal{P}_{\text{orig}}(I'_j)$ as discussed above). Let the set $\mathcal{P}_{\text{orig}}(I_j)$ be,

$$
\mathcal{P}_{\text{orig}}(I_j) = (P_1, P_2, \dots, P_{o_j}), \quad o_j \in \mathbb{Z}_q. \tag{65}
$$

There exists a secret index $m_j \in [o_j]$ for which the following equation holds.

$$
P_{m_j} = G^{x_{m_j}} \wedge I_j = H_p(P_{m_j})^{x_{m_j}}. \tag{66}
$$

Let $H_p(P_{m_j}) = G^{y_j}$ for some $y_j \in \mathbb{Z}_q$. Then we have the following decisional Diffie-Hellman (DDH) triple from Equation (66),

$$
\left(P_{m_j} = G^{x_{m_j}}, \quad H_p(P_{m_j}) = G^{y_j}, \quad I_j = G^{x_{m_j} y_j} \right). \tag{67}
$$

However when I_j is replaced by $I'_j = G^{z_j}$ (say), $z_j \in \mathbb{Z}_q$, then the triple $(P_{m_j}, H_p(P_{m_j}), I'_j)$ is not a DDH triple for any $m_j \in [o_j]$. Hence the collection $(\mathcal{P}_{\text{orig}}(I_j), \mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_j)), I_j)$ contains a single DDH triple for a secret combination $(P_{m_j}, H_p(m_j), I_j)$ for all $I_j \in \mathbf{I}_c$. So there are N such DDH triples in $(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c)$. However, with an overwhelming probability there is no such DDH triple in $(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}'_c)$. Suppose for a PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a polynomial $p(\lambda)$ such that,

$$
\begin{aligned}
\Big| &\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c) = 1] \\
&- \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}'_c) = 1] \Big| \ge \frac{1}{p(\lambda)}.
\end{aligned} \tag{68}
$$

So $\mathcal{D}_{\mathsf{MPP}}$ can distinguish between the two above scenarios with a non-negligible probability. We will show how to construct a DDH adversary $\mathcal{D}_{\mathsf{DDH}}$ using $\mathcal{D}_{\mathsf{MPP}}$ as a subroutine. A DDH challenger $\mathcal{C}$ samples $b \xleftarrow{\$} \{0, 1\}, x, y, z \xleftarrow{\$} \mathbb{Z}_q$. $\mathcal{C}$ sets $X = G^x, Y = G^y, Z = G^{z_b}$, where $z_0 = z, z_1 = xy$. $\mathcal{C}$ sends (X, Y, Z) to $\mathcal{D}_{\mathsf{DDH}}$. $\mathcal{D}_{\mathsf{DDH}}$ outputs b' as the estimate of b. $\mathcal{D}_{\mathsf{DDH}}$ wins if $b' = b$. We construct $\mathcal{D}_{\mathsf{DDH}}$ using a hybrid argument.

Consider a hybrid simulator $\mathcal{S}_{\text{hyb}}^{(n)}$ which is given $(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c)$, $n \in \{0, 1, 2, \dots, N\}$. $\mathcal{S}_{\text{hyb}}^{(n)}$ works as follows.

1) It keeps the first n elements of $\mathbf{I}_c$ as it is.
2) It sets the $n+1$ to N elements of $\mathbf{I}_c$ as uniform elements of $\mathbb{G}$. Let the modified $\mathbf{I}_c$ be $\mathbf{I}_c^{(n)}$.
3) It outputs $(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c^{(n)})$.

By observation we have,

$$
\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c) = 1] = \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(N)}) = 1], \tag{69}
$$

$$
\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}'_c) = 1] = \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(0)}) = 1]. \tag{70}
$$

The construction of $\mathcal{D}_{\mathsf{DDH}}$ having (X, Y, Z) as input is as follows.

[1] The elements are either uniformly distributed in $\mathbb{G}$ and $\mathbb{Z}_q$ or fixed by the verification equations (V1),(V2), and (V3).

[2] Some of the vertices containing one-time addresses and edges of the bipartite graphs are removed while constructing the originating sets. The removed vertices and edges are the same for both the cases. Hence we can ignore them as well.

1) It queries $\mathcal{D}_{\mathsf{MPP}}$ and obtains N where $2 \le N \le \sum_{i=1}^{f(\lambda)} n_i$.
2) It randomly chooses $k^*, o_1, o_2, \ldots, o_N \xleftarrow{\$} [N]$. It then chooses $m_1 \xleftarrow{\$} [o_1], m_2 \xleftarrow{\$} [o_2], \ldots, m_{k^*} \xleftarrow{\$} [o_{k^*}]$.
3) It defines the following sets for all $j \in [N]$,
$$\mathcal{P}_{\text{orig}}(I_j) = \{P_{j,1}, P_{j,2}, \ldots, P_{j,o_j}\},$$
$$\mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_j)) = \{Q_{j,1}, Q_{j,2}, \ldots, Q_{j,o_j}\}.$$
4) It chooses $x_1, y_1, x_2, y_2, \ldots, x_{k^*-1}, y_{k^*-1} \xleftarrow{\$} \mathbb{Z}_q$. For all $j \in [k^* - 1]$, it sets,
$$P_{j,m_j} = G^{x_j}, \quad Q_{j,m_j} = G^{y_j}, \quad I_j = G^{x_j y_j}.$$
It also sets,
$$P_{k^*,m_{k^*}} = X, \quad Q_{k^*,m_{k^*}} = Y, \quad I_{k^*} = Z.$$
5) It sets $I_{k^*+1}, I_{k^*+2}, \ldots, I_N$ as uniform elements from $\mathbb{G}$. It populates other elements of $(\mathcal{P}_{\text{orig}}(I_j), \mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_j)))$ with uniform elements from $\mathbb{G}$ for all $j \in [N]$.
6) It defines the following sets,
$$\mathbf{I}_{\text{DDH}} = \{I_1, \ldots, I_N\}$$
$$\mathcal{P}_{\text{orig,DDH}} = \{\mathcal{P}_{\text{orig}}(I_1), \ldots, \mathcal{P}_{\text{orig}}(I_N)\}$$
$$\mathbf{H}_p(\mathcal{P}_{\text{orig,DDH}}) = \{\mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_1)), \ldots, \mathbf{H}_p(\mathcal{P}_{\text{orig}}(I_N))\}.$$
7) It sends $(\mathcal{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathcal{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}})$ to $\mathcal{D}_{\mathsf{MPP}}$ and receives b' as output. It sends b' as its response to the challenger $\mathcal{C}$.

Now we have,
$$\begin{aligned}
&\Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 0\right] \\
&= \sum_{l=1}^{N} \Pr[k^* = l] \Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 0 \wedge k^* = l\right], \\
&= \sum_{l=1}^{N} \Pr[k^* = l] \\
&\Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathcal{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}}) = 1 \mid b = 0 \wedge k^* = l\right], \\
&\overset{(1)}{=} \sum_{l=1}^{N} \frac{1}{N} \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l-1)}) = 1\right], \\
&\overset{(2)}{=} \sum_{l=0}^{N-1} \frac{1}{N} \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)}) = 1\right].
\end{aligned} \tag{71}$$

Here equality (1) comes from the fact that when $b = 0$, $(P_{l,m_l}, Q_{l,m_l}, I_l)$ is not a DDH triple. Equality (2) is obtained by simple changes in the indices of the summation. Similarly we obtain the following equation,
$$\begin{aligned}
&\Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 1\right] \\
&= \sum_{l=1}^{N} \Pr[k^* = l] \Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 1 \wedge k^* = l\right], \\
&= \sum_{l=1}^{N} \Pr[k^* = l] \\
&\Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathcal{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}}) = 1 \mid b = 1 \wedge k^* = l\right], \\
&= \sum_{l=1}^{N} \frac{1}{N} \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)}) = 1\right].
\end{aligned} \tag{72}$$

We have,
$$\begin{aligned}
&\Big| \Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 0\right] - \\
&\quad \Pr\left[\mathcal{D}_{\mathsf{DDH}}(X,Y,Z) = 1 \mid b = 1\right] \Big| \\
&\overset{(3)}{=} \left| \frac{1}{N} \left(\sum_{l=0}^{N-1} \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)}) = 1\right] - \sum_{l=1}^{N} \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)}) = 1\right] \right) \right|, \\
&\overset{(4)}{=} \frac{1}{N} \left| \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(0)}) = 1\right] - \Pr\left[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(N)}) = 1\right] \right|, \\
&\overset{(5)}{=} \frac{1}{N} \Big| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}'_c) = 1] \\
&\qquad - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\text{orig}}, \mathbf{H}_p(\mathcal{P}_{\text{orig}}), \mathbf{I}_c) = 1] \Big|, \\
&\overset{(6)}{\geq} \frac{1}{N} p(\lambda), \\
&= p_1(\lambda) \text{ (say)}.
\end{aligned} \tag{73}$$

Here the equality (3) comes from equations (71) and (72), the equality (4) comes from cancellations, the equality (5) comes from equations (70) and (69), and the inequality (6) comes from the assumption given in the inequality (68). However, the inequality (73) is a contradiction under the DDH assumption. So the inequality (68) cannot be true for any polynomial $p(\lambda)$. Hence there exists a negligible function $\mathsf{negl}'(\lambda)$ such that the inequality (64) holds. ■

Appendix C
Proof of Theorem 4

Proof: We prove the theorem by contradiction. Suppose, there exists a PPT adversary $\mathcal{B}$ in the `MPPLink` experiment for which the following inequality holds,
$$\left| \Pr[\mathcal{B}(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{act}}) = \mathfrak{b}] - \frac{1}{2} \right| \geq \frac{1}{p(\lambda)}, \tag{74}$$
where $p(\lambda)$ is a polynomial of the security parameter λ. Let $\mathsf{MPP}_{\mathsf{sim}}$ denote $f(\lambda)$ simulated proofs which is generated by the method given in the proof of Theorem 3 given in Appendix B. Here we are using the singleton set $\{P\}$ as the $f(\lambda)$ anonymity sets. From Claim 1 in the proof, no PPT adversary can distinguish between $\mathsf{MPP}_{\mathsf{act}}$ and $\mathsf{MPP}_{\mathsf{sim}}$ with a probability non-negligibly better than $\frac{1}{2}$. So for the adversary $\mathcal{B}$, there exists another PPT adversary $\mathfrak{B}$ such that the following inequality holds,
$$\left| \Pr[\mathfrak{B}(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{sim}}) = \mathfrak{b}] - \frac{1}{2} \right| \geq \frac{1}{p(\lambda)}, \tag{75}$$
given that inequality (74) is true for $\mathcal{B}$. Next, we construct a PPT adversary $\mathcal{A}$ for the `MoneroLink` experiment using $\mathfrak{B}$ as a subroutine. The construction of $\mathcal{A}(X_0, Y_0, X_1, Y_1, R, P)$ is given below.

1) $\mathcal{A}$ generates $f(\lambda)$ simulated MProve+ proofs $\mathsf{MPP}_{\mathsf{sim}}$ using the singleton set $\{P\}$ as the anonymity set in all of them, following the same steps of the simulator $\mathcal{S}_{\mathsf{MPP}}$ given in the proof of Theorem 3 given in Appendix B.
2) $\mathcal{A}$ sends $(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{sim}})$ to $\mathfrak{B}$ and receives $\hat{\mathfrak{b}}$.
3) It outputs $\hat{\mathfrak{b}}$ as the estimation of b.

Now we have,

$$\begin{aligned}&\left|\ \Pr[\mathcal{A}(X_0, Y_0, X_1, Y_1, R, P) = b] - \frac{1}{2}\ \right| \\ &= \left|\ \Pr[\mathfrak{B}(X_0, Y_0, X_1, Y_1, R, P, \mathsf{MPP}_{\mathsf{sim}}) = \mathfrak{b}] - \frac{1}{2}\ \right| \\ &\geq \frac{1}{p(\lambda)}. \end{aligned} \tag{76}$$

This contradicts Lemma 1. Hence there cannot be a PPT adversary $\mathcal{B}$ for which the inequality (74) holds. ∎

References

[1] IDEX blog. A complete list of cryptocurrency exchange hacks [updated]. [Accessed 27-MAY-2020]. [Online]. Available: https://blog.idex.io/all-posts/a-complete-list-of-cryptocurrency-exchange-hacks-updated

[2] Wikipedia contributors. Mt. Gox — Wikipedia, the free encyclopedia. [Accessed 27-MAY-2020]. [Online]. Available: https://en.bitcoin.it/wiki/Mt._Gox

[3] Proof-of-Reserves tool for Bitcoin. [Online]. Available: https://github.com/ElementsProject/reserves

[4] C. Decker, J. Guthrie, J. Seidel, and R. Wattenhofer, "Making bitcoin exchanges transparent," in *20th European Symposium on Research in Computer Security (ESORICS)*, 2015, pp. 561–576.

[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] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 31–48. [Online]. Available: https://doi.org/10.1145/3319535.3345655

[7] A. Dutta and S. Vijayakumaran, "MProve: A proof of reserves protocol for Monero exchanges," in *2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, June 2019, pp. 330–339.

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

[9] A. Dutta and S. Vijayakumaran, "Revelio: A MimbleWimble proof of reserves protocol," in *2019 Crypto Valley Conference on Blockchain Technology (CVCBT)*, June 2019, pp. 7–11.

[10] T. E. Jedusor, "Mimblewimble," 2016. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt

[11] A. Poelstra, "Mimblewimble," 2016. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf

[12] A. Dutta, A. Jana, and S. Vijayakumaran, "Nummatus: A privacy preserving proof of reserves protocol for Quisquis." in *In: Hao F., Ruj S., Sen Gupta S. (eds) Progress in Cryptology – INDOCRYPT 2019. INDOCRYPT 2019.Lecture Notes in Computer Science, vol 11898. Springer, Cham*, 2019, pp. 195–215.

[13] P. Fauzi, S. Meiklejohn, R. Mercer, and C. Orlandi, "Quisquis: A new design for anonymous cryptocurrencies," Cryptology ePrint Archive, Report 2018/990, 2018, https://eprint.iacr.org/2018/990.

[14] K. Chalkias, K. Lewi, P. Mohassel, and V. Nikolaenko, "Distributed auditing proofs of liabilities," Cryptology ePrint Archive, Report 2020/468, 2020, https://eprint.iacr.org/2020/468.

[15] "CoinMarketCap Markets." [Online]. Available: https://coinmarketcap.com

[16] 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.

[17] 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.

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell, "Bulletproofs: Short proofs for confidential transactions and more," in *2018 IEEE Symposium on Security and Privacy (SP)*, May 2018, pp. 315–334.

[19] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling up private payments without trusted setup - formal foundations and constructions of ring confidential transactions with log-size proofs," Cryptology ePrint Archive, Report 2019/580, 2019, https://eprint.iacr.org/2019/580.

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

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

[22] 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.

[23] Koe, K. M. Alonso, and S. Noether, "Zero to Monero: Second Edition," The Monero Project Library, April 2020. [Online]. Available: https://web.getmonero.org/library/Zero-to-Monero-2-0-0.pdf

[24] S. Noether. (2018) Reserve proof pull request. [Online]. Available: https://github.com/monero-project/monero/pull/3027

[25] 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.

[26] J. Yu, M. H. A. Au, and P. Esteves-Verissimo, "Re-thinking untraceability in the Cryptonote-style blockchain," Cryptology ePrint Archive, Report 2019/186, 2019, https://eprint.iacr.org/2019/186.

[27] Z. Yu, M. H. Au, J. Yu, R. Yang, Q. Xu, and W. F. Lau, "New empirical traceability analysis of Cryptonote-style blockchains," in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds. Cham: Springer International Publishing, 2019, pp. 133–149.

[28] R. Diestel, "Graph Theory," Springer, 3rd edition, 2005.

[29] A. Hinteregger and B. Haslhofer, "An empirical analysis of Monero cross-chain traceability," *CoRR*, vol. abs/1812.02808, 2018. [Online]. Available: http://arxiv.org/abs/1812.02808

[30] S. Bagad and S. Vijayakumaran, "On the confidentiality of amounts in Grin," in *2020 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2020, pp. 78–82.

[31] Ristretto Encoding of Monero public key. [Online]. Available: https://github.com/suyash67/curve25519-dalek/blob/ddbffc9/src/edwards.rs#L1167

[32] MProve+ simulation code. [Online]. Available: https://github.com/suyash67/MProvePlus-Ristretto

[33] MProve simulation code. [Online]. Available: https://github.com/suyash67/MProve-Ristretto

[34] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," Ph.D. dissertation, ETH Zürich, 1998.

[35] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," Tech. Rep., 1997.

[36] The size of the spent key images set in monero. [Online]. Available: https://monero.stackexchange.com/questions/12312/what-is-the-size-of-the-set-of-spent-key-images-in-monero

[37] Monero block explorer. [Online]. Available: https://moneroblocks.info/stats/transaction-stats