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 a part of an MProve proof, key images of exchange-owned one-time addresses are revealed. These key images play a dual role of detecting collusion between different exchanges and avoiding double spending. When an exchange tries to spend funds from one of its addresses after publishing an MProve proof, the key image for that address appears again as a part of the Monero transaction. As the key images and the addresses are inherently linked in the MProve proof, an observer could easily recognize the exchange-owned address. 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 next stage, a protocol called *proof of liabilities* generates a Pedersen commitment C_{liab} to the total amount of bitcoins that the exchange has sold to its customers (say a_{liab}). It also gives range proofs to prove that each customer's amount lies in the set $\{0, 1, 2, \dots, 2^{51}\}$ where 2^{51} is a bound on the maximum number of bitcoins that could exist. Any customer can verify that her amount is included in C_{liab}. In the last stage, a range proof is used to prove that $C_{\text{res}}C_{\text{liab}}^{-1}$ commits[*] to a non-negative number.

Proof of liabilities is a general protocol which can be applied to any cryptocurrency technology. Maxwell proposed a protocol (summarized in [7]) based on Merkle trees which helps an exchange to prove that a_{liab} includes the corresponding amount of a verifying customer. However this protocol

All the authors are with the Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India. Email: {arijit.dutta,suyashbagad}@iitb.ac.in, sarva@ee.iitb.ac.in

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

reveals the deposit amount of the sibling customer in the tree and a_{liab} to a verifying customer. The proof of liabilities in Provisions improved on it by publishing a commitment to the balance of each customer and a Pedersen commitment to a_{liab}. However, if a customer fails to check if her amount is accounted in a_{liab}, the exchange could possibly omit that amount, effectively reducing its liabilities. Recently, Chalkias *et al.* [8] proposed a scheme called Distributed Auditing Proofs of Liabilities (DAPOL) which addresses this concern. DAPOL uses a deterministic sparse Merkle tree along with range proofs on the commitment to each customer's amount and a Pedersen commitment to a_{liab}. 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. Further, they used key distribution function and verifiable random function to deterministically shuffle the users in the Merkle tree in every audit.

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 [9] for Monero [10], Revelio [11] for Mimblewimble [12] [13], and Nummatus [14] for Quisquis [15]. 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] or DAPOL [8] to prove that the exchange is solvent. It 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 [16]. If exchanges publish proofs of reserves periodically, loss of assets by an exchange can be detected early.

Our contribution. MProve [9] 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*[†] 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 [17], [18]. 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 [19] and Omniring [6].
(ii) 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.

[†]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.

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 [19] and Omniring [6] 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. Section V gives the performance comparison of the MProve+ and MProve protocol. We draw conclusions in Section VI.

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 discrete logarithm 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. Pedersen commitments have homomorphic property i.e. amounts a_1, a_2 can be added by multiplying their corresponding Pedersen commitments $C_1 = G^{y_1} H^{a_1}$ and $C_2 = G^{y_2} H^{a_2}$. This property is used to obtain a Pedersen commitment to the total reserves of an exchange.

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 $u \in \mathbb{Z}_q^*$, 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 [10], 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 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 $P' = G^{H(B^r_{\text{vk}})} \cdot B_{\text{sk}}$, where $H : \mathbb{G} \mapsto \mathbb{Z}_q$ is a hash function which maps group elements to scalars[‡]. 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 $P'' = G^{H(R^{b_{\text{vk}}})} \cdot B_{\text{sk}}$. 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,

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

The above equality holds because $B^r_{\text{vk}} = G^{b_{\text{vk}} r} = R'^{b_{\text{vk}}}$. 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 $H(R'^{b_{\text{vk}}}) + b_{\text{sk}}$ because,

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

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.

[‡]We ignore the concatenation of output index while generating one-time address as given in [9] for the ease of representation.

Finally using confidential transactions, the amount ($a \in \{0, 1, 2, \ldots, N\}, N = 2^{64} - 1$) 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*,

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

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

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 $B^r_{\text{vk}} = G^{b_{\text{vk}} r} = R'^{b_{\text{vk}}}$. 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 [9], [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 [9] 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 in Appendix A.

Drawback of MProve[§]*:* 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.

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

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 in the ring, P_j is owned by *Ex*.
3) In *txn*, P_j is the source of the transaction.

All three statements affect the privacy of the exchange. However, statement 2 and statement 3 are more crucial because of the following reasons.

i. It is revealed exactly which exchange-owned one-time address (P_j) is being spent in *txn*.
ii. Source obfuscation in *txn* by linkable ring signature is no longer functional.
iii. Now P_j has to be pruned from the Monero UTXO set. Ring size reduces by 1 for all transactions which have used P_j as cover in the ring of linkable ring signatures so far.

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 [19] and Omniring [6] 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 [6] proposes a technique where the proof of validity of the transaction is logarithmic in the size of the ring. Omniring is motivated from Bulletproofs [19] 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 [19] gives a state-of-the-art range proof system with logarithmic proof size. Here, given a Pedersen commitment[¶] $C = G^v H^\gamma$, 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 $\mathbf{a}_R$.

$$\langle \mathbf{a}_L, \mathbf{2}^n \rangle = v \tag{5}$$

$$\langle \mathbf{a}_L, \mathbf{a}_R \circ \mathbf{y}^n \rangle = 0 \tag{6}$$

$$\langle \mathbf{a}_L - \mathbf{1}^n - \mathbf{a}_R, \mathbf{y}^n \rangle = 0, \tag{7}$$

where $y \xleftarrow{\$} \mathbb{Z}_q$ is a random challenge sent by the verifier. Here equation (5) ensures that $\mathbf{a}_L$ is the binary representation of v, equation (6) ensures that the component-wise product of $\mathbf{a}_L$ with $\mathbf{a}_R$ is always a zero vector, and equation (7) ensures that $\mathbf{a}_R$ is obtained by subtracting the elements of $\mathbf{a}_L$ from $\mathbf{1}^n$ vector. Both equation (6) and (7) 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, 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.

$$\langle \mathbf{a}_L - z\cdot\mathbf{1}^n, \mathbf{y}^n \circ (\mathbf{a}_R + z\cdot\mathbf{1}^n) + z^2\cdot\mathbf{2}^n \rangle = z^2\cdot v + \delta(y, z), \tag{8}$$

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

As x is chosen randomly, this is equivalent to checking equation (8). 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.

[¶]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 [19], it is the opposite. However, this is just a difference in notation.

2) Omniring: For a single source transaction in Omniring [6], 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

$$1_g = G^{-x}\mathbf{P}^{\mathbf{e}}, \tag{9}$$

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, 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 (9) can be alternatively represented as

$$(G\|\mathbf{P})^{\mathbf{a}} = 1_g. \tag{10}$$

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 (10) we observe that the base vectors to generate A must include $\mathbf{P}$ to show that the main equality (9) 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 (9) 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 (9) is implicitly verified. MProve+ follows this technique.

III. MProve+ : An Improvement over MProve

Below we discuss how we use the techniques of Bulletproofs and Omniring in the MProve+ protocol. The difference between the MProve+ protocol and the Omniring scheme is discussed in Appendix B.

A. Main Idea

In MProve+, a Monero exchange Ex publishes a list of one-time addresses $\mathscr{P}_{\text{anon}} = \{P_1, P_2, \ldots, P_n\}$ as the anonymity set. Let $\{C_1, C_2, \ldots, C_n\}$ be the list of corresponding Pedersen commitments which can be read from the Monero blockchain. We define four vectors as $\mathbf{P} = (P_1, P_2, \ldots, P_n)$, $\mathbf{C} = (C_1, C_2, \ldots, C_n)$, $\mathbf{H}_p = \big(H_p(P_1), H_p(P_2), \ldots, H_p(P_n)\big)$, and $\mathbf{I} = (I_1, I_2, \ldots, I_s)$, where s is the number of one-time addresses owned by Ex in $\mathscr{P}_{\text{anon}}$ and H_p is the hash function used to generate key images in Monero (see Section II.B). MProve+ reveals the number of addresses owned by the exchange by publishing vector $\mathbf{I}$. For the jth owned address ($j \in [s]$), there exists an index $i_j \in [n]$ for which Ex knows $(x_{i_j}, y_{i_j}, a_{i_j})$ such that the following relation holds.

$$P_{i_j} = G^{x_{i_j}} \wedge I_j = H_p(P_{i_j})^{x_{i_j}} \wedge C_{i_j} = G^{y_{i_j}} H^{a_{i_j}}. \tag{11}$$

As mentioned in Section II.B, using x_{i_j} anyone can derive y_{i_j} and a_{i_j}. So proving knowledge of x_{i_j} will be enough. For each source address, Ex uses a secret unit vector $\mathbf{e}_j$ of length n. For the jth source address, $\mathbf{e}_j$ has zeros in every position except where jth source address is located in $\mathbf{P}$. In that index, $\mathbf{e}_j$ has 1. Ex also publishes a Pedersen commitment to the total reserves which is calculated as follows.

$$C_{\text{res}} = (G_1)^{\gamma} \prod_{j=1}^{s} C_{i_j}, \tag{12}$$

where G_1 is a random curve point with unknown discrete logarithm relationship with respect to point G and H. It can be generated by hashing G and H. Pedersen commitments C_{i_j}s are owned by Ex and $\gamma \xleftarrow{\$} \mathbb{Z}_q$ is a blinding factor. The generated commitment C_{res} is a doubly blinded Pedersen commitment. To do range proof on C_{res} for proof of solvency, we can follow the protocol described by Jivanyan [Appendix B, [25]]. Ex proves that it knows witnesses satisfying the following language.

$$\mathcal{L}_{\mathsf{MP+}}^{\mathsf{crs}} = \left\{ \left(\begin{array}{c} \mathbf{P},\ \mathbf{C},\ G_1 \\ \{I_j\}_{j=1}^{s},\ C_{\text{res}} \end{array} \right) \middle| \begin{array}{l} \exists(\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \gamma) \\ \text{such that } \mathbf{P}^{\mathbf{e}_j} = G^{x_j}, \\ \mathbf{H}_p^{\mathbf{e}_j} = I_j^{x_j^{-1}}\ \forall j \in [s], \\ G_1^{\gamma} \prod_{j=1}^{s} \mathbf{C}^{\mathbf{e}_j} = C_{\text{res}}. \end{array} \right\} \tag{13}$$

Here crs specifies the necessary details like description of the group, its generators, the hash function $H_p(\cdot)$ to be used. We define $(\mathbf{P}, \mathbf{C}, \{I_j\}_{j=1}^{s}, C_{\text{res}}, G_1)$ as the statement stmt of the language. We also define $(\boldsymbol{x} = (x_1, x_2, \ldots, x_s), \mathbf{e}_1, \ldots, \mathbf{e}_s, \gamma)$ as the witness wit of the language.

B. Forming the Main Equality

We observe that the last condition in the language in (13) can be alternatively expressed by the following equation.

$$G_1^{\gamma} \mathbf{C}^{\mathbf{1}^s \mathcal{E}} = C_{\text{res}}, \tag{14}$$

where $\mathcal{E}$ is the $s \times n$ matrix containing $\mathbf{e}_1, \ldots, \mathbf{e}_s$ as the rows. We define the following quantities where u, v are public coin challenges sent by the verifier. They can be generated by *Ex* by using Fiat-Shamir heuristic.

$$\hat{\mathbf{Y}} := \mathbf{P}^{\circ u} \circ \mathbf{H}_p^{\circ u^2} \tag{15}$$

$$\hat{\mathbf{I}} := \mathbf{I}^{\circ - u^2 \boldsymbol{v}^s} \tag{16}$$

$$\hat{\mathbf{e}} := \boldsymbol{v}^s \mathcal{E} \tag{17}$$

$$\mathbf{e}' := \mathbf{1}^s \mathcal{E}. \tag{18}$$

The three conditions given in the language in (13) can be expressed by the following equations.

$$C_{\text{res}}^{-1} G_1^{\gamma} \mathbf{C}^{\mathbf{e}'} = 1_g \tag{19}$$

$$G^{-x_j} \mathbf{P}^{\mathbf{e}_j} = 1_g, \forall j \in [s] \tag{20}$$

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

where 1_g is the identity element of $\mathbb{G}$. Equations (20) and (21) represent s equations each. In both cases, we exponentiate the jth equation with v^{j-1}, $j \in [s]$. Then we combine all the equations by exponentiating all s equations obtained from (20) by u, the last s equations obtained from (21) by u^2, and multiplying all of them with equation (19). These operations give us the following equation.

$$1_g = G^{\xi} C_{\text{res}}^{-1} G_1^{\gamma} \hat{\mathbf{Y}}^{\hat{\mathbf{e}}} \mathbf{C}^{\mathbf{e}'} \hat{\mathbf{I}}^{\boldsymbol{x}^{\circ -1}}, \tag{22}$$

where $\xi = -\langle \boldsymbol{x}, u\boldsymbol{v}^s \rangle$. We call equation (22), the main equality for MProve+.

C. Defining Secret Vectors and the Constraint Equations

Now we define the secret vectors of length $N = sn+2n+s+3$ as 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. Vector $\mathbf{c}_L$ is the vector containing all the exponents of the main equality (22) and $\mathsf{vec}(\mathcal{E})$. Vector $\mathbf{c}_R$ is an auxiliary vector used to prove the constraints on the witnesses.

Figure 2 gives some constraint vectors which are used to express the constraints of various parts of the secret vector in terms of inner products. All these inner product constraint equations are given in Figure 4. Here equations (23), (28), and (29) verify that $\mathbf{e}_j$ are indeed unit vectors and the second position of $\mathbf{c}_L$ contains -1. Equation (24) verifies that the elements in the vectors $\mathbf{c}_L$ and $\mathbf{c}_R$ located from the $(2n+4)$th to $(2n+3+s)$th positions are inverses of each other. Equations (25), (26), and (27) verify the definitions of ξ, $\hat{\mathbf{e}}$, and $\mathbf{e}'$ respectively.

$$\begin{aligned} \mathbf{c}_L &:= (\ \xi \ \|\ -1 \ \|\ \gamma \ \|\ \hat{\mathbf{e}} \ \|\ \mathbf{e}' \ \|\ \boldsymbol{x}^{\circ -1} \ \|\ \ vec(\mathcal{E}) \ \ ) \\ \mathbf{c}_R &:= (\ \qquad \mathbf{0}^{2n+3} \qquad \ \|\ \ \boldsymbol{x} \ \ \|\ \mathbf{1}^{sn} - vec(\mathcal{E})\ ) \end{aligned}$$

Fig. 1: Honest encoding of witness.

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

Fig. 2: Definitions of constraint vectors (dots mean zero scalars and vectors).

$$\boldsymbol{\theta} := \mathbf{v}_0 + z \cdot \mathbf{v}_1, \quad \boldsymbol{\zeta} := \sum_{i=2}^{6} z^i \cdot \mathbf{v}_i, \quad \boldsymbol{\nu} = z^2 \cdot \mathbf{v}_7 + z^6 \cdot \mathbf{v}_6,$$

$$\kappa = z \cdot \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle + z^5 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle + z^6 \cdot \langle \mathbf{1}^{sn}, \mathbf{v}_6 \rangle,$$

$$\boldsymbol{\pi} = \boldsymbol{\nu} \circ \boldsymbol{\theta}^{\circ -1}, \quad \boldsymbol{\beta} = \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\zeta}, \quad \delta = \kappa + \langle \boldsymbol{\pi}, \boldsymbol{\zeta} \rangle.$$

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

$$\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0 \iff$$

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_0 \rangle = 0 \tag{23}$$

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_1 \rangle = \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle \tag{24}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_2 \rangle + \langle \boldsymbol{\gamma}_R, \mathbf{v}_7 \rangle = 0 \tag{25}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_3 \rangle = 0 \tag{26}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_4 \rangle = 0 \tag{27}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_5 \rangle = \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle \tag{28}$$

$$\langle \boldsymbol{\gamma}_L + \boldsymbol{\gamma}_R - \mathbf{1}^{sn}, \mathbf{v}_6 \rangle = 0 \tag{29}$$

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

D. 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 (29) (see Figure 4) by consecutive powers of z namely $1, z, z^2, \ldots, z^6$ and adding them gives,

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\zeta} \rangle + \langle \boldsymbol{\nu}, \boldsymbol{\gamma}_R \rangle = \kappa, \tag{30}$$

where $\boldsymbol{\theta}, \boldsymbol{\zeta}, \boldsymbol{\nu}$, and κ are defined in Figure 3. To get a single inner product, we modify (30) as follows,

$$\begin{aligned} \langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\zeta} \rangle + \langle \boldsymbol{\nu} \circ \boldsymbol{\theta}^{\circ -1}, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\zeta} \rangle &= \kappa + \langle \boldsymbol{\nu} \circ \boldsymbol{\theta}^{\circ -1}, \boldsymbol{\zeta} \rangle \\ \implies \langle \boldsymbol{\gamma}_L + \boldsymbol{\pi}, \boldsymbol{\gamma}_R \circ \boldsymbol{\theta} + \boldsymbol{\zeta} \rangle &= \delta, \end{aligned} \tag{31}$$

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

In the following protocol, we prove the inner product given in equation (31) 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.

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

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

crs $= (\mathbb{G}, q, G, H, H_p(\cdot))$.

Generate:

$G_1 = H_p(G \| H)$, $\gamma \xleftarrow{\$} \mathbb{Z}_q$, $C_{\text{res}} = G_1^{\gamma} \prod_{j=1}^{s} C_{i_j}$, $\mathbf{P} = \{P_i\}_{i=1}^{n}$, $\mathbf{H}_p = \{H_p(P_i)\}_{i=1}^{n}$, $\mathbf{I} = \{I_j\}_{j=1}^{s} = \{H_p(P_{i_j})^{x_{i_j}}\}_{j=1}^{s}$, $\mathbf{C} = \{C_i\}_{i=1}^{n}$, wit $= (\boldsymbol{x}, \mathbf{e}_1, \ldots, \mathbf{e}_s, \gamma)$

Output: stmt $= (\mathbf{P}, \mathbf{C}, \mathbf{I}, C_{\text{res}}, G_1)$

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

$\mathcal{V}$: $u, v \xleftarrow{\$} \mathbb{Z}$, $H' \xleftarrow{\$} \mathbb{G}$, $\mathbf{Q} \xleftarrow{\$} \mathbb{G}^{2n+s+3}$, $\mathbf{G}' \xleftarrow{\$} \mathbb{G}^{sn}$, $\mathbf{H} \xleftarrow{\$} \mathbb{G}^{N}$, where $N = sn + 2n + s + 3$.

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

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

(i) Compute $\hat{\mathbf{Y}} = \mathbf{P}^{u} \circ \mathbf{H}_p^{\circ u^2}$ and $\hat{\mathbf{I}} = \mathbf{I}^{\circ -u^2 v^s}$

(ii) For a variable $k \in \mathbb{Z}_q$, define the vector

$$\mathbf{G}_k := \left[\left((G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{\circ k} \circ \mathbf{Q} \right) \| \mathbf{G}' \right] \tag{32}$$

$\mathcal{P}$:

(i) $r_A \xleftarrow{\$} \mathbb{Z}_q$

(ii) $A := (H')^{r_A} \mathbf{G}_0^{\mathbf{c}_L} \mathbf{H}^{\mathbf{c}_R}$

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

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

$\mathcal{P}$:

(i) $r_S \xleftarrow{\$} \mathbb{Z}_q, \mathbf{s}_L \xleftarrow{\$} \mathbb{Z}_q^N$, for $\mathbf{s}_R \in \mathbb{Z}_q^N$ s.t. for $j \in [N]$

$$\mathbf{s}_R[j] = \begin{cases} s_j \xleftarrow{\$} \mathbb{Z}_q, & \text{for } \mathbf{c}_R[j] \neq 0 \\ 0, & \text{for } \mathbf{c}_R[j] = 0 \end{cases}$$

(ii) $S = (H')^{r_S} \mathbf{G}_w^{\mathbf{s}_L} \mathbf{H}^{\mathbf{s}_R}$

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

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

$\mathcal{P}$:

(i) Define the following polynomials as a function of X in $\mathbb{Z}_q^N[X]$:

$$\begin{aligned} l(X) &:= \mathbf{c}_L + \boldsymbol{\pi} + \mathbf{s}_L \cdot X \\ r(X) &:= \boldsymbol{\theta} \circ (\mathbf{c}_R + \mathbf{s}_R \cdot X) + \boldsymbol{\zeta} \\ t(X) &:= \langle l(X), r(X) \rangle = t_2 X^2 + t_1 X + t_0 \end{aligned}$$

for some $t_2, t_1, t_0 \in \mathbb{Z}_q$. Also, $t_0 = \delta$.

(ii) Compute $T_1 = G^{t_1} H^{\tau_1}, T_2 = G^{t_2} H^{\tau_2}$ where $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$.

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

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

$\mathcal{P}$:

(i) $\boldsymbol{\ell} := l(x) = \mathbf{c}_L + \boldsymbol{\pi} + \mathbf{s}_L \cdot x \in \mathbb{Z}_q^N$

(ii) $\boldsymbol{\mathfrak{r}} := r(x) = \boldsymbol{\theta} \circ (\mathbf{c}_R + \mathbf{s}_R \cdot x) + \boldsymbol{\zeta} \in \mathbb{Z}_q^N$

(iii) $\hat{t} := \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle \in \mathbb{Z}_q$

(iv) $\tau_x := \tau_2 x^2 + \tau_1 x$

(v) $r := r_A + r_S x$

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

$\mathcal{V}$:

(i) $\hat{t} \overset{?}{=} \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle$ *// $\hat{t}$ was computed correctly*

(ii) $G^{\hat{t}} H^{\tau_x} \overset{?}{=} G^{\delta} T_1^{x} T_2^{x^2}$ *// $\hat{t}$ equals to $t_0 + t_1 x + t_2 x^2$*

(iii) $(H')^{r} \mathbf{G}_w^{\boldsymbol{\ell}} \mathbf{H}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}} \overset{?}{=} A S^{x} \mathbf{G}_w^{\boldsymbol{\pi}} \mathbf{H}^{\boldsymbol{\beta}}$ *//verify $\boldsymbol{\ell} = l(x)$ and //$\boldsymbol{\mathfrak{r}} = r(x)$*

Verification equations (i) and (iii) need $\boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \in \mathbb{Z}_q^N$ which requires $\mathcal{O}(N)$ size communication from the prover. Instead, we can use the inner product protocol which is used in Bulletproofs [19] and Omniring [6]. The inner product argument is expressed by the following language.

$$\mathcal{L}_{IP} = \left\{ P \in \mathbb{G}, c \in \mathbb{Z}_q \;\middle|\; \begin{array}{c} \exists (\mathbf{a}, \mathbf{b}) \text{ such that} \\ P = U^{c} \mathbf{G}^{\mathbf{a}} \mathbf{H}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle \end{array} \right\} \tag{33}$$

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 $\boldsymbol{\ell}$ and $\boldsymbol{\mathfrak{r}}$ without knowing $\boldsymbol{\ell}$ and $\boldsymbol{\mathfrak{r}}$ as

$$P = U^{\hat{t}} \mathbf{G}_w^{\boldsymbol{\ell}} (\mathbf{H}')^{\boldsymbol{\mathfrak{r}}} = U^{\hat{t}} (H')^{-r} A S^{x} \mathbf{G}_w^{\boldsymbol{\pi}} \mathbf{H}^{\boldsymbol{\beta}} \tag{34}$$

by verification equation (iii), where $\mathbf{H}' = \mathbf{H}^{\boldsymbol{\theta}^{\circ -1}}$. With this, the prover and the verifier engage in the inner product argument to prove verification equations (i) and (iii). So the prover does not send $\boldsymbol{\ell}$ and $\boldsymbol{\mathfrak{r}}$ in the previous step reducing the communication cost to $\mathcal{O}(\log_2(N))$. 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 whose proofs are given in Appendix C.

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

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

E. Proof Generation and Verification

The exchange follows the $\Pi_{\text{MProve+}}$ protocol and publishes $(\mathbf{P}, \mathbf{I}, C_{\text{res}}, G_1)$ and a $\Pi_{\text{MProve+}}$ proof. The verifier of $\Pi_{\text{MProve+}}$ protocol does the following verification steps.

1) Computes $\mathbf{H}_p$ using the hash function H_p. Reads $\mathbf{C}$ by looking at the Monero blockchain and using $\mathbf{P}$. Checks $G_1 \overset{?}{=} H_p(G \| H)$.
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 $\Pi_{\text{MProve+}}$ 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 in the implementation which is discussed in Appendix D.

IV. Security Properties

The MProve+ protocol has the following security properties.

A. Reserves Confidentiality and Inflation Resistance

While showing the well-formedness of the extracted witnesses in the proof of Theorem 2 (see Section C.B), we have shown that the following equations hold.

$$\mathbf{P}^{\mathbf{e}'_l} = G^{x'_l}, \quad l \in [s], \tag{35}$$

$$\mathbf{H}_p^{\mathbf{e}'_l} = I_l^{(x'_l)^{-1}}, \quad l \in [s], \tag{36}$$

$$C_{\text{res}} = G_1^{\gamma'} \prod_{l\in[s]} \mathbf{C}^{\mathbf{e}'_l}, \tag{37}$$

where $\mathbf{e}'_l$ are proved to be unit vectors. As all $I_l \in \mathbf{I}$, $l \in [s]$ are distinct, all the unit vectors $\mathbf{e}_l$, $l \in [s]$ are also distinct by equation (36). So equations (35) and (37) imply that only s distinct Pedersen commitments ($C_l = G^{y_l} H^{a_l} \in \mathbf{C}, l \in [s]$) are chosen from $\mathbf{C}$ for which the corresponding secret keys (x'_ls) are known to the exchange. These Pedersen commitments are homomorphically multiplied along with a blinding factor γ' to give a doubly blinded Pedersen commitment to the total reserves i.e. C_{res}. As γ' is uniformly chosen from $\mathbb{Z}_q$, C_{res} is uniformly distributed $\mathbb{G}$, containing no information about the total reserves amount a_{res}. In this way the total reserves amount a_{res} remains confidential in the MProve+ protocol. Now let us consider the possibility of inflation.

Let $\prod_{l\in[s]} \mathbf{C}^{\mathbf{e}'_l} = G^{y_{\text{res}}} H^{a_{\text{res}}}$, where y_{res} is a random blinding factor obtained by adding all the blinding factors of the exchange-owned Pedersen commitments. If the exchange inflates the total reserves amount a_{res} with $a'_{\text{res}} \neq a_{\text{res}}$, then it has to come up with scalars y'_{res} and a'_{res} such that $\prod_{l\in[s]} \mathbf{C}^{\mathbf{e}'_l} = G^{y'_{\text{res}}} H^{a'_{\text{res}}}$. This implies that it can find the discrete logarithm of H with respect to G as $H = G^{(y_{\text{res}}-y'_{\text{res}})(a'_{\text{res}}-a_{\text{res}})^{-1}}$. So we have the following theorem.

Theorem 3: The MProve+ protocol preserves the confidentiality of the total reserves amount and is resistant to inflation under the discrete logarithm assumption.

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,

$$P = G^x \wedge I = (H_p(P))^x, \tag{38}$$

where x is a secret scalar. Notice that for a given P, a unique I could satisfy equation (38). Hence if two exchanges use a common one-time address as source to generate MProve+ proofs, a common key image corresponding to that one-time address will appear in both of their key image vectors. Thus a verifier can easily detect collusion between exchanges. Therefore, the MProve+ protocol is collusion resistant.

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 [9], 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 E, 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 E.

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

Because of the challenges discussed in Appendix E, 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 consequence when a probabilistic polynomial time (PPT) adversary 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[‖] from any one-time address in $\mathbf{P}^{(i)}$. 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. For a key image $I \in \{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$, we define $\mathcal{P}_{\text{orig}}(I)$ as the *originating set* for I. For a PPT adversary which is given some information (e.g. Monero blockchain, MProve+ proofs), the set $\mathcal{P}_{\text{orig}}(I)$ is the set of one-time addresses of minimal cardinality which could have originated I. Suppose I has appeared in j_1th, j_2th,..., j_rth proofs among the overall $f(\lambda)$ MProve+ proofs.

[‖]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.

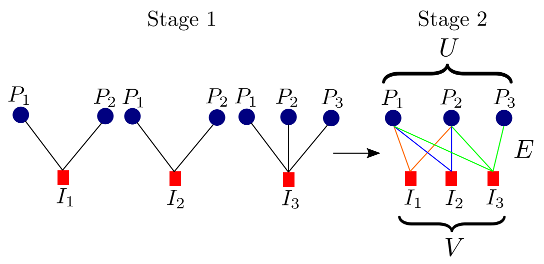

Fig. 5: Illustration of Example 1.

Then it is obvious that,

$$\mathscr{P}_{\text{orig}}(I) \subset \bigcap_{k=j_1}^{j_r} \mathbf{P}^{(k)}. \tag{39}$$

Notice that each MProve+ proof induces a complete bipartite graph[**]. For example, the ith MProve+ proof induces a complete bipartite graph 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. The bipartite graph is complete because each key image in $\mathbf{I}^{(i)}$ is likely to be originated from any one-time address in $\mathbf{P}^{(i)}$. We give the following example.

Example 1: Let $f(\lambda) = 3$. 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 $\mathscr{P}_{\text{orig}}(\cdot)$ as given in equation (39), then we get

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

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 $\{I_1, I_2\}$[††]. This makes P_3 as the only member of $\mathbf{P}^{(3)}$ which could possibly originate I_3. To get a precise definition of the originating set, we construct the simple[‡‡] 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 comes 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.

Theorem 4: The only information that a PPT adversary can obtain from the $f(\lambda)$ MProve+ proofs is the $f(\lambda)$ bipartite graphs $\left(\mathbf{P}^{(i)}, \mathbf{I}^{(i)}, \mathbf{P}^{(i)} \times \mathbf{I}^{(i)} \right)_{i=1}^{f(\lambda)}$.

The proof of Theorem 4 is given in Appendix F. Though Theorem 4 vindicates the MProve+ protocol to be privacy preserving to some extent, the elements of $\{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}$ can bring privacy concerns. 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) Whether some source addresses which were used before, are being used again. 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.

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

[††]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.

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

[§§]One set of disjoint nodes in each of the $f(\lambda)$ bipartite graphs.

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[¶¶]. 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.* [18] and Kumar *et al.* [17]. 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 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.

[¶¶]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.

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 [9], and MProve+ are some examples. Removing this drawback is an open problem because of the challenges discussed in Appendix E. 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 and the implications have been discussed[***]. To have a more clear view, let us 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 adversary $\mathcal{A}$ who observes this MProve proof, I_j could have originated either from P_j or from a group element $C'_jC_j^{-1}$. So in this case we have $\mathscr{P}_{\text{orig}}(I_j) = \{P_j, C'_jC_j^{-1}\}$. 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 $C'_jC_j^{-1}$ 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,

$$\Pr[\mathcal{A}(\mathscr{P}_{\text{orig}}(I_j), \mathbf{R}(txn)) = P_j] = 1. \tag{40}$$

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

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

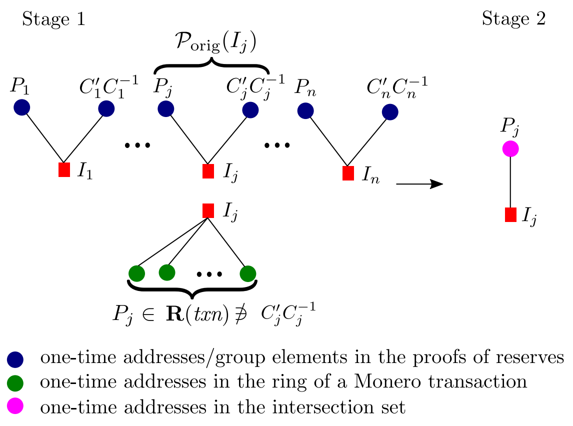
Stage 1
Stage 2
$\mathcal{P}_{\text{orig}}(I_j)$
P_1 $C'_1C_1^{-1}$ P_j $C'_jC_j^{-1}$ P_n $C'_nC_n^{-1}$
I_1 I_j I_n
P_j
I_j
$P_j \in \mathbf{R}(txn) \not\ni C'_jC_j^{-1}$
● one-time addresses/group elements in the proofs of reserves
● one-time addresses in the ring of a Monero transaction
● one-time addresses in the intersection set

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

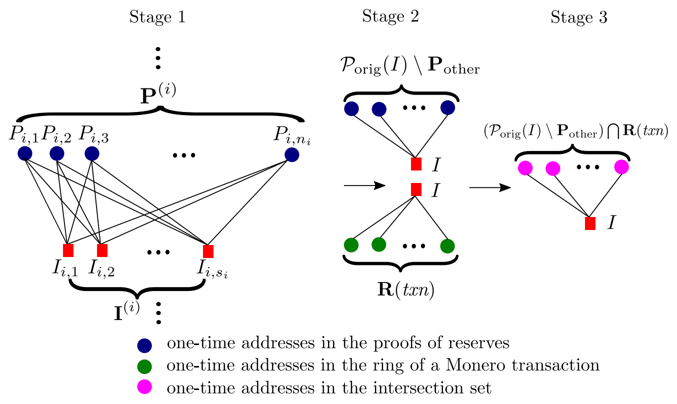
Stage 1
Stage 2
Stage 3
$\mathbf{P}^{(i)}$
$P_{i,1} P_{i,2} P_{i,3}$ P_{i,n_i}
$I_{i,1}$ $I_{i,2}$ I_{i,s_i}
$\mathbf{I}^{(i)}$
$\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}$
I
$\mathbf{R}(txn)$
$(\mathcal{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}) \bigcap \mathbf{R}(txn)$
● one-time addresses in the proofs of reserves
● one-time addresses in the ring of a Monero transaction
● one-time addresses in the intersection set

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

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 an active adaptive adversary as mentioned above, 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 when txn has not appeared in the Monero blockchain. $\mathcal{A}$ knows that any address in $\mathbf{P}_{\text{other}}$ cannot be the originating address for I. So the probability that $\mathcal{A}$ successfully outputs P as the originating address for I is given by,

$$\Pr[\mathcal{A}(\mathscr{P}_{\text{orig}}(I), \mathbf{P}_{\text{other}}) = P] = \frac{1}{\left|\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right|}. \tag{41}$$

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 $\left(\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$ 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.,

$$\text{InfoMPP} = (\mathscr{P}_{\text{orig}}(I), \mathbf{P}_{\text{other}}, \mathbf{R}(txn)). \tag{42}$$

The equation (41) is modified as follows to give the probability that $\mathcal{A}$ successfully links P with I.

$$\Pr[\mathcal{A}(\text{InfoMPP}) = P] = \frac{1}{\left|\left(\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)\right|}. \tag{43}$$

It is desirable for Ex that the probability given in equation (43) 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 (43) 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 $\mathscr{P}_{orig}(I)$ becomes as large as possible. Later when Ex spends from P in txn, Ex should choose $\mathbf{R}(txn)$ as a proper subset of $\mathscr{P}_{orig}(I)$.

Now observe equation (40) and (43). 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. However for MProve+, the exchange can choose the anonymity sets and the ring of the transaction in a proper way and make the linking probability considerably low than 1. Hence we conclude that MProve+ is better than MProve when the privacy of the entire Monero network including the exchange is of concern. The case when multiple source addresses from the MProve+ proofs are spent is discussed in Appendix G.

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]. Yu *et al.* [26] have introduced two kinds of untraceability for Monero i.e. *individual untraceability* and *global untraceability*. Let the size of the ring of a Monero transaction be l. Then the individual untraceability denotes the number of addresses ($\leq l-1$) an active adaptive adversary needs to de-anonymize (to make sure that the given address is not being spent) to find the real spent address. Whereas, the global untraceability denotes the number of addresses the adversary needs to de-anonymize in all transactions of the blockchain to determine a single spent address of any transaction. Motivated by this approach, we discuss about the untraceability in the context of proof of reserves protocol (for MProve+ in particular) for Monero.

First of all, note that we have already discussed about the individual untraceability for the MProve+ proofs. In particular, the individual untraceability for MProve+ proofs refers to the number of addresses an active adaptive adversary needs to de-anonymize to find the originating address corresponding to a particular key image published in the MProve+ proofs. The probability given in equation (43) gives the probability to obtain the originating address corresponding to a published key image when multiple MProve+ proofs are published and source addresses are spent in future transactions. From equation (43), the individual untraceability for a given key image I is given by the number $\left|\left(\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)\right| - 1$, wherein txn the address corresponding to I is spent.

Now let us consider the global untraceability for MProve+ proofs. We consider the situation when $f(\lambda)$ MProve+ proofs are published and some source addresses are spent by the exchange after publishing the MProve+ proofs. As mentioned above, when a particular key image I appears in a Monero transaction, the corresponding originating set shrinks to

$$\mathscr{P}'_{\text{orig}}(I) = \left(\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn). \tag{44}$$

Recall that $\mathscr{P}_{\text{orig}}$ denote the set of originating sets for all distinct published key image (equation (80)). Let the set $\mathscr{P}'_{\text{orig}}$ denote the set of originating sets where the originating sets are updated as in equation (44) considering all the source spending transactions. Let M denote the number of distinct one-time addresses in all sets in $\mathscr{P}'_{\text{orig}}$. Recall that N denotes the number of distinct published key images, hence the number of distinct source addresses in $f(\lambda)$ MProve+ proofs. Hence the probability that the adversary finds a source address is $\frac{N}{M}$. The adversary needs to de-anonymize atleast $M - N - 1$ addresses to find a source address. Hence the global untraceability for $f(\lambda)$ MProve+ proofs is denoted by the number $M - N - 1$.

MProve+ proofs and the unlinkability property of Monero. Another design goal for Monero is to achieve *unlinkability*. Roughly speaking, unlinkability means that two transactions paying the same user cannot be linked [21]. This property also implies that given a one-time address, it is impossible for a PPT adversary to obtain the corresponding public key. We want to show that no PPT adversary can link any one-time address with its corresponding public key pair using the published $f(\lambda)$ MProve+ proofs. Suppose there exists a PPT adversary $\mathscr{A}_1$ who can determine the public key pair of a one-time address P by observing $f(\lambda)$ MProve+ proofs containing P. So $\mathscr{A}_1$ is given $f(\lambda)$ MProve+ proofs as input and it outputs a triple (P, X, Y). Here P is a one-time address in the corresponding $f(\lambda)$ anonymity sets used in the MProve+ proofs and (X, Y) is the public key pair of P. We construct another PPT adversary $\mathscr{A}_2$ using $\mathscr{A}_1$ as a subroutine. The adversary $\mathscr{A}_2$ observes $f(\lambda)$ transactions in the Monero blockchain and outputs a triple (P, X, Y). Here P is a one-time address which is present in one or more rings of the $f(\lambda)$ transactions and (X, Y) is the public key pair of P. The construction of $\mathscr{A}_2$ is as follows.

1) $\mathscr{A}_2$ sets the $f(\lambda)$ rings as the anonymity sets $\{\mathbf{P}^{(i)}\}_{i=1}^{f(\lambda)}$ of the $f(\lambda)$ MProve+ proofs to be generated. $\mathscr{A}_2$ generates $\{\mathbf{H}_p^{(i)}\}_{i=1}^{f(\lambda)}$ accordingly using $\{\mathbf{P}^{(i)}\}_{i=1}^{f(\lambda)}$ and the hash function used in Monero.
2) Let n_i be the size of the ith ring. $\mathscr{A}_2$ randomly generates $s_i \xleftarrow{\$} [n_i]$ for all $i \in [f(\lambda)]$. For all $i \in [f(\lambda)]$, $\mathscr{A}_2$ samples s_i random group elements from $\mathbb{G}$ to construct $\mathbf{I}^{(i)}$.
3) $\mathscr{A}_2$ has already obtained $\left(\{\mathbf{P}^{(i)}\}_{i=1}^{f(\lambda)}, \{\mathbf{H}_p^{(i)}\}_{i=1}^{f(\lambda)}, \{\mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}\right)$. To generate other parts of the $f(\lambda)$ MProve+ proofs, $\mathscr{A}_2$ follows the remaining steps after step 2 of the simulator $\mathcal{S}_{\text{MPP}}$ given in the Proof (given in Appendix F) of Theorem 4.
4) $\mathscr{A}_2$ sends the $f(\lambda)$ MProve+ proofs to $\mathscr{A}_1$ and receives (P, X, Y). $\mathscr{A}_2$ outputs (P, X, Y).

From the unlinkability property of Monero, $\mathscr{A}_2$ cannot exist. So the PPT adversary $\mathscr{A}_1$ does not exist. Hence MProve+ preserves the unlinkability property of Monero.

V. Performance

We compare our proof of reserves protocol MProve+ with MProve [9] which is the first and only proof of reserves protocol for Monero that attempts to provide some privacy to the prover. 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 own-set size is s. The proof sizes of MProve+ and MProve are respectively $(n + s + 2\log_2 N + 4)$ group elements, 5 scalars and $3n$+2 group elements, $6n$ scalars. 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 own-set size $s = 10^3$, 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 [30]. 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 [31], [32].

Figure 8(b) shows the proof generation and verification times of MProve+ and MProve. For a constant own-set size 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 N_1, N_2 such that $\lceil \log_2 N_1 \rceil = \lceil \log_2 N_2 \rceil$, 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 4 hours in a MProve+ proof generation and the proof verification would take 40 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 7X faster than its generation because the inner product protocol can be verified using a single multi-exponentiation of size $\mathcal{O}(2s\cdot n + 2\log(s\cdot n))$. 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

(a) Proof size in KB for $s = 100$ (log-log) (b) Running times in min, $s = 100$ (log-log) (c) Running times in min, $n = 5000$ (log-log)

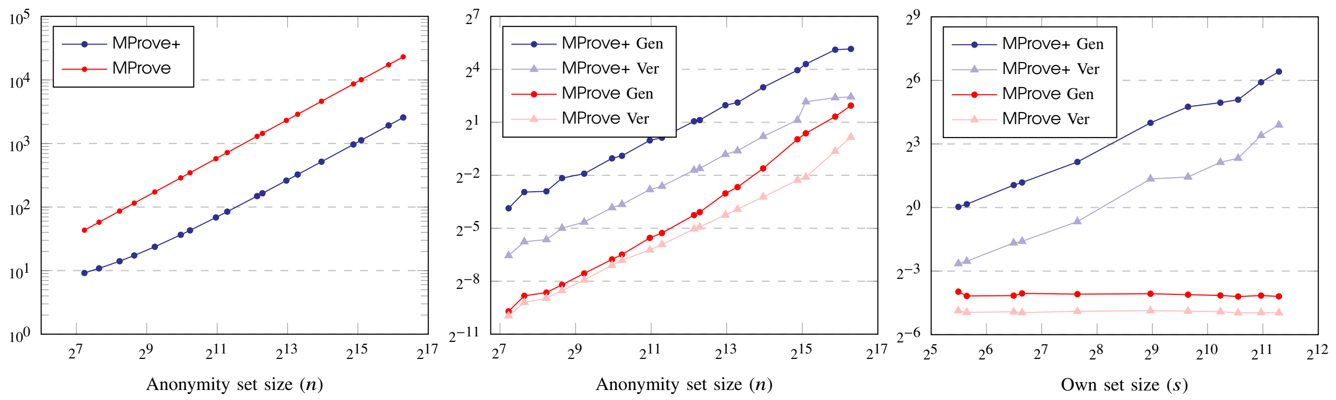

Fig. 8: Performance comparison of MProve+ and MProve for $\mathbb{G}$ = Ristretto elliptic curve.

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 own-set size. An implication of revealing the own-set size 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 own-set size and for a constant anonymity set size. While timings for MProve remain constant, MProve+ timings show a linear growth with s.

VI. Conclusion

We present the MProve+ protocol which gives better privacy than the MProve protocol using techniques of Bulletproofs [19] and Omniring [6]. The MProve+ protocol provides a significant improvement in terms of proof size over the MProve protocol. The performance of MProve+ protocol are 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 E. However, unlike the MProve protocol, the MProve+ protocol does not let a source spending transaction become a zero-mixin transaction. 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.

VII. Acknowledgments

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

Appendix A
A summary of the MProve Protocol

In the MProve protocol, the exchange creates an anonymity set of one-time addresses i.e. $\mathscr{P}_{\text{anon}} = \{P_1, P_2, \ldots, P_n\}$ of which it knows the secret keys corresponding to some addresses. Let $\mathscr{P}_{\text{own}} \subset \mathscr{P}_{\text{anon}}$ be the set of exchange-owned one-time addresses. For each $P_i \in \mathscr{P}_{\text{anon}}$, 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 $P_i \in \mathscr{P}_{\text{anon}}$ such that the following equation holds.

$$C_{\text{res}} = \prod_{i=1}^{n} C_i C_i'^{-1}. \tag{45}$$

To satisfy equation (45), C_i's should be constructed as

$$C_i' = \begin{cases} G^{z_i} & \text{if } P_i \in \mathscr{P}_{\text{own}} \\ G^{z_i} C_i & \text{if } P_i \notin \mathscr{P}_{\text{own}}, \end{cases} \tag{46}$$

where z_is are some randomly chosen scalars from $\mathbb{Z}_q$. So the verifier of the proof will check the equality of equation (45) 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 (46) 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 [33] 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 $C_i'C_i^{-1}$, $C_iC_i'^{-1}$ 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 (45). 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 $\mathcal{I}$ 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 $P_i \in \mathcal{P}_{\text{own}}$ or $P_i \notin \mathcal{P}_{\text{own}}$.

Appendix B
Difference with Omniring

Omniring proposes a protocol for Monero transactions which is similar to the MProve+ protocol. In particular, if we run the protocol given in Appendix F of the Omniring paper [6] for a single output and keeping all the exchange-owned one-time addresses in the input set, that gives us the desired commitment to the total reserves. However, the MProve+ protocol differs in the following ways.

1) Omniring proposes a protocol to validate a Monero transaction, where some coins are being transferred from some source one-time addresses to some destination one-time addresses. While MProve+ proves the ownership of some one-time addresses, it does not involve any transfer of coins. In fact, the MProve+ protocol does not specify any destination addresses.
2) In Omniring, in addition to knowledge of secret keys, knowledge of amounts and blinding factors used in the associated commitments are proved. However, we observe that the latter proofs are unnecessary in a Monero proof of reserves protocol. The knowledge of the secret key is enough to recover the amount and the blinding factor of the associated commitment. Therefore proving knowledge of the secret key is enough to claim ownership of a one-time address and its associated amount in Monero. This is discussed in detail in Section II.B.
3) In Omniring, for every output amount, it is proved that the corresponding bit representation is a binary vector of proper length. This is essentially a range proof. MProve+ does not require such range proof since it proves that the reserve amount is the sum of all valid input amounts which are already proven to be in a proper range. This is because they are taken from the Monero blockchain where the outputs of all transactions have range proofs associated with them.

Appendix C
Security Proofs for $\Pi_{\text{MProve+}}$

In this appendix we give proofs for Theorem 1 and 2. They are motivated by and very similar to Omniring [6] security proofs.

A. *Proof of Theorem 1*

The protocol is public coin under the random oracle model using the Fiat-Shamir heuristics i.e. all verifier challenges (u, v, w, y, x) are generated by hashing the protocol transcript available at the time they are generated. The protocol $\Pi_{\text{MProve+}}$ has 8 rounds of interaction between $\mathcal{P}$ and $\mathcal{V}$. Completeness can be checked by verifying that the following verification equations hold.

$$\hat{t} \stackrel{?}{=} \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle, \tag{47}$$

$$G^{\hat{t}} H^{\tau_x} \stackrel{?}{=} G^{\delta} T_1^x T_2^{x^2}, \tag{48}$$

$$(H')^r \mathbf{G}_w^{\boldsymbol{\ell}} \mathbf{H}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}} \stackrel{?}{=} A S^x \mathbf{G}_w^{\boldsymbol{\pi}} \mathbf{H}^{\boldsymbol{\beta}}. \tag{49}$$

Now we show that $\Pi_{\text{MProve+}}$ is perfect special honest verifier zero-knowledge by constructing an efficient simulator $\mathcal{S}$. Simulator $\mathcal{S}$ does not know witness wit for $\Pi_{\text{MProve+}}$. The job of $\mathcal{S}$ is to generate a transcript for $\Pi_{\text{MProve+}}$ which is indistinguishable from an actual transcript to a PPT adversary $\mathcal{A}$. Let the statement $\mathsf{stmt} = (\mathbf{P}, \mathbf{C}, \mathbf{H}_p, \mathbf{I}, C_{\text{res}}, G_1)$ and verifier challenges (u, v, w, y, z, x) be provided by $\mathcal{A}$. Simulator $\mathcal{S}$ computes $\hat{\mathbf{Y}}, \hat{\mathbf{I}}, \mathbf{G}_w$ as in $\Pi_{\text{MProve+}}$. That is,

$$\hat{\mathbf{Y}} = \mathbf{P}^u \circ \mathbf{H}_p^{\circ u^2}, \tag{50}$$

$$\hat{\mathbf{I}} = \mathbf{I}^{\circ -u^2 \boldsymbol{v}^s}, \tag{51}$$

$$\mathbf{G}_w = \left[\left((G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{\circ w} \circ \mathbf{Q} \right) \| \mathbf{G}' \right]. \tag{52}$$

Now, $\mathcal{S}$ samples $A, T_2 \xleftarrow{\$} \mathbb{G}, \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \xleftarrow{\$} \mathbb{Z}_q^N, \tau_x, r \xleftarrow{\$} \mathbb{Z}_q$. It computes $\hat{t} = \langle \boldsymbol{\ell}, \boldsymbol{\mathfrak{r}} \rangle$. It then computes S and T_1 as follows.

$$S = \left((H')^{-r} A \mathbf{G}_w^{\boldsymbol{\pi} - \boldsymbol{\ell}} \mathbf{H}^{\boldsymbol{\beta} - \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}} \right)^{-\frac{1}{x}}, \quad \text{// from equation (49)} \tag{53}$$

$$T_1 = \left(G^{\delta - \hat{t}} H^{-\tau_x} T_2^{x^2} \right)^{-\frac{1}{x}}. \quad \text{// from equation (48)} \tag{54}$$

By observation, we can say that all elements in the transcripts produced by both $\mathcal{S}$ and $\Pi_{\text{MProve+}}$ are either independently randomly generated or fully determined by the verification equations involving some random and independently generated quantities. The latter case also makes them randomly distributed quantities. So in each case, all elements of the transcripts are identically distributed and hence the transcripts are indistinguishable to $\mathcal{A}$. ■

B. *Proof of Theorem 2*

To prove that $\Pi_{\text{MProve+}}$ has witness-extended emulation, first we state a couple of useful lemmas and a corollary. Before we proceed, we define some notations and a new system of constraint equations CS . Let $\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \in \mathbb{Z}_q^N$ be two vectors of same format as $\mathbf{c}_L$ and $\mathbf{c}_R$. That is for $i \in \{L, R\}$, $\gamma_{i,j} \in \mathbb{Z}_q$ for $j \in \{1, 2, 3\}$, $\boldsymbol{\gamma}_{i,j} \in \mathbb{Z}_q^n$ for $j \in \{4, 5\}$, $\boldsymbol{\gamma}_{i,6} \in \mathbb{Z}_q^s$, and for some matrices $\Gamma_i \in \mathbb{Z}_q^{s \times n}$, $\boldsymbol{\gamma}_{i,7} = \mathsf{vec}(\Gamma_i) \in \mathbb{Z}_q^{sn}$. We now define the constraint system CS with parameter v such that $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0 \iff$

$$\begin{cases} \boldsymbol{\gamma}_{L,7} \circ \boldsymbol{\gamma}_{R,7} & = \mathbf{0}^{sn} & (55) \\ \boldsymbol{\gamma}_{R,6} & = \boldsymbol{\gamma}_{L,6}^{\circ -1} & (56) \\ \gamma_{L,1} + \langle \boldsymbol{\gamma}_{R,6}, \boldsymbol{uv}^s \rangle & = 0 & (57) \\ \boldsymbol{\gamma}_{L,4} & = \boldsymbol{v}^s \Gamma_L & (58) \\ \boldsymbol{\gamma}_{L,5} & = \mathbf{1}^s \Gamma_L & (59) \\ \Gamma_L \mathbf{1}^n & = \mathbf{1}^s & (60) \\ \gamma_{L,2} & = -1 & (61) \\ \boldsymbol{\gamma}_{R,7} & = -\boldsymbol{\gamma}_{L,7} + \mathbf{1}^{sn} & (62) \end{cases}$$

Lemma 1: For $q > 2^\lambda, v \in \mathbb{Z}_q$, suppose there exists $\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \in \mathbb{Z}_q^N$ such that for sn different values of y we have $\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$, then $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$.

Proof: Given $\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$, we have the following equations.

$$\begin{aligned} \langle \boldsymbol{\gamma}_{L,7} \circ \boldsymbol{\gamma}_{R,7}, \boldsymbol{y}^{sn} \rangle &= 0 && \text{by } Eq\ (23), \\ \langle (\boldsymbol{\gamma}_{L,6} \circ \boldsymbol{\gamma}_{R,6}) - \mathbf{1}^s, \boldsymbol{y}^s \rangle &= 0 && \text{by } Eq\ (24), \\ \gamma_{L,1} + \langle \boldsymbol{\gamma}_{R,6}, \boldsymbol{uv}^s \rangle &= 0 && \text{by } Eq\ (25), \\ \langle \boldsymbol{v}^s \Gamma_L - \boldsymbol{\gamma}_{L,4}, \boldsymbol{y}^s \rangle &= 0 && \text{by } Eq\ (26), \\ \langle \mathbf{1}^s \Gamma_L - \boldsymbol{\gamma}_{L,5}, \boldsymbol{y}^s \rangle &= 0 && \text{by } Eq\ (27), \\ -(\gamma_{L,2} + 1) y^s + \langle \Gamma_L \mathbf{1}^n - \mathbf{1}^s, \boldsymbol{y}^s \rangle &= 0 && \text{by } Eq\ (28), \\ \langle \boldsymbol{\gamma}_{L,7} + \boldsymbol{\gamma}_{R,7} - \mathbf{1}^{sn}, \boldsymbol{y}^{sn} \rangle &= 0 && \text{by } Eq\ (29). \end{aligned}$$

It is given that the above equations hold for sn different values of y. However the polynomial in the left hand side of the equations are at most of degree $sn - 1$. Hence all the coefficients of all powers of y of the above equations are zero i.e. all polynomials above are zero polynomials. Equating all coefficients to zero, we get $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$. ∎

Lemma 2: If $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$ then each row of Γ_L is a unit vector of length n.

Proof: This follows from observing equations (55), (60), and (62) of the constraint system CS. ∎

Now we need the following definition of interactive discrete logarithm assumption and the theorem stating its equivalence with the discrete logarithm assumption to proceed further.

Definition 2: (Interactive Discrete Logarithm Assumption.) We say that the interactive discrete logarithm assumption holds over $\mathcal{G}$ if for all $l \in \mathsf{poly}(\lambda)$, every PPT adversary $\mathcal{A}$

$$\Pr[iDL_{\mathcal{A},l}(\mathcal{G})] \leq \mathsf{negl}(\lambda),$$

where negl is a negligible function of the security parameter λ and the game $iDL_{\mathcal{A},l}(\mathcal{G})$ is defined by the following sequence of events.

1) $\mathcal{A}$ having group description $(\mathbb{G}, q, G)$ as input produces a vector generator $\mathbf{G} \in \mathbb{G}^l$ and sends to a challenger.
2) The challenger on receiving $\mathbf{G}$, generates a random vector $\mathbf{H} \xleftarrow{\$} \mathbb{G}^l$ and sends it to $\mathcal{A}$.
3) On receiving $\mathbf{H}$, $\mathcal{A}$ outputs $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^l$ such that

$$1_g = \mathbf{G}^{\mathbf{a}} \mathbf{H}^{\mathbf{b}},$$

where $\mathbf{b} \neq \mathbf{0}^l$.

Theorem 5: The interactive discrete logarithm assumption holds over $\mathcal{G}$ if and only if the discrete logarithm assumption holds over $\mathcal{G}$.

Proof: See Theorem A.6 in the Omniring paper [6]. ∎

Next we present the following corollary which is similar to Corollary 1 in Omniring [6]. The corollary will be used to extract elements of witness wit.

Corollary 1: Assuming the discrete logarithm assumption holds over $\mathbb{G}$, if there exists a PPT adversary $\mathcal{A}$ who does the following with a non-negligible probability:

1) On input $(\mathbb{G}, q, G)$, choose a vector of group elements $(C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})$ and a scalar $w \in \mathbb{Z}_q$.
2) Receive a uniformly random vector $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$ of appropriate dimensions.
3) Produce a non-zero scalar vector $(r \| \mathbf{a} \| \mathbf{b})$ such that $1_g = (H')^r \mathbf{G}_w^{\mathbf{a}} \mathbf{H}^{\mathbf{b}}$, where
$\mathbf{G}_w = \left[\left((G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{\circ w} \circ \mathbf{Q} \right) \| \mathbf{G}' \right]$

then there exists a PPT algorithm which solves the discrete logarithm problem over $\mathbb{G}$ with a non-negligible probability.

Proof: Suppose there exists a PPT adversary $\mathcal{A}$ which succeeds in the above experiment with a non-negligible probability. We shall construct a PPT adversary $\mathcal{B}$ which solves the interactive discrete logarithm problem using $\mathcal{A}$ as a subroutine. Then by Theorem 5, $\mathcal{B}$ should be able to solve the discrete logarithm problem with a non-negligible probability. Suppose there exists a challenger $\mathcal{C}$ which wants to test $\mathcal{B}$ with the interactive discrete logarithm problem. $\mathcal{C}$ and $\mathcal{B}$ proceeds as follows.

1) $\mathcal{B}$ receives $(\mathbb{G}, q, G)$ from $\mathcal{C}$ and sends to $\mathcal{A}$.
2) It then receives a vector of group elements $(C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})$ and a scalar w from $\mathcal{A}$.
3) $\mathcal{B}$ then passes $(G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})$ to $\mathcal{C}$. This is the first vector generator in the interactive discrete logarithm experiment. $\mathcal{B}$ receives from $\mathcal{C}$ a uniformly random vector of group elements $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$. This is the second vector generator of the interactive discrete logarithm game. $\mathcal{B}$ passes $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$ to $\mathcal{A}$.
4) On receiving $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$, $\mathcal{A}$ produces a non-zero scalar vector $(r \| \mathbf{a} \| \mathbf{b})$ such that $1_g = (H')^r \mathbf{G}_w^{\mathbf{a}} \mathbf{H}^{\mathbf{b}}$ and forwards this to $\mathcal{B}$.
5) Let $\mathbf{a} = (\mathbf{a}_1, \mathbf{a}_2)$ be of appropriate dimensions. As $\mathbf{G}_w = \left[\left((G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{\circ w} \circ \mathbf{Q} \right) \| \mathbf{G}' \right]$ and $1_g = (H')^r \mathbf{G}_w^{\mathbf{a}} \mathbf{H}^{\mathbf{b}}$ we can write

$$1_g = (G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{w \cdot \mathbf{a}_1} (H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})^{(r \| \mathbf{a} \| \mathbf{b})}.$$

Since $(r \| \mathbf{a} \| \mathbf{b})$ is non-zero, $(w \cdot \mathbf{a}_1, r \| \mathbf{a} \| \mathbf{b})$ is a valid solution to the interactive discrete logarithm problem with respect to the bases $(G \| C_{\text{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})$ and $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$. Therefore, $\mathcal{B}$ sends $(w \cdot \mathbf{a}_1, r \| \mathbf{a} \| \mathbf{b})$ to $\mathcal{C}$.

Therefore if $\mathcal{A}$ can succeed in the above experiment with a non-negligible probability, then there exists a PPT adversary which can solve the discrete logarithm problem over $\mathbb{G}$ with a non-negligible probability. ∎

With the above lemmas and the corollary, we shall construct an extractor $\mathcal{E}$. Suppose a PPT challenger $\mathcal{C}$ generates common reference string, statement, and witness as follows: $\mathsf{crs} \leftarrow \mathsf{Setup}(\lambda)$ and $(\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathcal{C}(\mathsf{crs})$. $\mathcal{C}$ provides crs and stmt to

$\mathscr{E}$. $\mathscr{E}$ also has oracle access to $\langle \mathscr{P}^\star(\mathsf{crs}, \mathsf{stmt}; \mathsf{wit}), \mathcal{V}(\mathsf{crs}, \mathsf{stmt})\rangle$ for any prover $\mathscr{P}^\star$. The aim of $\mathscr{E}$ is to produce an accepting transcript and consequently the witness wit' corresponding to that transcript. As $\mathscr{E}$ has oracle access to prover $\mathscr{P}^\star$, producing an accepting transcript is trivial for $\mathscr{E}$. So we focus on how $\mathscr{E}$ could extract a valid witness for an accepting transcript.

Extractor $\mathscr{E}$ runs $\mathscr{P}^\star$ on one uniformly chosen challenge (u, v), 2 different values of w, sn different values of y, 7 different values of z, and 3 different values of x. This results in $42 \times sn$ transcripts. $\mathscr{E}$ fixes the values of (w, y, z) and runs $\mathscr{P}^\star$ for $x = (x_1, x_2, x_3)$. Let the transcripts for the respective x be $(A, S, T_1, T_2, \tau_{x_i}, r_{x_i}, \boldsymbol{\ell}_{x_i}, \boldsymbol{\mathfrak{r}}_{x_i}, \hat{t}_{x_i})$ for $i = 1, 2, 3$. Now $\mathscr{E}$ will extract the discrete logarithm representations of A, S, T_1, T_2 using the above transcripts. From those representations we can obtain wit.

Extracting A: Choose $k_i \in \mathbb{Z}_q$ for $i = 1, 2$ such that $\sum_{i=1}^2 k_i = 1$ and $\sum_{i=1}^2 k_i x_i = 0$. Thus, we write from verification equation which is reproduced in equation (49),

$$
\begin{aligned}
A^{k_i} &= (H')^{r_{x_i} k_i} S^{-x_i k_i} \mathbf{G}_w^{k_i \cdot (\boldsymbol{\ell}_{x_i} - \boldsymbol{\pi})} \mathbf{H}^{k_i \cdot (\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i} - \boldsymbol{\beta})}, \forall i \in \{1, 2\} \\
\implies \prod_{i=1}^{2} A^{k_i} &= (H')^{\sum_i r_{x_i} k_i} S^{-\sum_i x_i k_i} \mathbf{G}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\pi}(\sum_i k_i)} \\
&\quad \mathbf{H}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}(\sum_i k_i)} \\
\implies A &= (H')^{\sum_i r_{x_i} k_i} \mathbf{G}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\pi}} \mathbf{H}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}} \\
\implies A &= (H')^{r'_A} \mathbf{G}_w^{\mathbf{c}'_L} \mathbf{H}^{\mathbf{c}'_R},
\end{aligned}
$$

where $r'_A = \sum_i r_{x_i} k_i$, $\mathbf{c}'_L = (\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\pi}$, and $\mathbf{c}_R = (\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}$. Since we have considered the above extraction for a particular w out of the 2 of its values, $r'_A, \mathbf{c}'_L, \mathbf{c}'_R$ depend on w. To show that the discrete logarithm representation of A is independent of w, $\mathscr{E}$ repeats the above for a different w'. In particular, we have $A = H^{r''_A} \mathbf{G}_{w'}^{\mathbf{c}''_L} \mathbf{H}^{\mathbf{c}''_R}$. Now we have two possibly different representations of A. We write $\mathbf{c}'_L = (\mathbf{c}'_{L,1} \| \mathbf{c}'_{L,2})$ and $\mathbf{c}''_L = (\mathbf{c}''_{L,1} \| \mathbf{c}''_{L,2})$ with appropriate dimensions. We have

$$
\begin{aligned}
(H')^{r'_A} \mathbf{G}_w^{\mathbf{c}'_L} \mathbf{H}^{\mathbf{c}'_R} &= (H')^{r''_A} \mathbf{G}_{w'}^{\mathbf{c}''_L} \mathbf{H}^{\mathbf{c}''_R} \\
\implies 1_g &= (H')^{r'_A - r''_A} \mathbf{G}_w^{\mathbf{c}'_L - \mathbf{c}''_L} \mathbf{H}^{\mathbf{c}'_R - \mathbf{c}''_R} \\
\implies 1_g &= (G \| C_{\mathrm{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{w \cdot \mathbf{c}'_{L,1} - w' \cdot \mathbf{c}''_{L,1}} \\
&\quad \left((H')^{r'_A - r''_A} (\mathbf{Q} \| \mathbf{G}')^{\mathbf{c}'_L - \mathbf{c}''_L} \mathbf{H}^{\mathbf{c}'_R - \mathbf{c}''_R} \right).
\end{aligned}
$$

Now, suppose $r'_A \neq r''_A$, $\mathbf{c}'_L \neq \mathbf{c}''_L$, $\mathbf{c}'_R \neq \mathbf{c}''_R$. But $(H' \| \mathbf{Q} \| \mathbf{G}' \| \mathbf{H})$ is uniformly chosen after fixing $(G \| C_{\mathrm{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})$. So we have violated the interactive discrete logarithm assumption as well as the discrete logarithm assumption according to Theorem 5. Thus, $r'_A = r''_A$, $\mathbf{c}'_L = \mathbf{c}''_L$, $\mathbf{c}'_R = \mathbf{c}''_R$. Let $\mathbf{c}'_{L,1} = (\xi' \| \psi' \| \gamma' \| \hat{\mathbf{e}}' \| \mathbf{e}'' \| (\boldsymbol{x}^{\circ -1})')$, we get

$$
\begin{aligned}
1_g &= (G \| C_{\mathrm{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{(w - w') \cdot \mathbf{c}'_{L,1}} \\
\implies 1_g &= (G \| C_{\mathrm{res}} \| G_1 \| \hat{\mathbf{Y}} \| \mathbf{C} \| \hat{\mathbf{I}})^{\mathbf{c}'_{L,1}}, \text{ since } w \neq w' \\
\implies 1_g &= G^{\xi'} \cdot C_{\mathrm{res}}^{\psi'} \cdot G_1^{\gamma'} \cdot \hat{\mathbf{Y}}^{\hat{\mathbf{e}}'} \cdot \mathbf{C}^{\mathbf{e}''} \cdot \hat{\mathbf{I}}^{(\boldsymbol{x}^{\circ -1})'}. \qquad (63)
\end{aligned}
$$

We shall use equation (63) to show the validity of the extracted witnesses.

Extracting S: Similar to the extraction of A, $\mathscr{E}$ again samples some $k_1, k_2 \in \mathbb{Z}_q$ such that $k_1 + k_2 = 0$ and $k_1 x_1 + k_2 x_2 = 1$. From equation (49), we have

$$
\begin{aligned}
S^{x_i} &= H^{r_{x_i}} A^{-1} \mathbf{G}_w^{\boldsymbol{\ell}_{x_i} - \boldsymbol{\pi}} \mathbf{H}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i} - \boldsymbol{\beta}}, \forall i \in \{1, 2\} \\
\implies \prod_{i=1}^{2} S^{k_i x_i} &= H^{\sum_i k_i r_{x_i}} A^{-\sum_i k_i} \mathbf{G}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - (\sum_i k_i) \boldsymbol{\pi}} \\
&\quad \mathbf{H}^{(\sum_i k_i \cdot \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - (\sum_i k_i) \boldsymbol{\beta}} \\
\implies S &= H^{r'_S} \mathbf{G}_w^{\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}} \mathbf{H}^{\sum_i k_i \cdot \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}} \\
\implies S &= H^{r'_S} \mathbf{G}_w^{\mathbf{s}'_L} \mathbf{H}^{\mathbf{s}'_R},
\end{aligned}
$$

where $r'_S = \sum_i k_i r_{x_i}$, $\mathbf{s}'_L = \sum_i k_i \cdot \boldsymbol{\ell}_{x_i}$, and $\mathbf{s}'_R = \sum_i k_i \cdot \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}$. For a fixed w, the extracted A, S hold for all possible (x, y, z). Otherwise a non-trivial discrete logarithm representation of 1_g with respect to the base $(H' \| \mathbf{G}_w \| \mathbf{H})$ would be known which is not possible owing to Corollary 1.

Substituting A, S in equation (49), we get

$$
\begin{aligned}
\boldsymbol{\ell}'_x &= \mathbf{c}'_L + \boldsymbol{\pi} + \mathbf{s}'_L \cdot x \in \mathbb{Z}_q^N, \\
\boldsymbol{\mathfrak{r}}'_x &= \boldsymbol{\theta} \circ (\mathbf{c}'_R + \mathbf{s}'_R \cdot x) + \boldsymbol{\zeta} \in \mathbb{Z}_q^N.
\end{aligned}
$$

These vectors are same for all (x, y, z) otherwise we would know a non-trivial discrete logarithm representation of 1_g with respect to the base $(H' \| \mathbf{G}_w \| \mathbf{H})$ which is not possible due to Corollary 1.

Extracting T_1, T_2: From equation (48) we get the following equations.

$$
\begin{aligned}
T_1^x &= G^{\hat{t} - \delta} H^{\tau_x} T_2^{-x^2}, \qquad (64) \\
T_2^{x^2} &= G^{\hat{t} - \delta} H^{\tau_x} T_1^{-x}. \qquad (65)
\end{aligned}
$$

To extract T_1, $\mathscr{E}$ chooses $k_i \in \mathbb{Z}_q$ for $i \in \{1, 2, 3\}$ such that $\sum_{i=1}^3 k_i = 0$, $\sum_{i=1}^3 k_i x_i = 1$ and $\sum_{i=1}^3 k_i x_i^2 = 0$. Exponentiating equation (64) with all k_is and multiplying them all gives

$$
\prod_{i=1}^{3} T_1^{k_i x_i} = G^{\sum_{i=1}^3 k_i \hat{t}_{x_i}} H^{\sum_{i=1}^3 k_i \tau_{x_i}} \implies T_1 = G^{t'_1} H^{\tau'_1} \qquad (66)
$$

Similarly, to extract T_2, $\mathscr{E}$ chooses $k'_i \in \mathbb{Z}_q$ for $i \in \{1, 2, 3\}$ such that $\sum_{i=1}^3 k'_i = 0$, $\sum_{i=1}^3 k'_i x_i = 0$ and $\sum_{i=1}^3 k'_i x_i^2 = 1$. Then exponentiating equation (65) with all k'_i and multiplying all of them gives

$$
\prod_{i=1}^{3} T_2^{k'_i x_i^2} = G^{\sum_{i=1}^3 k'_i \hat{t}_{x_i}} H^{\sum_{i=1}^3 k'_i \tau_{x_i}} \implies T_2 = G^{t'_2} H^{\tau'_2}. \qquad (67)
$$

Again, the above expressions for T_1, T_2 hold for all x, or otherwise we would have obtained a non-trivial discrete logarithm representation of 1_g with respect to base $(G \| H)$ which directly violates the discrete logarithm assumption.

Extracting witness: $\mathscr{E}$ parses $\mathbf{c}'_L = (\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\pi}$ and outputs witness wit' as follows.

$$
\begin{aligned}
\mathbf{c}'_L &= (\xi' \| \psi' \| \gamma' \| \hat{\mathbf{e}}' \| \mathbf{e}'' \| (\boldsymbol{x}')^{\circ -1} \| \mathsf{vec}(\mathcal{E}')) \\
\mathsf{wit}' &= (\boldsymbol{x}', \mathbf{e}'_1, \ldots, \mathbf{e}'_s, \gamma').
\end{aligned}
$$

Here $\boldsymbol{x}'$ can be generated by inverting elements of scalar vector $(\boldsymbol{x}')^{\circ -1}$ and $\mathbf{e}'_1, \ldots, \mathbf{e}'_s$ can be generated by taking n elements of $\mathsf{vec}(\mathcal{E}')$ at a time.

Showing well-formedness: Next we will show that the extracted witness wit' is a valid witness for the statement stmt. We will use $\delta(u, v, y, z)$ to make the dependence of δ on the challenges explicit. Putting the value of extracted T_1 and T_2 in equation (48) and comparing the exponent of G gives us the following equation.

$$t'_x = \delta(u, v, y, z) + t'_1 x + t'_2 x^2,$$

for all (x, y, z) or else we would violate the discrete logarithm assumption by having a discrete logarithm relation between G and H. Let

$$\begin{aligned} t'_0 &:= \delta(u, v, y, z), \\ l'(X) &:= \mathbf{c}'_L + \boldsymbol{\pi} + \mathbf{s}'_L \cdot X, \\ r'(X) &:= \boldsymbol{\theta} \circ (\mathbf{c}'_R + \mathbf{s}'_R \cdot X) + \boldsymbol{\zeta}, \\ t'(X) &:= \langle l'(X), r'(X) \rangle. \end{aligned}$$

Now consider the polynomial $t'(X) - (t'_0 + t'_1 X + t'_2 X^2)$. For all (y, z), it has at least 3 roots because of 3 choices of x. But it is of degree 2. Hence it must be a zero polynomial. So we have $t'(X) = t'_0 + t'_1 X + t'_2 X^2$ and particularly, $t'(0) = t'_0$. The latter two quantities are given by,

$$\begin{aligned} t'_0 = \delta(u, v, y, z) &= z \cdot \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle + z^5 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle \\ &\quad + \langle \boldsymbol{\pi}, \boldsymbol{\zeta} \rangle + z^6 \langle \mathbf{1}^{sn}, \mathbf{v}_6 \rangle, \qquad (68) \\ t'(0) &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \langle \boldsymbol{\pi}, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \boldsymbol{\pi}, \boldsymbol{\zeta} \rangle, \\ &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \langle \boldsymbol{\theta} \circ \boldsymbol{\pi}, \mathbf{c}'_R \rangle + \langle \boldsymbol{\pi}, \boldsymbol{\zeta} \rangle, \\ &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \langle \mathbf{c}'_R, \boldsymbol{\nu} \rangle + \langle \boldsymbol{\pi}, \boldsymbol{\zeta} \rangle. \qquad (69) \end{aligned}$$

Equating equation (68) and (69) we get,

$$\begin{aligned} z \cdot \langle \mathbf{1}^s, \boldsymbol{y}^s \rangle + z^5 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle \\ &\quad + \langle \mathbf{c}'_R, \boldsymbol{\nu} \rangle - z^6 \langle \mathbf{1}^{sn}, \mathbf{v}_6 \rangle, \\ = \sum_{i=0}^{1} z^i \langle \mathbf{c}'_L, \mathbf{c}'_R \circ \mathbf{v}_i \rangle &+ \sum_{i=2}^{5} z^i \langle \mathbf{c}'_L, \mathbf{v}_i \rangle + z^2 \langle \mathbf{c}'_R, \mathbf{v}_7 \rangle \\ + z^6 \langle \mathbf{c}'_L - \mathbf{c}'_R - \mathbf{1}^{sn}, \mathbf{v}_6 \rangle. & \qquad (70) \end{aligned}$$

The above equation can be obtained multiplying the ith equation of $\mathsf{EQ}(\mathbf{c}'_L, \mathbf{c}'_R)$ (defined in Figure 3) with z^{i-1} and adding them together. The summation given in equation (70) is a polynomial of degree 6 in z and is equated to zero for 7 different values of z. So $\mathsf{EQ}(\mathbf{c}'_L, \mathbf{c}'_R) = 0$ is satisfied for sn different values of y as each equation represents a coefficient of a zero polynomial. Hence by Lemma 1, we have $\mathsf{CS}(\mathbf{c}'_L, \mathbf{c}'_R) = 0$. Then by Lemma 2, each row vector of $\mathscr{E}'$ is a unit vector of length n. Let $\mathsf{vec}(\mathscr{E}') = (\mathbf{e}'_1, \ldots, \mathbf{e}'_s)$. By $\mathsf{CS}(\mathbf{c}'_L, \mathbf{c}'_R) = 0$, we also have,

$$\begin{aligned} \xi' &= -\langle \boldsymbol{x}', u\boldsymbol{v}^s \rangle && \text{by } Eq\ (57), \\ \psi' &= -1 && \text{by } Eq\ (61), \\ \hat{\mathbf{e}}' &= \boldsymbol{v}^s \mathscr{E}' = \sum_{l \in [s]} v^{l-1} \boldsymbol{e}'_l && \text{by } Eq\ (58), \\ \mathbf{e}'' &= \mathbf{1}^s \mathscr{E}' = \sum_{l \in [s]} \boldsymbol{e}'_l && \text{by } Eq\ (59). \end{aligned}$$

From equation (63), we get

$$\begin{aligned} 1_g &= G^{\xi'} \cdot C_{\text{res}}^{\psi'} \cdot G_1^{\gamma'} \cdot \hat{\mathbf{Y}}^{\hat{\mathbf{e}}'} \cdot \mathbf{C}^{\mathbf{e}''} \cdot \hat{\mathbf{I}}^{(\boldsymbol{x}^{\circ-1})'}, \\ &= G^{-\langle \boldsymbol{x}', u\boldsymbol{v}^s \rangle} C_{\text{res}}^{-1} G_1^{\gamma'} \left(\mathbf{P}^u \circ \mathbf{H}_p^{\circ u^2} \right)^{\boldsymbol{v}^s \mathscr{E}'} \mathbf{C}^{\mathbf{1}^s \mathscr{E}'} \prod_{l \in [s]} I_l^{-\frac{u^2 v^{l-1}}{x'_l}}, \\ &= \prod_{l \in [s]} G^{-x'_l u v^{l-1}} \left(C_{\text{res}}^{-1} G_1^{\gamma'} \prod_{l \in [s]} \mathbf{C}^{\mathbf{e}'_l} \right) \prod_{l \in [s]} \mathbf{P}^{u v^{l-1} \mathbf{e}'_l} \\ &\qquad \prod_{l \in [s]} \mathbf{H}_p^{u^2 v^{l-1} \mathbf{e}'_l} \prod_{l \in [s]} I_l^{-\frac{u^2 v^{l-1}}{x'_l}}, \\ &= \prod_{l \in [s]} \left(G^{-x'_l} \mathbf{P}^{\mathbf{e}'_l} \right)^{u v^{l-1}} \prod_{l \in [s]} \left(I_l^{\frac{-1}{x'_l}} \mathbf{H}_p^{\mathbf{e}'_l} \right)^{u^2 v^{l-1}} \\ &\qquad \left(C_{\text{res}}^{-1} G_1^{\gamma'} \prod_{l \in [s]} \mathbf{C}^{\mathbf{e}'_l} \right). \end{aligned}$$

The last equality can be viewed as an evaluation of a degree $(s + 1)$ polynomial (in the exponent) at a random point (u, v) to zero. By the Schwartz-Zippel lemma, the probability that this happens when the polynomial is non-zero (bases are not 1_g) is bounded by $\frac{s+1}{q}$ which is negligible as $q > 2^\lambda$. We can therefore assume that the polynomial is always zero. So for all $l \in [s]$, the following equations hold.

$$\begin{aligned} \mathbf{P}^{\mathbf{e}'_l} &= G^{x'_l} \\ \mathbf{H}_p^{\mathbf{e}'_l} &= I_l^{\frac{1}{x'_l}} \\ C_{\text{res}} &= G_1^{\gamma'} \prod_{l \in [s]} \mathbf{C}^{\mathbf{e}'_l} \end{aligned}$$

Hence we can say that wit' extracted by extractor $\mathscr{E}$ is indeed a valid witness corresponding to statement stmt. ■

Appendix D
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 equation (ii) and the inner product argument. As noted in [19], an inner product argument $\Pi_{\mathsf{IP}} = \left(\{L_j, R_j\}_{j=1}^{\log_2 N} \in \mathbb{G}, a, b \in \mathbb{Z}_q \right)$ for the language in (33) can be verified in a single multi-exponentiation check as

$$\mathbf{G}^{a \cdot \mathbf{s}} \cdot \mathbf{H}^{a \cdot \mathbf{s}^{\circ-1}} \cdot U^{a \cdot b} = P \cdot \prod_{j=1}^{\log_2 N} L_j^{x_j^2} \cdot R_j^{x_j^{-2}}. \qquad (71)$$

where $\mathbf{s} = \{s_i\}_{i=1}^N, s_i = \prod_{j=1}^{\log_2 N} x_j^{b(i,j)}$ 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 $\{x_j\}_{j=1}^{\log_2 N}$. For the inner product argument associated with $\Pi_{\mathsf{MProve+}}$, substituting the expression of P from (34), we get

$$\mathbf{G}_w^{a \cdot \mathbf{s}} \cdot \mathbf{H}^{b \cdot (\boldsymbol{\theta} \circ \mathbf{s})^{\circ-1}} \cdot U^{a \cdot b} = \left(U^{\hat{t}} (H')^{-r} A S^x \mathbf{G}_w^{\boldsymbol{\pi}} \mathbf{H}^{\boldsymbol{\beta}} \right) \cdot \prod_{j=1}^{\log_2 N} L_j^{x_j^2} \cdot R_j^{x_j^{-2}}$$

Moving everything to the LHS, we get

$$\mathbf{G}_w^{a\cdot\mathbf{s}-\boldsymbol{\pi}}\cdot\mathbf{H}^{b\cdot(\boldsymbol{\theta}\circ\mathbf{s})^{\circ-1}-\boldsymbol{\beta}}\cdot U^{a\cdot b-\hat{t}}\cdot(H')^r\cdot A^{-1}\cdot S^{-x}\cdot\prod_{j=1}^{\log_2 N}L_j^{-x_j^2}\cdot R_j^{-x_j^{-2}}=1. \quad (72)$$

Thus, using a single multi-exponentiation of size $2N + 2\log_2 N + 4$, the inner product argument of $\Pi_{\mathsf{MProve+}}$ can be verified.[†††] Furthermore, we can combine the verification equations (ii) and (72) using a random scalar $e \xleftarrow{\$} \mathbb{Z}_q$ as

$$\mathbf{G}_w^{a\cdot\mathbf{s}-\boldsymbol{\pi}}\cdot\mathbf{H}^{a\cdot(\boldsymbol{\theta}\circ\mathbf{s})^{\circ-1}-\boldsymbol{\beta}}\cdot U^{a\cdot b-\hat{t}}\cdot(H')^r\cdot A^{-1}\cdot S^{-x}\cdot\prod_{j=1}^{\log_2 N}L_j^{-x_j^2}\cdot R_j^{-x_j^{-2}}\cdot\left(G^{\delta-\hat{t}}\cdot H^{-\tau_x}\cdot T_1^x\cdot T_2^{x^2}\right)^e=1$$

Thus, the ZK argument of knowledge $\Pi_{\mathsf{MProve+}}$ can be verified using a single multi-exponentiation equation of size $2N + 2\log_2 N + 8$. Using Pippenger's algorithm [34], multi-exponentiations can be computed efficiently, resulting in faster verification of MProve+ proofs of reserves.

Appendix E
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,

$$I = H_p(P)^{H(B_{\text{vk}}^r)+b_{\text{sk}}}. \quad (73)$$

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 $\text{TX}(P) = \{txn_1, txn_2, \ldots, txn_n\}$ represents the set of all transactions where P has appeared as a ring member. Each transaction in $\text{TX}(P)$ has one or more key images. UnspentProof proves that each such key image in $\text{TX}(P)$ 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_{vk}^r. Let $I_?$ be a key image which has appeared in one of the transactions in $\text{TX}(P)$. Using $I_?$, B_{vk}^r, and P, the verifier computes the following quantity, termed as the *partial spend image*.

$$I_{?,s} = I_? H_p(P)^{-H(B_{\text{vk}}^r)}. \quad (74)$$

From equation (73) and (74), if $I = I_?$ i.e. P is spent in the transaction being tested, then we have,

$$I_{?,s} = H_p(P)^{b_{\text{sk}}}. \quad (75)$$

[†††]Note that the inner product argument holds only for vectors with size a power of 2. In cases otherwise (as with N in MProve+), we need to append the secrets with 0's and accordingly change the sizes of base vectors.

UnspentProof consists of two multi-base proof of knowledge signatures which are non-interactive Schnorr-like proofs [35], [36]. They involve sets of base elements and public keys. First, there is a three-base signature σ_3 with a base set $\{G, B_{\text{sk}}, I_{?,s}\}$. 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,

$$Q = G^k \wedge R = B_{\text{sk}}^k \wedge S = I_{?,s}^k. \quad (76)$$

Here, σ_3 is signed with b_{sk} i.e. $k = b_{\text{sk}}$. So we have the set of public keys $\{Q, R, S\} = \{B_{\text{sk}}, B_{\text{sk}}^{b_{\text{sk}}}, I_{?,s}^{b_{\text{sk}}}\}$. 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,

$$X = G^{k'} \wedge Y = H_p(P)^{k'}, \quad (77)$$

where $\{X, Y\}$ is the set of public keys for σ_2. Here, σ_2 is signed with $b_{\text{sk}} * b_{\text{sk}}$ i.e. $k' = b_{\text{sk}} * b_{\text{sk}}$. So the set of public keys for σ_2 is $\{X, Y\} = \{B_{\text{sk}}^{b_{\text{sk}}}, H_p(P)^{b_{\text{sk}}*b_{\text{sk}}}\}$. When σ_3 is signed with b_{sk} and σ_2 is signed with $b_{\text{sk}} * b_{\text{sk}}$, we have $S = I_{?,s}^{b_{\text{sk}}}$ and $Y = H_p(P)^{b_{\text{sk}}*b_{\text{sk}}}$. From equation (75), 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 $b_{\text{sk}} * b_{\text{sk}}$. 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 $\text{TX}(P)$. In each case, the same σ_2 can be used whereas σ_3 changes every time. If the verification passes for every transaction in $\text{TX}(P)$, 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_{vk}^r 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 $I_? H_p(P)^{-H(B_{\text{vk}}^r)}$ 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.[‡‡‡]
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.

[‡‡‡]As of July 24, 2020, there are about 3.58×10^7 key images in the Monero blockchain [37]. Also, the average number of transactions per day (in the last one year) in Monero is ≈ 9000 [38]. Since each transaction contains 2 inputs on an average, the set $\mathcal{I}$ grows approximately by 18,000 every day.

Appendix F
Proof of Theorem 4

Proof: Let the $f(\lambda)$ MProve+ proofs $\{\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}^{(i)}_p, \mathbf{I}^{(i)}, C^{(i)}_{\mathrm{res}}, G_1, \Pi^{(i)}_{\mathsf{MPP}}\}_{i=1}^{f(\lambda)}$ 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 $\left(\{\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}^{(i)}_p, \mathbf{I}^{(i)}\}_{i=1}^{f(\lambda)}, G_1\right)$ 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 $C'^{(i)}_{\mathrm{res}} \xleftarrow{\$} \mathbb{G}$ 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 sends $\mathsf{stmt}^{(i)} = \left(\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}^{(i)}_p, \mathbf{I}'^{(i)}, C'^{(i)}_{\mathrm{res}}, G_1\right)$ and the challenges $(u_i, v_i, w_i, y_i, z_i, x_i)$ to the simulator $\mathcal{S}$ in the Proof of Theorem 1. $\mathcal{S}_{\mathsf{MPP}}$ obtains the ith transcript as $\Pi^{(i)}_{\mathsf{MPP,S}} = (A^{(i)}_{\mathsf{S}}, S^{(i)}_{\mathsf{S}}, T^{(i)}_{1,\mathsf{S}}, T^{(i)}_{2,\mathsf{S}}, \boldsymbol{\ell}^{(i)}_{\mathsf{S}}, \boldsymbol{\mathfrak{r}}^{(i)}_{\mathsf{S}}, \hat{t}^{(i)}_{\mathsf{S}}, \tau^{(i)}_{x,\mathsf{S}}, r^{(i)}_{\mathsf{S}})$ from $\mathcal{S}$.
5) $\mathcal{S}_{\mathsf{MPP}}$ outputs the simulated proof as $\mathsf{MPP}_{\mathsf{sim}} = \{\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}^{(i)}_p, \mathbf{I}'^{(i)}, C'^{(i)}_{\mathrm{res}}, G_1, \Pi^{(i)}_{\mathsf{MPP,S}}\}_{i=1}^{f(\lambda)}$.

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). Next, we shall prove the following claim.

Claim 1: For any PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a negligible function $\mathsf{negl}(\lambda)$, such that

$$\left| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{act}}) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathsf{MPP}_{\mathsf{sim}}) = 1] \right| \leq \mathsf{negl}(\lambda). \tag{78}$$

Let us consider the elements of $\mathsf{MPP}_{\mathsf{act}}$ and $\mathsf{MPP}_{\mathsf{sim}}$. For each $i \in f(\lambda)$, $\mathbf{P}^{(i)}, \mathbf{C}^{(i)}, \mathbf{H}_p^{(i)}, G_1$ are common in both of them. $C_{\mathrm{res}}^{(i)}$ and $C'^{(i)}_{\mathrm{res}}$ are distributed uniformly in $\mathbb{G}$. By an argument similar to that is given in the Proof of Theorem 1, the distribution of the transcript $\Pi^{(i)}_{\mathsf{MPP,S}}$ is computationally indistinguishable to the distribution of the transcript $\Pi^{(i)}_{\mathsf{MPP}}$.

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

$$\mathbf{I}_c = \{I_1, I_2, \ldots, I_N\}, \tag{79}$$

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

$$\mathcal{P}_{\mathrm{orig}} = \{\mathcal{P}_{\mathrm{orig}}(I_1), \mathcal{P}_{\mathrm{orig}}(I_2), \ldots, \mathcal{P}_{\mathrm{orig}}(I_N)\}, \tag{80}$$

$$\mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}) = \{\mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_1)), \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_2)), \ldots, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_N))\}, \tag{81}$$

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

$$\mathcal{P}_{\mathrm{orig}}(I_k) = \mathcal{P}_{\mathrm{orig}}(I'_k), \quad \forall I_k \in \mathbf{I}_c, I'_k \in \mathbf{I}'_c, k \in [N]. \tag{82}$$

From the above discussion, it is clear that to prove Claim 1, it is enough[§§§] to prove that for every PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a negligible function $\mathsf{negl}'(\lambda)$ such that,

$$\Big| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}'_c) = 1] \Big| \leq \mathsf{negl}'(\lambda). \tag{83}$$

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

$$\mathcal{P}_{\mathrm{orig}}(I_j) = (P_1, P_2, \ldots, P_{o_j}), \quad o_j \in \mathbb{Z}_q. \tag{84}$$

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

$$P_{m_j} = G^{x_{m_j}} \wedge I_j = H_p(P_{m_j})^{x_{m_j}}. \tag{85}$$

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 (85),

$$\left(P_{m_j} = G^{x_{m_j}}, \quad H_p(P_{m_j}) = G^{y_j}, \quad I_j = G^{x_{m_j} y_j}\right). \tag{86}$$

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 $(\mathcal{P}_{\mathrm{orig}}(I_j), \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_j)), I_j)$ 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 triple in $(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c)$. However, there is no such DDH triple in $(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}'_c)$. Suppose for a PPT distinguisher $\mathcal{D}_{\mathsf{MPP}}$, there exists a polynomial $p(\lambda)$ such that,

$$\Big| \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c) = 1] - \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}'_c) = 1] \Big| \geq \frac{1}{p(\lambda)}. \tag{87}$$

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 $\mathcal{S}^{(n)}_{\mathrm{hyb}}$ which is given $(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c)$, $n \in \{0, 1, 2, \ldots, N\}$. $\mathcal{S}^{(n)}_{\mathrm{hyb}}$ 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 $(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c^{(n)})$

By observation we have,

$$\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}_c) = 1] = \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}^{(N)}_{\mathrm{hyb}}) = 1], \tag{88}$$

$$\Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{P}_{\mathrm{orig}}, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}), \mathbf{I}'_c) = 1] = \Pr[\mathcal{D}_{\mathsf{MPP}}(\mathcal{S}^{(0)}_{\mathrm{hyb}}) = 1]. \tag{89}$$

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

1) Queries $\mathcal{D}_{\mathsf{MPP}}$ and obtains $2 \leq N, o_1, o_2, \ldots, o_N \leq q-1$.
2) Randomly chooses $k^* \xleftarrow{\$} [N], m_1 \xleftarrow{\$} [o_1], m_2 \xleftarrow{\$} [o_2], \ldots, m_{k^*} \xleftarrow{\$} [o_{k^*}]$.
3) Defines the following sets for all $j \in [N]$,

$$\mathcal{P}_{\mathrm{orig}}(I_j) = \{P_{j,1}, P_{j,2}, \ldots, P_{j,o_j}\},$$
$$\mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_j)) = \{Q_{j,1}, Q_{j,2}, \ldots, Q_{j,o_j}\}.$$

4) Chooses $x_1, y_1, x_2, y_2, \ldots, x_{k^*-1}, y_{k^*-1} \xleftarrow{\$} \mathbb{Z}_q$. For all $j \in [k^*-1]$, sets,

$$P_{j,m_j} = G^{x_j}, \quad Q_{j,m_j} = G^{y_j}, \quad I_j = G^{x_j y_j}.$$

Also sets,

$$P_{k^*,m_j} = X, \quad Q_{k^*,m_j} = Y, \quad I_{k^*} = Z.$$

5) Sets $I_{k^*+1}, I_{k^*+2}, \ldots, I_N$ as uniform elements from $\mathbb{G}$. Populates other elements of $(\mathcal{P}_{\mathrm{orig}}(I_j), \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_j)))$ with uniform elements from $\mathbb{G}$ for all $j \in [N]$.
6) Defines the following sets,

$$\mathbf{I}_{\mathrm{DDH}} = \{I_1, I_2, \ldots, I_N\}$$
$$\mathcal{P}_{\mathrm{orig,DDH}} = \{\mathcal{P}_{\mathrm{orig}}(I_1), \mathcal{P}_{\mathrm{orig}}(I_2), \ldots, \mathcal{P}_{\mathrm{orig}}(I_N)\}$$
$$\mathbf{H}_p(\mathcal{P}_{\mathrm{orig,DDH}}) = \{\mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_1)), \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_2)), \ldots, \mathbf{H}_p(\mathcal{P}_{\mathrm{orig}}(I_N))\}.$$

[§§§]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.

7) Sends $(\mathscr{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathscr{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}})$ to $\mathscr{D}_{\mathsf{MPP}}$ and receives b' as output. Sends b' as its response to the challenger $\mathcal{C}$.

Now we have,

$$\begin{aligned}
&\Pr\left[\mathscr{D}_{\mathsf{DDH}}(X,Y,Z)=1 \,\middle|\, b=0\right] = \sum_{l=1}^{N} \Pr[k^*=l] \\
&\Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathscr{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathscr{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}})=1 \,\middle|\, b=0 \wedge k^*=l\right], \\
&\overset{(1)}{=} \sum_{l=1}^{N} \frac{1}{N} \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l-1)})=1\right], \\
&\overset{(2)}{=} \sum_{l=0}^{N-1} \frac{1}{N} \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)})=1\right].
\end{aligned} \tag{90}$$

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,

$$\begin{aligned}
&\Pr\left[\mathscr{D}_{\mathsf{DDH}}(X,Y,Z)=1 \,\middle|\, b=1\right] = \sum_{l=1}^{N} \Pr[k^*=l] \\
&\Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathscr{P}_{\text{orig,DDH}}, \mathbf{H}_p(\mathscr{P}_{\text{orig,DDH}}), \mathbf{I}_{\text{DDH}})=1 \,\middle|\, b=1 \wedge k^*=l\right], \\
&= \sum_{l=1}^{N} \frac{1}{N} \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)})=1\right].
\end{aligned} \tag{91}$$

We have,

$$\begin{aligned}
&\Big| \Pr\left[\mathscr{D}_{\mathsf{DDH}}(X,Y,Z)=1 \,\middle|\, b=0\right] - \\
&\Pr\left[\mathscr{D}_{\mathsf{DDH}}(X,Y,Z)=1 \,\middle|\, b=1\right] \Big| \\
&\overset{(3)}{=} \left| \frac{1}{N} \left(\sum_{l=1}^{N-1} \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)})=1\right] - \sum_{l=1}^{N} \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(l)})=1\right] \right) \right|, \\
&\overset{(4)}{=} \frac{1}{N} \left| \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(0)})=1\right] - \Pr\left[\mathscr{D}_{\mathsf{MPP}}(\mathcal{S}_{\text{hyb}}^{(N)})=1\right] \right|, \\
&\overset{(5)}{=} \frac{1}{N} \Big| \Pr[\mathscr{D}_{\mathsf{MPP}}(\mathscr{P}_{\text{orig}}, \mathbf{H}_p(\mathscr{P}_{\text{orig}}), \mathbf{I}'_c)=1] \\
&- \Pr[\mathscr{D}_{\mathsf{MPP}}(\mathscr{P}_{\text{orig}}, \mathbf{H}_p(\mathscr{P}_{\text{orig}}), \mathbf{I}_c)=1] \Big|, \\
&\overset{(6)}{\geq} \frac{1}{N} p(\lambda), \\
&= p_1(\lambda) \text{ (say)}.
\end{aligned} \tag{92}$$

Here the equality (3) comes from equations (90) and (91), the equality (4) comes from cancellations, the equality (5) comes from equations (89) and (88), and the inequality (6) comes from the assumption given in the inequality (87). However, the inequality (92) is a contradiction under the DDH assumption. So the inequality (87) cannot be true for any polynomial $p(\lambda)$. Hence there exists a negligible function $\mathsf{negl}'(\lambda)$ such that the inequality (83) holds. ■

Appendix G
Multiple Transactions Spending from Sources of the MProve+ Proofs

In Example 2, we have considered *txn* where a single source address P is being spent. Consider the situation when *Ex* spends another source address P' in some other transaction *txn'*. Let the key image for P' be I'. When *txn* appears in the blockchain, the originating set of I for the adversary $\mathcal{A}(\mathscr{P}_{\text{orig}}(I), \mathbf{P}_{\text{other}}, \mathbf{R}(txn))$ becomes the set $\left(\mathscr{P}_{\text{orig}}(I) \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn)$. So some the edges in the graph (U,V,E) connecting P and I are removed. However, the originating sets for other key images remain as they were before. To see this, observe Definition 1. Only some edges of the matchings of $\mathcal{M}$ are removed when *txn* appears. All these removed edges have I and no other key image as a vertex. So by Definition 1, the originating sets of key images other than I remain the same before and after *txn* appears in the blockchain. For example, the originating set for I' remains $\mathscr{P}_{\text{orig}}(I')$ after *txn* appears in the blockchain. When *txn'* appears in the blockchain, for the adversary $\mathcal{A}(\mathscr{P}_{\text{orig}}(I'), \mathbf{P}_{\text{other}}, \mathbf{R}(txn'))$, the originating set for I' is $\left(\mathscr{P}_{\text{orig}}(I') \setminus \mathbf{P}_{\text{other}}\right) \bigcap \mathbf{R}(txn')$. Similar analysis follows when other source addresses apart from P and P' are spent.

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 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.

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

[8] 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.

[9] 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.

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

[11] 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.

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

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

[14] 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.

[15] 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.

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

[17] A. Kumar, C. Fischer, S. Tople, and P. Saxena, "A traceability analysis of monero's blockchain," in *Computer Security – ESORICS 2017*. Springer International Publishing, 2017, pp. 153–173.

[18] 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.

[19] 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.

[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, 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] A. Jivanyan, "Lelantus: Towards confidentiality and anonymity of blockchain transactions from standard assumptions," Cryptology ePrint Archive, Report 2019/373, 2019, https://eprint.iacr.org/2019/373.

[26] J. Yu, M. H. A. Au, and P. Esteves-Verissimo, "Re-thinking untraceability in the cryptonote-style blockchain," in *2019 IEEE 32nd Computer Security Foundations Symposium (CSF)*, 2019, pp. 94–9413.

[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, "Short paper: An empirical analysis of monero cross-chain traceability," in *Financial Cryptography and Data Security*, I. Goldberg and T. Moore, Eds. Cham: Springer International Publishing, 2019, pp. 150–157.

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

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

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

[33] 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.

[34] D. J. Bernstein, "Pippenger's exponentiation algorithm," 2002.

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

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

[37] 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

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