Shorter, Privacy-Preserving Proof of Reserves Protocols for Cryptocurrency Exchanges

A Seminar Report
Submitted in partial fulfillment of
the requirements for the degree of
Dual Degree (B.Tech & M.Tech) in Electrical Engineering with Specialization in Communication & Signal Processing
by

Suyash Bagad
(Roll No. 15D070007)

Supervisor:
Prof. Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai 400076 (India)

20 June 2020

Dedicated to ...

Acceptance Certificate

Department of Electrical Engineering
Indian Institute of Technology, Bombay

The dissertation entitled "Shorter, Privacy-Preserving Proof of Reserves Protocols for Cryptocurrency Exchanges" submitted by Suyash Bagad (Roll No. 15D070007) may be accepted for being evaluated.

Date: 20 June 2020

Prof. Saravanan Vijayakumaran

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I declare that I have properly and accurately acknowledged all sources used in the production of this report. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Suyash Bagad

Date: 20 June 2020 (Roll No. 15D070007)

Abstract

The rise of cryptocurrencies began with the inception of Bitcoin in 2009. Since then, several cryptocurrencies with better privacy and security guarantees are being developed. Cryptocurrencies gained popularity among general masses with the establishment of cryptocurrency exchanges. Also known as digital currency exchanges or crypto exchanges, they are essentially businesses that allow customers to trade cryptocurrencies or digital currencies for other assets including conventional fiat money or different digital currencies. From a customer point of view, exchanges not only made owning cryptocurrencies possible to non-miners but also provided them with fast trading platforms for transactions within cryptocurrencies and fiat. Customers were also provided with custodial wallets freeing them from the hassle of storing and remembering private keys. In the early days of cryptocurrency, crypto exchanges were very few and less-known, but not too long ago their number increased dramatically and they became an integral part of the cryptoeconomic ecosystem. They were responsible for the boost in the transaction volumes of the vast majority of the cryptocurrency sales and liquidity.

The downside of such cryptocurrency exchanges is that they are required to store sensitive information of customers like the private keys and account balances. If in case an exchange is hacked, it might result in loss of customer-owned cryptocurrency assets. There have been many high-profile hacks over the years, many of which went unnoticed for some time [1]. Although having a fool-proof method to avoid such hacks might be a difficult task, proof of reserves is one way to uphold the trust of customers. A proof of reserves is the guarantee by the exchange that it owns reserves at least as much as its total liabilities towards customers. In this way, even after cases of hacking, the exchange could repay its liabilities to the customers.

The simplest way to publish a proof of reserves for an exchange is to reveal all the addresses or account details it owns so that the customers are convinced about the assets owned by the exchange. Another way could be to send all the reserves it owns from all its addresses to a single addresses it owns. If amounts involved in

a transaction are public as in the case of Bitcoin, such a self-transaction would be a proof of the exchange's reserves. For example, in 2011, Mt. Gox cryptocurrency exchange transferred 424,242 bitcoins from its wallets to a previously revealed Bitcoin address [2]. However, such proofs of reserves do not preserve the privacy of the exchanges. Information of an exchange's addresses or accounts and the total assets it owns are crucial for aspects of its business. Exchanges naturally would not be in a position to compromise such critical information as a part of proofs of reserves. The main challenge in design of proofs of reserves is to preserve privacy and confidentiality of exchanges but at the same time convince customers about an exchange's actual asset ownership. Regaining *trust* of the customers without compromising exchanges' *privacy* is the primary motivation behind the design of better proofs of reserves. Advanced cryptographic techniques make it possible to design proofs of reserves which reveal *nothing* beyond an assertation of the form:

Exchange X owns ? amount of the cryptocurrency Y.

Note that here we do not intend to reveal even the total amount. A publicly verifiable proof backing up such a claim is a cryptographic tool known as a *Non-Interactive Zero-Knowledge* proof.

In this work, we study the existing proof of reserves protocols for privacy-centric cryptocurrencies Grin, Beam and Monero. The existing proof of reserves protocols possess some shortcomings which becomes a hurdle in their practical deployment. With an aim to alleviate limitations of previously designed proofs of reserves, we design novel proofs of reserves for crypto exchanges supporting the above cryptocurrencies. Our protocols are shorter and privacy enhancing in comparison to the existing state-of-the-art proofs of reserves. Previous state-of-the-art proofs of reserves protocols provided some privacy to exchanges by hiding the exchange-owned addresses (or outputs) in a larger anonymity set. The proof sizes for these protocols scaled linearly with the anonymity set size. Since the level of privacy in a proof of reserves directly depends on how large the size of the anonymity set size is as compared to the number of exchange-owned addresses, larger anonymity sets imply stronger privacy. However, as the previous protocols proof sizes grew linearly as the anonymity set grows, it bought practical limitations (with regards to that of proof storage and broadcast) on what level of privacy could be attained. With an aim to improve scalability of the previously design proof of reserves protocols, we design a strategy based on Bulletproofs technique [3] to design proofs of reserves scaling logarithmically in the anonymity set. This brings flexibility in the choice of the

size of anonymity set and therefore enhances the attainable level of privacy. Along with this, we also use the concept of *key images* to ensure that different exchanges cannot share their addresses. The key images subsequently also helps in detecting double-spending of addresses by an exchange. Going a step further, we also devise a cryptographic technique to enforce different exchanges to publish proofs of reserves corresponding to a same blockchain state, which is absent in previous work. This ensures that exchanges can publish proofs of reserves only at particular time instances (for example, after each block is mined), preventing them from cheating customers with ambiguous choices of anonymity sets. We also implement the protocols we design as well as the previous state-of-the-art protocols for comparison and show feasibility in practical deployment of our protocols. We believe that our work on proof of reserves could be well adopted by crypto exchanges and benefit several of their customers. Furthermore, our work also opens up avenues of exploration of establishing trust without compromising privacy or anonymity in a more general setting. We are hopeful that this work acts as a small but crucial contribution towards the goal of establishing a *trustless, decentralized economy.*

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of cryptocurrencies have openend up unending possibilities of how minimal or no trust based systems could be established owing to decentralization. The concept of blockchain was introduced with the founding of Bitcoin by Satoshi Nakamoto [4]. Satoshi Nakamoto proposed and implemented idea of a consensus based, trustless, truly peer-to-peer system for financial transactions. Notwithstanding an intrumental step towards establishing a decentralized and a trustless system, Bitcoin has several practical limitations with regards to privacy, security and scalability [5]. This led to the development of more privacy and anonymity focussed cryptocurrencies like Monero [6] and Zcash [7]. Grin [8] and Beam [9] are two relatively new projects which are backed by the MimbleWimble protocol [10] and claim to promise scalability, anonymity and fungibility all at once. The rise in privacy-centric cryptocurrencies further led to growth in popularity of cryptocurrencies not only among investors but also common people.

1.1 What is a Blockchain?

The concept of a Blockchain originates from the idea of having a decentralized, publicly visible and trust-free ledger. The term *blockchain* was first coined by Satoshi Nakamoto, the creator of Bitcoin [4]. In literal terms, a blockchain implies that blocks containing financial transactions would be added to a publicly verifiable ledger in a timely manner, forming a *chain* of *blocks*. We expand on the need and the basic idea of a decentralized ledger in the following subsection.

1.1.1 Decentralized Ledger

A ledger is a principal book of records which keeps a track of all transactions measured in terms of a monetary unit. Bitcoin is a decentralized, public ledger, decentralized because there is no trusted third party controlling the ledger and public because anyone with bitcoin can participate in the network, receive and send bitcoins, and even hold a copy of this ledger (essentially, the history of all transactions) if they want to. In that sense, the ledger is "non-trusted" and transparent to public. As opposed to such a decentralized ledger setup, traditional banks use centralized ledger system where each bank has it's own ledger visible only to the concerned user.

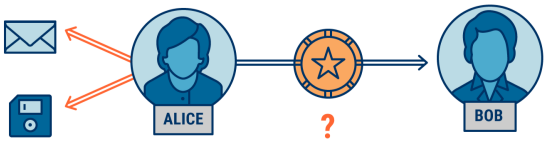
In reference to Figure* 1.1a, in a physical transaction, the Alice hands Bob a physical arcade coin. Bob now has one coin, and Alice has zero. The transaction is complete. If the same transaction is to be performed digitally (Figure 1.1b), Alice would send a string of some bits (corresponding to the desired amount) to Bob. Now, since it is just a string of bits, what is the guarantee that Alice would not reproduce that same string of bits in her mail or hard-disk? If this happens, it would mean that although Alice sent Bob the amount, Alice still possesses the same amount. This clearly is a violation. To tackle this, we must have a ledger or a record of all transactions between Alice and Bob. When Alice gives Bob the digital token, the ledger records the transaction (Figure 1.1c). Bob has the token, and Alice does not. Let us call the one who maintains this ledger as Dave. This setup assumes that Dave is a trusted middleman for maintaining ledger. Now, what if Alice bribes Dave to erase her transaction? What if Dave decides to charge a fee that neither Alice or Bob want to pay? What happens when Alice and Bob cannot trust the third party? This results in a decentralized and public ledger system (Figure 1.1d). When a lot of people have a copy of the same ledger, it becomes very difficult to tamper the system and cheat. This works because everyone is holding a copy of the same digital ledger and more the number of trusted people holding this ledger, the stronger it becomes.

Blockchain technology offers a way for untrusted parties to reach a consensus on a common digital history. The World Bank defines Blockchain as follows [11]:

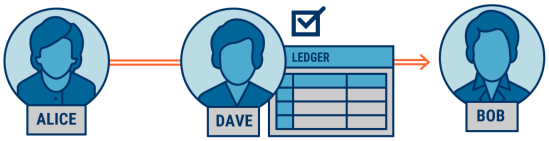
Definition 1.1.1 (Blockchain) *A 'blockchain' is a particular type of data structure used in some distributed ledgers which stores and transmits data in packages called "blocks" that are connected to each other in a digital 'chain'.*

*Credits: `https://www.cbinsights.com/research/what-is-blockchain-technology/`

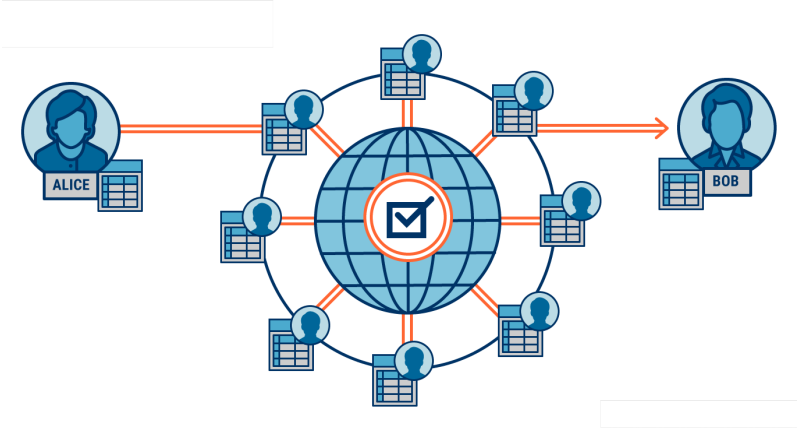
(a) A physical transaction



(b) A digital transaction



(c) A digital transaction with ledger



(d) A decentralized ledger

Figure 1.1: Understanding decentralized ledger

Blockchains employ cryptographic algorithms to record and synchronize data across a network in an unchangeable manner. The *state* of the Blockchain is the current status of the ledger visible to public. In case of Bitcoin, any independent observer can verify the state of the blockchain as well as the validity of all the transactions on the ledger. This is a *serious* limitation for Bitcoin and could possibly prohibit many use cases. As an example, if employees of a company were to receive their salaries in bitcoin, would they agree if their salaries were published on the

public blockchain? This naturally demands for a transaction system which preserves *anonymity* and *confidentiality*.

1.2 Notion of Privacy on a Blockchain

The very concept of having all the information about transactions (predominantly financial) public via a distributed ledger or a blockchain demands for privacy and anonymity of the users. Before we talk about privacy on the blockchain, it is necessary to understand what we do mean by terms like *privacy* and *anonymity*. Modern day cryptography is based on the following primary functions [12]:

(i) *Privacy/confidentiality*: To ensure that no one can read or access the message except the intended receiver

(ii) *Authentication*: Proving one's identity

(iii) *Integrity*: Ensuring that the message intended to be received by the receiver is not altered in the path

(iv) *Non-repudiation*: A protocol for checking if a message was actually generated by the sender

(v) *Key exchange*: The protocol which determines how key(s) are shared between the sender and the receiver

All of the above functions form an necessary part of a cryptographic system. A well-designed system ensures all of the above functions are taken care of considering the computational bounds for carrying out any protocol. In a confidential transaction, it is desirable to have *confidentiality* and *anonymity*. This means that in a valid confidential transaction, the identity of the sender and receiver must be confidential and the amounts transferred must also be hidden. The idea of having such private transactions in digital currencies could be traced back to David Chaum's work on blind signatures [13]. A similar concept of privacy is required in the blockchain framework. The digital assets owned by users must not be publicly visible. There should be a mechanism to unlink the identity of a user with his or her digital identity, i.e. public key address. The interaction between the sender and the receiver in a transaction must not reveal anything to them other than the amounts being transferred. A user must not be able to reuse his or her digital assets. Making the transactions digital and public at the same time brings about several challenges

in ensuring that the above requirements are being satisfied. Active research is still being pursued in the direction of not only solving such practical problems but to do them efficiently and in a scalable manner.

1.3 Cryptocurrency Exchanges & Security

1.4 Proof of Solvency

1.4.1 Proof of Reserves

1.4.2 Proof of Liabilities

Chapter 2

Cryptographic Preliminaries

Elliptic-curve cryptography (ECC) is essentially a public-key cryptography system, design of which is based on the algebraic structure of elliptic curves over finite fields [14]. ECC enables significant reduction in memory usage as the public-key length in ECC framework is much smaller than other public-key cryptography schemes like RSA for the same security level. It is widely used for many applications like encryption, digital signatures, pseudo-random generators and other tasks.

All cryptocurrencies are built on the Elliptic-curve cryptography framework. The security of such systems depend on the security guarantees provided by the ECC framework. We will see a couple of cryptographic assumptions which promise us the security guarantees for cryptocurrency systems. Thus, before delving into the details of proof of reserves protocols, it is worthwhile spending some time learning about the background math of cryptocurrencies. Note that we assume familiarity of the reader with basic concepts of group theory and modular arithmetic. A short but sufficient primer on both of these topics in present in [15].

2.1 Notation

Let $\mathcal{G} = \{\mathbb{G}, q, g\}$ be the description of a cyclic group $\mathbb{G}$ of prime order q with generator g of $\mathbb{G}$. 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. Let $\mathbb{G}^n$ and $\mathbb{Z}_q^n$ be the n-ary Cartesian powers of sets $\mathbb{G}$ and $\mathbb{Z}_q$ respectively. Group elements which are Pedersen commitments are denoted by uppercase letters and randomly chosen group elements are denoted by lowercase letters. Bold font denotes vectors. Inner product of two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle \coloneqq \sum_{i=1}^{n} a_i \cdot b_i$ where $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n)$. Further, Hadamard and Kronecker products are defined respectively

as, $\mathbf{a} \circ \mathbf{b} \coloneqq (a_1 \cdot b_1, \ldots, a_n \cdot b_n) \in \mathbb{Z}_q^n$, $\mathbf{a} \otimes \mathbf{c} \coloneqq (a_1\mathbf{c}, \ldots, a_n\mathbf{c}) \in \mathbb{Z}_q^{nm}$ where $\mathbf{c} \in \mathbb{Z}_q^m$. 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 $\boldsymbol{u}^n \coloneqq (1, u, u^2, \ldots, u^{n-1})$. To represent the exponentiation of all components of a vector $\mathbf{a}$ by the same scalar $k \in \mathbb{Z}_q$, we use $\mathbf{a}^{\circ k}$ to mean $(a_1^k, a_2^k, \ldots, a_n^k)$. If an element a is chosen uniformly from a set A, such a choice is denoted by $a \xleftarrow{\$} A$. We denote the relation *Relation* using the specified input and witness as $\{(Public\ Input;\ Witness) : Relation\}$. We refer to $\mathcal{A}$ as a PPT adversary which is a probabilistic Turing Machine that runs in polynomial time in the security parameter λ. An *interactive proof* for the decision problem π is described as follows:

1. There are two participants, a prover $\mathcal{P}$ and a verifier $\mathcal{V}$.
2. The proof consists of a specified number of rounds.
3. In the beginning, both participants get the same input.
4. In each round, the verifier challenges the prover, and the prover responds to the challenge.
5. Both the verifier and the prover can perform some private computation.
6. At the end, the verifier states whether he was convinced or not.

2.2 Basics of Elliptic Curves

Let $a, b \in \mathbb{R}$ such that $4a^3 + 27b^2 \neq 0$. Let E be the set of solutions $(x, y) \in \mathbb{R}$ to the equation

$$y^2 = x^3 + ax + b. \tag{2.1}$$

An elliptic curve over $\mathbb{R}$ is given by the set $E \cup \{\mathcal{O}\}$ where $\mathcal{O}$ is known as the *point at infinity*. An example of an elliptic curve over $\mathbb{R}$ is given in Figure 2.1. Note that the condition $4a^3 + 27b^2 \neq 0$ ensures that the curve does not have repeating roots. This condition is necessary in the discussion of elliptic curve groups.

2.2.1 Point Addition in Elliptic Curves

The set $E \cup \{\mathcal{O}\}$ needs to be an algebraic group for us to be able to define operations on it. Thus, we define a group operation over $E \cup \{\mathcal{O}\}$ known as *point addition*. Suppose we have two points $P = (x_1, y_1), Q = (x_2, y_2) \in \mathbb{R} \times \mathbb{R}$ such that $x_1 \neq x_2$. We show them by blue and yellow coloured points in Figure 2.2(a). We

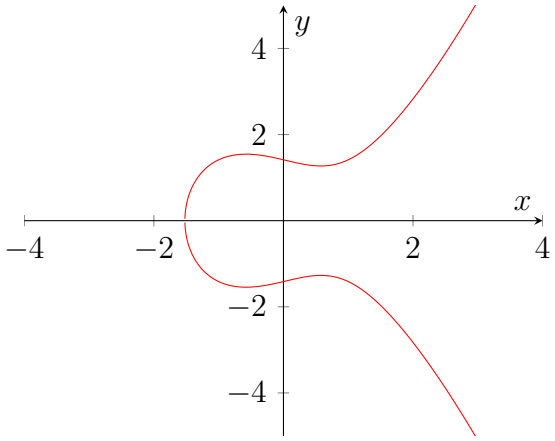


Figure 2.1: An elliptic curve given by the equation $y^2 = x^3 - x + 2$ over $\mathbb{R}$

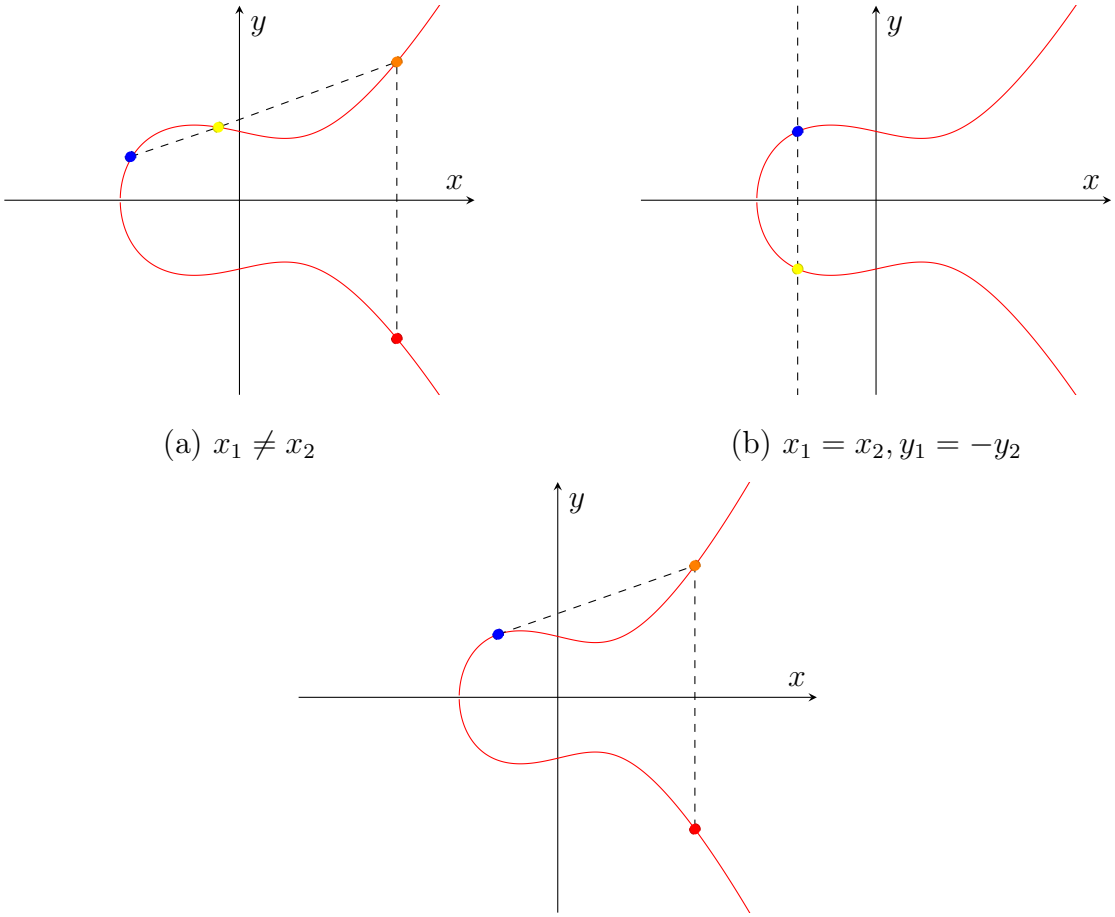
draw a line passing through P and Q. As the degree of an elliptic curve equation is 3, any non-tangent line must intersect the curve in 3 distinct points. Suppose the line passing through P and Q intersects the curve at point $R' = (x_3, -y_3) \in \mathbb{R} \times \mathbb{R}$, shown in orange colour. We define the result of point addition of points P and Q to be the point R which is the mirror reflection of R'. Therefore, we have $R = (x_3, y_3)$ shown in red colour. Simple calculation shows that given $P = (x_1, y_1), Q = (x_2, y_2)$ and $x_1 \neq x_2$, point addition gives us $P + Q = (x_3, y_3)$ such that

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2, \;\; y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1. \tag{2.2}$$

Now if P and Q are such that $x_1 = x_2$ but $y_1 = -y_2$, the line passing through P and Q is vertical and possibly intersects the curve at infinity. In this case, we define the point addition operation as $P + Q = \mathcal{O}$. Therefore, the special point $\mathcal{O}$ acts as an identity point in the group $E \cup \{\mathcal{O}\}$. Further, if we have $P = Q$, i.e. $x_1 = x_2, y_1 = y_2 \neq 0$, we draw tangent to the curve at point P. As the degree of the curve is 3, any tangent will intersect the curve at only and only one point, say point R'. The result R in this case is again the mirror reflection of point R' about the x-axis. The addition of a point to itself is known as *point doubling*. The explicit formula for point doubling of point $P = (x_1, y_1)$ can be written as $P + P = 2P = (x_2, y_2)$ such that

$$x_2 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1, \;\; y_2 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_2) - y_1. \tag{2.3}$$

The point addition operation is closed in the set $E \cup \{\mathcal{O}\}$ by construction. Further, $\mathcal{O}$ acts as an identity element in the set $E \cup \{\mathcal{O}\}$. For each point $P \in E$, we can find its *inverse* $Q \in E$ as its reflection about the x-axis as for such cases, we have $P + Q = \mathcal{O}$. Therefore, the set $E \cup \{\mathcal{O}\}$ is a group under point addition.



(a) $x_1 \neq x_2$

(b) $x_1 = x_2, y_1 = -y_2$

(c) $x_1 = x_2, y_1 = y_2 \neq 0$

Figure 2.2: Point addition in elliptic curves over $\mathbb{R}$

In practice, we use elliptic curves over finite fields instead of real numbers. The group operations are now defined over an underlying finite field F. The division by a non-zero field element $x \in F$ is interpreted as multiplication by its multiplicative inverse x^{-1}. Similarly, subtraction of a field element $x \in F$ is interpreted as addition by its additive inverse $-x$. This is the basis of the elliptic curve cryptography used in modern day systems except that of For example, Bitcoin and Grin cryptocurrency systems use the prime-ordered elliptic curve `secp256k1` [16].

Note that from hereon we use multiplicative notation to denote group operation on elliptic curves $\mathbb{G}$ on finite fields $\mathbb{F}_q$ for a large prime q. Lastly, addition of a point $P \in \mathbb{G}$ for k times is shown as scalar multiplication in additive notation and exponentiation in multiplicative respectively.

$$\underbrace{P + P + \cdots + P}_{k \text{ times}} = kP \in \mathbb{G}.$$

Similarly, in multiplicative notation, such an operation is called as exponentiation and is shown below

$$\underbrace{P \cdot P \cdot \ldots \cdot P}_{k \text{ times}} = P^k \in \mathbb{G}.$$

2.3 Cryptographic Assumptions

Each practical cryptographic systems is built on certain hardness assumptions. For example, the widely popular RSA digital signature algorithm was based on the assumption that it computationally hard to factorize big primes [17]. Elliptic curve cryptography is similarly based on the assumption that the *Discrete Log Problem* is difficult to be solved by a computationally bounded adversary.

Definition 2.3.1 (Discrete Log Relation) *For all PPT adversaries $\mathcal{A}$ and for all $n \geq 2$, $\exists$ a negligible function $\mu(\lambda)$ s.t*

$$\Pr\left[\begin{array}{ll} \mathbb{G} = Setup(1^\lambda), g_1, \ldots, g_n \leftarrow \mathbb{G} \ ; & \\ a_1, \ldots, a_n \in \mathbb{Z}_p \leftarrow \mathcal{A}(\mathbb{G}, g_1, \ldots, g_n) & : \exists a_i \neq 0 \wedge \prod_{i=1}^n g_i^{a_i} = 1 \end{array}\right] \leq \mu(\lambda)$$

We say $\prod_{i=1}^n g_i^{a_i} = 1$ is a non trivial discrete log relation between $g_1, \ldots, g_n$. If the Discrete Log Relation assumption stands, it implies that no PPT adversary can find a non-trivial relation between randomly chosen group elements. This is known as the Discrete Log Problem.

We use additional cryptographic assumptions such as Decisional Diffie-Hellman and its variants as described in [18].

2.4 Cryptographic Commitments

Cryptographic commitments are an important preliminary widely used to anonymise data like amounts. We also briefly discuss some key properties of commitments of our interest.

Definition 2.4.1 (Commitments) *A non-interactive commitment consists of two PPT algorithms* (Setup, Com)*. For a message* $x \in \mathbf{M}_{pp}$ *(message space), the algorithm proceeds as follows:*

1. *public parameters* $pp \leftarrow Setup(1^\lambda)$ *for security paramter* λ
2. $\mathrm{Com}_{pp} : \mathbf{M}_{pp} \times \mathbf{R}_{pp} \to \mathbf{C}_{pp}$*, where* $\mathbf{R}_{pp}$ *is randomness space*
3. $r \leftarrow \mathbf{R}_{pp}$ *and compute* $\mathbf{com} = \mathrm{Com}_{pp}(x; r)$

Definition 2.4.2 (Homomorphic Commitments) *A homomorphic commitment is a non-interactive commitment such that* $\mathbf{M}_{pp}$*,* $\mathbf{R}_{pp}$*,* $\mathbf{C}_{pp}$ *are all abelian groups, and* $\forall\ x_1, x_2 \in \mathbf{M}_{pp}$*,* $r_1, r_2 \in \mathbf{R}_{pp}$*, we have*

$$\mathrm{Com}(x_1; r_1) + \mathrm{Com}(x_2; r_2) = \mathrm{Com}(x_1 + x_2; r_1 + r_2)$$

Definition 2.4.3 (Hiding Commitment) *A commitment scheme is said to be hiding if for all PPT adversaries* $\mathcal{A}$*,* $\exists \mu(\lambda)$*, a negligible function such that,*

$$\left| \Pr\left[\ b\text{'}{=}b \ \middle| \begin{array}{l} pp \leftarrow \mathrm{Setup}(1^\lambda); \\ (x_0, x_1) \in \mathbf{M}_{pp}^2 \leftarrow \mathcal{A}(pp),\ b \leftarrow \{0,1\},\ r \leftarrow \mathbf{R}_{pp}, \\ \mathbf{com} = Com(x_b; r),\ b' \leftarrow \mathcal{A}(pp,\ \mathbf{com}) \end{array} \right] - \frac{1}{2} \right| \leq \mu(\lambda)$$

where the probability is over $b', r, Setup$ *and* $\mathcal{A}$*. For perfectly hiding schemes,* $\mu(\lambda) = 0$*.*

In simple words, a commitment scheme is *hiding* if it is impossible for a computationally bounded adversary to find what the message is hidden in a commitment or what randomness was used in computing the commitment.

Definition 2.4.4 (Binding Commitment) *A commitment scheme is said to be binding if for all PPT adversaries* $\mathcal{A}$*,* $\exists \mu(\lambda)$*, a negligible function such that,*

$$\Pr\left[\ \mathrm{Com}(x_0; r_0) = \mathrm{Com}(x_1; r_1) \wedge x_0 \neq x_1 \ \middle| \begin{array}{c} pp \leftarrow \mathrm{Setup}(1^\lambda), \\ x_0, x_1, r_0, r_1 \leftarrow \mathcal{A}(pp) \end{array} \right] \leq \mu(\lambda)$$

where the probability is over Setup and $\mathcal{A}$*. Again, if* $\mu(\lambda) = 0$ *then we say the scheme is perfectly binding.*

A commitment scheme is known as *binding* if it is impossible for a computationally bounded adversary to change the message a commitment commits to once it has published the commitment to the original message.

Definition 2.4.5 (Pedersen Commitment) $\mathbf{M}_{pp}, \mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order* p.

1. Setup*:* $g, h \leftarrow \mathbb{G}$
2. $\mathrm{Com}(x; r) = (g^x h^r)$

Definition 2.4.6 (Pedersen Vector Commitment) $\mathbf{M}_{pp} = \mathbb{Z}_p^n$, $\mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order* p.

1. Setup*:* $\boldsymbol{g} = (g_1, \ldots, g_n), h \leftarrow \mathbb{G}$
2. $\mathrm{Com}(\boldsymbol{x} = (x_1, \ldots, x_n); r) = (h^r \boldsymbol{g}^{\boldsymbol{x}})$

The Pedersen vector commitment is *perfectly hiding* and *computationally binding* under the discrete logarithm assumption. This means that no matter how much computational power an adversary possesses, he cannot find the message hidden in a Pedersen commitment. Further, for a computationally bounded adversary, it is infeasible to find another opening to a Pedersen commitment once he has committed it to original message.

2.5 Zero-Knowledge Arguments of Knowledge

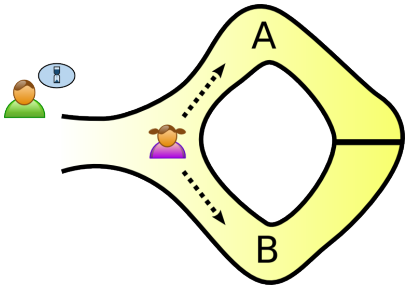
2.5.1 Zero-Knowledge Arguments

A protocol in which a prover convinces a verifier that a statement is true *without* revealing any information about why it holds is known as a Zero-knowledge argument. An argument is a proof only if the prover is computationally bounded and some computational hardness holds. Hereafter, we use the terms *proof* and *argument* interchangeably.

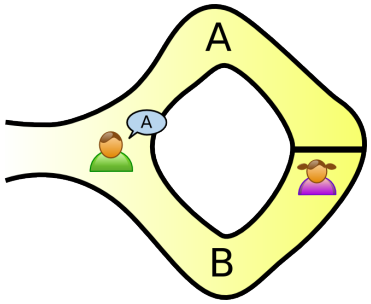
We illustrate the idea of zero-knowledge arguments of proof using the example of *Ali-Baba's secret cave* [19]. *

In the above example, Peggy knows the secret word used to open a mysterious door in a cave. The cave is shaped like a horse-hoe. The entrance is on one side and the magic door blocking the opposite side. Victor wants to know whether Peggy

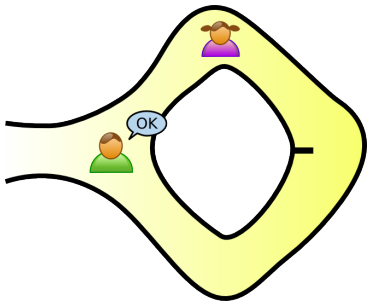
*Figure courtesy: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`.



(a) Peggy chooses a path uniformly from A, B without Victor knowing.



(b) Victor asks her to come out of the cave from path A.



(c) If Peggy had entered from path A, she returns trivially. Otherwise, she could open the door using the secret key and return from path A.

Figure 2.3: Example of a zero knowledge proof

knows the secret word; but Peggy, does not want to reveal her knowledge (the secret word) to Victor or to reveal the fact of her knowledge to anyone in the world.

Peggy and Victor run the protocol described in figure 2.3. Provided she really does know the magic word, and the path she enters and path Victor asks her to come from are same, then it's trivial for Peggy to succeed and Victor to believe that she actually knows the secret key. Further, if the chosen path by Peggy and asked by Victor doesn't match, even then she could open the door and return from a desired path. If they were to repeat this protocol many times, say 15 times in a row, her chance of successfully "guessing" all of Victor's requests would become exponentially small (about three in a lakh).

For zero-knowledge arguments presented in this report, we will consider arguments consisting of three interactive probabilistic polynomial time algorithms $(\text{Setup}, \mathcal{P}, \mathcal{V})$. These algorithms are described by:

1. Setup: $\sigma \leftarrow \text{Setup}(1^\lambda)$, σ is common reference string
2. $\mathcal{P}$: prover, $\mathcal{V}$: verifier
3. Transcript $tr \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle$
4. $\langle \mathcal{P}, \mathcal{V} \rangle = b$, $b = 0$ if the verifier rejects or $b = 1$ accepts

Further, we define the relation $\mathcal{R}$ and the CRS-dependent language as:

$$\mathcal{R} := \{(\sigma, u, w) \in \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* : \ w \text{ is a witness for } u \mid \sigma\}$$
$$\mathcal{L}_\sigma := \{x \mid \exists w \in (\sigma, u, w) \in \mathcal{R}\}$$

So, $\mathcal{L}_\sigma$ is essentially the set of statements x that have a witness w in the relation $\mathcal{R}$.

2.5.2 Defining Zero-Knowledge Arguments of Knowledge

To mathematically define the notion of zero-knowledge and zero-knowledge arguments, we will provide the necessary definitions below.

Definition 2.5.1 (Argument of Knowledge) *The triple $(Setup, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for relation $\mathcal{R}$ if it is perfectly complete and has computational witness-extended emulation.*

Definition 2.5.2 (Perfect completeness) *$(Setup, \mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

$$\Pr\left[\ (\sigma, u, w) \notin \mathcal{R} \text{ or } \langle \mathcal{P}(\sigma, u, w), \mathcal{V}(\sigma, u) \rangle = 1 \ \middle| \ \begin{array}{l} \sigma \leftarrow Setup(1^\lambda) \\ (u, w) \leftarrow \mathcal{A}(\sigma) \end{array} \right] = 1$$

Perfect completeness implies that if a statement is actually true, then an honest verifier is convinced with probability 1 about the truth of the statement by an honest prover.

Definition 2.5.3 (Computational Witness-Extended Emulation) *$(Setup, \mathcal{P}, \mathcal{V})$ has witness-extended emulation if for all deterministic polynomial time P^* there exists an expected polynomial time emulator $\mathcal{E}$ such that for all*

pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$ there exists a negligible function $\mu(\lambda)$ such that

$$\left| \Pr\left[\mathcal{A}_1(tr) = 1 \;\middle|\; \begin{array}{l} \sigma \leftarrow \mathit{Setup}(1^\lambda,) \\ (u,s) \leftarrow \mathcal{A}_2(\sigma), \\ tr \leftarrow \langle (\mathcal{P}^*(\sigma,u,s), V(u,s)) \rangle \end{array} \right] - \Pr\left[\begin{array}{l} \mathcal{A}_1(tr) = 1 \wedge \\ \\ (tr \text{ accepted} \implies (\sigma,u,w) \in \mathcal{R}) \end{array} \;\middle|\; \begin{array}{l} \sigma \leftarrow \mathit{Setup}(1^\lambda,) \\ (u,s) \leftarrow \mathcal{A}_2(\sigma), \\ (tr,w) \leftarrow \mathcal{E}^{\mathcal{O}}(\sigma,u) \end{array} \right] \right| \leq \mu(\lambda)$$

where the oracle is given by $\mathcal{O} = \langle (\mathcal{P}^*(\sigma,u,s), V(u,s)) \rangle$, and permits rewinding to a specific point and resuming with fresh randomness for the verifier from this point onwards. We can also define com- putational witness-extended emulation by restricting to non-uniform polynomial time adversaries $\mathcal{A}_1$ and $\mathcal{A}_2$.

Computational witness-extended emulation implies that when an adversary produces an argument to convince the verifier with some probability, then we have a corresponding emulator producing identically distributed argument with same probability, but also a witness.

Definition 2.5.4 (Public coin) *An argument of knowledge $(\mathit{Setup}, \mathcal{P}, \mathcal{V})$ is called public coin if all messages sent from the verifier to the prover are chosen uniformly at random and independent of the prover's messages, i.e., the challenges correspond to the verifier's randomness ρ.*

Definition 2.5.5 (Zero Knowledge Argument of Knowledge) *An argument of knowledge $(\mathit{Setup}, \mathcal{P}, \mathcal{V})$ is zero knowledge if it reveals no information about w apart from what could be deduced from the fact that $(\sigma, u, w) \in \mathcal{R}$.*

An argument of knowledge is zero knowledge if it does not leak information about w apart from what can be deduced from the fact that $(\sigma, u, w) \in \mathcal{R}$. More explicitly, we note that, a zero knowledge argument of knowledge ensures that no PPT adversary (or verifier) can ever recover w given it's relation with σ, u.

Definition 2.5.6 (Perfect Special Honest-Verifier Zero-Knowledge) *A public coin argument of knowledge $(\mathit{Setup}, \mathcal{P}, \mathcal{V})$ is a perfect special honest verifier zero knowledge (SHVZK) argument of knowledge for $\mathcal{R}$ if there exists a probabilistic polynomial time simulator $\mathcal{S}$ such that for all pairs of interactive adversaries $\mathcal{A}_1, \mathcal{A}_2$*

$$\Pr\left[\ (\sigma,u,w)\in\mathcal{R}\wedge\mathcal{A}_1(tr)=1 \;\middle|\; \begin{array}{l}\sigma\leftarrow \mathit{Setup}(1^\lambda,)\\ (u,w,\rho)\leftarrow\mathcal{A}_2(\sigma),\\ tr\leftarrow\langle(\mathcal{P}(\sigma,u,s),V(\sigma,u;\rho)\rangle\end{array}\right]$$

$$=\Pr\left[\ (\sigma,u,w)\in\mathcal{R}\wedge\mathcal{A}_1(tr)=1 \;\middle|\; \begin{array}{l}\sigma\leftarrow \mathit{Setup}(1^\lambda,)\\ (u,w,\rho)\leftarrow\mathcal{A}_2(\sigma),\\ tr\leftarrow\mathcal{S}(u,\rho)\end{array}\right]$$

PSHVZK AoK implies that even if an adversary chooses a distribution over statements and witnesses, it isn't able to distinguish between simulated transcript and honestly generated transcript for $u\in\mathcal{L}_\sigma$.

We will be using these definitions of Zero knowledge argument and its properties in the discussion further without redefining them, unless explicitly stated.

Chapter 3

RevelioBP - Shorter MimbleWimble Proof of Reserves Protocol

3.1 Introduction

A proof of reserves protocol is used by a cryptocurrency exchange to prove that it owns a certain amount of cryptocurrency. If privacy of the amount or outputs owned by the exchange is not an issue, then proving reserves involves a straightforward proof of the ability to spend the exchange-owned outputs (for example, see [20]). Non-private proof of reserves protocols are unlikely to be adopted by exchanges as they may reveal business strategy. Privacy-preserving proof of reserves protocols have been proposed for Bitcoin [21, 22], Monero [23], and MimbleWimble [24]. In fact, the protocols proposed by Decker *et al* [21] and Dagher *et al* [22] go one step further and give a privacy-preserving proof of solvency, i.e. they prove that the reserves owned by the exchange exceed its liabilities towards its customers. However, the work in [21] relies on a trusted hardware assumption. And the proof of liabilities protocol in [22] is secure only if every exchange customer checks the proof. In general, it seems that designing proof of reserves protocols is easier than designing proof of liabilities protocols as the former depend only the blockchain state while the latter depend on the exchange's private customer data.

Even without a robust proof of liabilities protocol, a privacy-preserving proof of reserves protocol based on homomorphic commitments is valuable. For example, the proof of reserves protocols in [22, 23, 24] generate a Pedersen commitment C_{res} to the amount of reserves. Exchanges can easily prove that C_{res} is a commitment to an amount which exceeds a base amount a_{base}. While the base amount may not be exactly equal to the total liabilities of the exchange, it can be based on the trade

volume data published by the exchange [25]. This technique will help early detection of exchange hacks and exit scams. For example, in Februrary 2019 the Canadian exchange QuadrigaCX claimed that it had lost access to wallets containing customer funds due to the death (in December 2018) of their CEO who had sole custody of the corresponding passwords and keys. But an official investigation found that the wallets had been empty since April 2018, several months before the CEO's death [26, 27]. This discrepancy would have been detected earlier if the exchange had been required to give perioidic proofs of reserves.

MimbleWimble is a design for a scalable cryptocurrency which was proposed in 2016 [28]. Beam and Grin are two implementations of the MimbleWimble protocol which are available on several exchanges [25]. Revelio [24] was the first proof of reserves protocol for MimbleWimble coins which provided some privacy to exchanges by hiding the exchange-owned outputs inside an anonymity set of outputs. As the anonymity set is revealed as part of the proof of reserves, a larger anonymity set results in better privacy for the exchange. Since the Revelio proof size scales linearly with the anonymity set, it becomes an impediment in scaling the anonymity set to the set of all unspent transaction outputs (UTXOs). To solve the scalability issue of Revelio, we designed RevelioBP leveraging the Bulletproofs [3] framework, resulting in the proof size being logarithmic in the anonymity set size.

Our Contribution. In this paper, we present RevelioBP, a proof of reserves protocol for MimbleWimble with proof sizes scaling *logarithmcally* in the size of the anonymity set and *linearly* in the size of the exchange-owned output set. This makes it feasible to choose the anonymity set to be the set of all UTXOs on the blockchain. To make quantitative comparisons, we have implemented RevelioBP in Rust. At the time of writing this paper, the number of UTXOs on the Grin blockchain is approximately 161,000 [29]. A Revelio proof of reserves for this anonymity set will have size 42 MB as against 0.27 MB using RevelioBP instead.* This reduction in proof size, however, comes at the cost of larger proof generation and verification times. If an exchange is willing to compromise on the size of the proof and is required to give frequent proofs of reserves, Revelio serves as a better choice. If proof sizes are critical for an exchange and it is willing to spend more time generating the proof, RevelioBP clearly outperforms Revelio. In conclusion, we quantitatively highlight the trade-off between proof size and performance in using Revelio and RevelioBP, both of which are based on the discrete log assumption.

*Under the assumption that the exchange owns 5% of all UTXOs.

3.2 Preliminaries

3.2.1 Notation

Let $\mathcal{G} = \{\mathbb{G}, q, g\}$ be the description of a cyclic group $\mathbb{G}$ of prime order q with generator g of $\mathbb{G}$. 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. Let $\mathbb{G}^n$ and $\mathbb{Z}_q^n$ be the n-ary Cartesian products of sets $\mathbb{G}$ and $\mathbb{Z}_q$ respectively.

Group elements which are Pedersen commitments are denoted by uppercase letters and randomly chosen group elements are denoted by lowercase letters.

Bold font denotes vectors. Inner product of two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^n$ is defined as $\langle \mathbf{a}, \mathbf{b} \rangle \coloneqq \sum_{i=1}^{n} a_i \cdot b_i$ where $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n)$. Further, Hadamard and Kronecker products are defined respectively as, $\mathbf{a} \circ \mathbf{b} \coloneqq (a_1 \cdot b_1, \ldots, a_n \cdot b_n) \in \mathbb{Z}_q^n$, $\mathbf{a} \otimes \mathbf{c} \coloneqq (a_1\mathbf{c}, \ldots, a_n\mathbf{c}) \in \mathbb{Z}_q^{nm}$ where $\mathbf{c} \in \mathbb{Z}_q^m$. 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 $\boldsymbol{u}^n \coloneqq (1, u, u^2, \ldots, u^{n-1})$. To represent the exponentiation of all components of a vector $\mathbf{a}$ by the same scalar $k \in \mathbb{Z}_q$, we use $\mathbf{a}^{\circ k}$ to mean $(a_1^k, a_2^k, \ldots, a_n^k)$. 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, let $[n]$ denote the set $\{1, 2, \ldots, n\}$.

3.2.2 Outputs in MimbleWimble

In MimbleWimble, coins are stored in outputs which consist of Pedersen commitments of the form $C = g^r h^a \in \mathbb{G}$ where $g, h \in \mathbb{G}$ and $r, a \in \mathbb{Z}_q$. Here a represents the amount of coins stored in the output and r is a blinding factor. Each commitment is accompanied by a range proof which proves that the amount a lies in the range $\{0, 1, 2, \ldots, 2^{64} - 1\}$.

The group elements g and h are assumed to have an unknown discrete logarithm relationship. For example, in Grin $\mathbb{G}$ is the secp256k1 elliptic curve group, g is the base point of the secp256k1 curve, and h is obtained by hashing g with the SHA256 hash function [30]. The unknown discrete logarithm relationship makes the commitment computationally binding, i.e. a polynomial-time adversary cannot find $r' \neq r$ and $a' \neq a$ such that $C = g^r h^a = g^{r'} h^{a'}$.

To spend an output having the commitment $C = g^r h^a$, knowledge of the blinding factor r is required [31]. As spending ability is equivalent to ownership, a proof of reserves protocol for MimbleWimble involves a proof of knowledge of blinding factors of several outputs.

3.2.3 From Omniring to RevelioBP

In Monero, source addresses in a transaction are obfuscated using ring signatures and the amounts are hidden in Pedersen commitments [32]. The current transaction structure in Monero, called *ring confidential transaction (RingCT)*, has proof sizes which scale linearly in the ring size. Omniring [18] is a recent proposal for RingCTs with proof sizes which scale logarithmically in the ring size. It relies on Bulletproofs [3] to achieve this size reduction. Given a ring $\mathcal{R} = (R_1, R_2, \ldots, R_n)$ of public keys where $R_i = h^{x_i}$ for $h \in \mathbb{G}, x_i \in \mathbb{Z}_q$, the Omniring construction enables a prover to prove knowledge of the private keys $x_{i_1}, x_{i_2}, \ldots, x_{i_m}$ corresponding to a subset $\mathcal{R}_\mathcal{S}$ of $\mathcal{R}$ without revealing $\mathcal{R}_\mathcal{S}$. For each public key R_j in this subset $\mathcal{R}_\mathcal{S}$, the prover also outputs a *tag* given by $tag_j = g^{x_j^{-1}}$ for $g \in \mathbb{G}$. This tag is used to detect double spending from a source address.

The design of RevelioBP is inspired by the Omniring construction. Given the set of UTXOs $\mathcal{C}_{\text{utxo}} = (C_1, C_2, \ldots, C_n)$ on the blockchain where $C_i = g^{r_i} h^{a_i}$ for some $r_i, a_i \in \mathbb{Z}_q$, the prover in RevelioBP proves knowledge of blinding factors r_i and amounts a_i for all C_i in a subset $\mathcal{C}_{\text{own}}$ of $\mathcal{C}_{\text{utxo}}$ without revealing $\mathcal{C}_{\text{own}}$. For each output $C_j \in \mathcal{C}_{\text{own}}$, the prover outputs a tag $tag_j = g_t^{r_j} h^{a_j}$ where $g_t \in \mathbb{G}$ is a randomly chosen group element. In RevelioBP, the tag has a dual role. Firstly, it is used to detect output sharing between exchanges. Secondly, the product of the tags is a Pedersen commitment to the total reserves of the exchange.

3.3 RevelioBP Proof of Reserves Protocol

To spend a MimbleWimble output having the commitment $C = g^r h^a$, knowledge of the blinding factor r is required [31]. Technically, the ability to spend an output also requires knowledge of the amount a. But the amount can be at most $2^{64} - 1$, and hence can be found by brute force search given C and r.

Let $\mathcal{C}^t_{\text{utxo}}$ be the set of UTXOs on a MimbleWimble blockchain after the block with height t has been mined. An exchange will own a subset $\mathcal{C}^t_{\text{own}} \subset \mathcal{C}^t_{\text{utxo}}$, where ownership implies knowledge of the blinding factor for each output $C \in \mathcal{C}^t_{\text{own}}$. Using the RevelioBP protocol, the exchange can construct a Pedersen commitment C_{res} to an amount which is equal to the sum of the amounts commited to by each of the outputs in $\mathcal{C}^t_{\text{own}}$. Given a Pedersen commitment C_{liab} to the total liabilities of the exchange, it can give a proof of solvency via a range proof which shows that the amount committed to in $C_{\text{res}} C_{\text{liab}}^{-1}$ is non-negative. If there is no suitable method to

construct $\mathcal{C}_{\text{liab}}$, then the exchange can reveal a base amount a_{base} and prove that $\mathcal{C}_{\text{res}} h^{-a_{\text{base}}}$ is a commitment to a non-negative amount.

While RevelioBP does not reveal $\mathcal{C}^t_{\text{own}}$, it does reveal its cardinality $s_t = |\mathcal{C}^t_{\text{own}}|$. We give a reasonable workaround for this issue in Section 3.3.1.

If the Decisional Diffie-Hellman (DDH) assumption holds in the group $\mathbb{G}$, the RevelioBP proof of reserves protocol satisfies the following properties:

- *Inflation resistance*: Using RevelioBP, a probabilistic polynomial time (PPT) exchange will not be able to generate a commitment to an amount which exceeds the reserves it actually owns.
- *Collusion detection*: Situations where two exchanges share an output while generating their respective RevelioBP proofs will be detected.
- *Output privacy*: A PPT adversary who observes RevelioBP proofs from an exchange cannot do any better than random guessing while identifying members of $\mathcal{C}^t_{\text{own}}$.

The security proofs are given in Section 3.5.

3.3.1 Proof Generation

The RevelioBP protocol requires one randomly chosen group element $g_t \in \mathbb{G}$ per block such that the discrete log relation between g_t and g, h is unknown. All the exchanges need to agree upon the procedure used to generate the sequence of g_ts. For example, g_t could be generated by hashing the contents of the block at height t. An exchange giving a RevelioBP proof of its reserves at the block with height t performs the following procedure:

1. From the UTXO set $\mathcal{C}^t_{\text{utxo}}$ at block t, the exchange constructs the vector $\mathbf{C} = (C_1, C_2, \ldots, C_n)$ where the C_is are all the UTXOs arranged in the order of their appearance on the blockchain. So $n = |\mathcal{C}^t_{\text{utxo}}|$. To keep the notation simple, we do not make the dependence of $\mathbf{C}$ and n on t explicit.
2. The exchange owns a subset $\mathcal{C}^t_{\text{own}} = \{C_{i_1}, C_{i_2}, \ldots, C_{i_s}\}$ of $\mathcal{C}^t_{\text{utxo}}$ where $1 \leq i_1 < \cdots < i_s \leq n$. For each $C_{i_j} \in \mathcal{C}^t_{\text{own}}$, the exchange knows r_j and a_j such that $C_{i_j} = g^{r_j} h^{a_j}$. Using this information, the exchange constructs the *tag vector* $\mathbf{I} = (I_1, I_2, \ldots, I_s)$ where $I_j = g_t^{r_j} h^{a_j}$. Note that I_j is a Pedersen commitment to the amount a_j with blinding factor r_j using bases g_t, h. So the only difference between C_{i_j} and I_j is that the base g in the former is replaced with g_t in the latter.

3. Let $\mathbf{a} = (a_1, a_2, \ldots, a_s)$ and $\mathbf{r} = (r_1, r_2, \ldots, r_s)$ be the amount and blinding factor vectors corresponding to the exchange-owned outputs. Let $\mathbf{e}_{i_j} \in \{0,1\}^n$ be the unit vector with a 1 in position i_j and 0s everywhere else. Let $\mathcal{E} \in \{0,1\}^{s \times n}$ be the matrix with $\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \ldots, \mathbf{e}_{i_s}$ as rows.

   The exchange publishes $(t, \mathbf{I})$ and generates a zero-knowledge argument of knowledge Π_{RevBP} of quantities $(\mathcal{E}, \mathbf{a}, \mathbf{r})$ such that for all $j = 1, 2, \ldots, s$

$$\mathbf{C}^{\mathbf{e}_{i_j}} = g^{r_j} h^{a_j}, \; I_j = g_t^{r_j} h^{a_j}. \tag{3.1}$$

4. The exchange publishes its RevelioBP proof as $(t, \mathbf{I}, \Pi_{\mathsf{RevBP}})$ and claims that $C_{\text{res}} = \prod_{j=1}^{s} I_j$ is a Pedersen commitment to its reserves $\sum_{j=1}^{s} a_j$.

Note that the tag I_j is a deterministic function of the output C_{i_j} at a given block height t. So if two exchanges try to use the same output in their respective RevelioBP proofs, the same tag I_j will appear in both their proofs, revealing the collusion.

The reason for changing the base g_t with the block height is to change the tag of the same output across RevelioBP proofs at different block heights. If g_t were unchanged (as in Revelio [24]), then the appearance of the same tag in two RevelioBP proofs at different block heights will reveal that some exchange-owned output has remained unspent between these two block heights.

As C_{res} is a Pedersen commitment with respect to bases g_t and h, the Pedersen commitment C_{liab} to the exchange's liabilities should also be generated using these bases. Otherwise, it will be not be possible to generate a range proof on $C_{\text{res}} C_{\text{liab}}^{-1}$.

The proof reveals the cardinality s of $\mathcal{C}_{\text{own}}^t$. An exchange which wants to hide the number of outputs it owns can create some outputs which commit to the zero amount and use these to pad the outputs with non-zero amounts. For example, suppose that the number of outputs owned by the exchange is expected to be in the range 600 to 1000. It can create 400 outputs which commit to the zero amount and use these to always pad the number s revealed in the proof to be always 1000. Of course, the exchange would need to spend a nominal amount as transaction fees in creation of such outputs.

Finally, note that an exchange can under-report its reserves by excluding an output it owns from the subset $\mathcal{C}_{\text{own}}^t$ used to generate the RevelioBP proof. An exchange may choose to do this if its liabilities are much lower than its reserves.

3.3.2 Proof Verification

Given a RevelioBP proof of reserves $(t, \mathbf{I}, \Pi_{\mathsf{RevBP}})$ from an exchange referring to the block height t, the verifier performs the following procedure:

1. First, it reads the set of all UTXOs at block height t and forms the vector $\mathbf{C} = (C_1, \ldots, C_n)$ such that C_is are listed in the order of their appearance on the blockchain.

2. It verifies the argument of knowledge Π_{RevBP} by checking that the verification equations described in Figure 3.1 hold.

3. Finally, the verifier checks if any of the tags in the $\mathbf{I}$ vector appear in another exchange's RevelioBP proof for the same block height t. If the same tag I_j appears in the RevelioBP proofs of two different exchanges, then collusion is declared and the proofs is considered invalid.

3.4 ZK Argument of Knowledge Π_{RevBP}

Let $\mathbf{C} = (C_1, \ldots, C_n)$ be the vector representation of the UTXO set $\mathcal{C}^t_{\text{utxo}}$ at block height t. Let $\mathbf{I} = (I_1, \ldots, I_s)$ be the tag vector published by the exchange as part of the RevelioBP proof. The exchange constructs an argument of knowledge Π_{RevBP} to convince a verifier of the following:

(i) It knows r_j and a_j such that $I_j = g_t^{r_j} h^{a_j}\ \forall j \in [s]$.

(ii) $\exists i_j \in [n]$ (index) such that $C_{i_j} = g^{r_j} h^{a_j}\ \forall j \in [s]$.

Note that while the existence of the indices i_j will be proved by Π_{RevBP}, the indices themselves are not revealed. Since the term h^{a_j} is common in both equations in the above statements, combining the two equations, we can equivalently state that the exchange knows r_j and i_j such the following holds for all $j \in [s]$

$$I_j g_t^{-r_j} = C_{i_j} g^{-r_j}. \tag{3.2}$$

In other words, knowledge of r_j and i_j for all $j \in [s]$ suffices for an honest exchange to construct Π_{RevBP}.

Consider the language $\mathcal{L}_{\mathsf{RevBP}}$ given in (3.3) where $\mathbf{r} = (r_1, r_2, \ldots, r_s) \in \mathbb{Z}_q^s$ and $\mathbf{e}_{i_j} \in \{0,1\}^n$ is a unit vector with a 1 at index i_j and zeros everywhere else. The language depends on the common reference string $crs = \{\mathbb{G}, q, g, h, g_t\}$.

$$\mathcal{L}_{\mathsf{RevBP}} = \left\{ (\mathbf{C}, \mathbf{I}) \;\middle|\; \begin{array}{c} \exists (\mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s}) \text{ such that} \\ I_j g_t^{-r_j} = \mathbf{C}^{\mathbf{e}_{i_j}} g^{-r_j}\ \forall j \in [s] \end{array} \right\} \tag{3.3}$$

To leverage the Bulletproofs framework for the construction of a *log*-sized argument of knowledge for the language $\mathcal{L}_{\mathsf{RevBP}}$, we need to do the following:

(i) Embed the secrets $(\mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$ as the exponents in a Pedersen vector commitment satisfying some inner product relation.

(ii) Using the public information $(\mathbf{C}, \mathbf{I})$, construct the base vectors of the Pedersen vector commitment in such a way that the prover would not know the discrete logarithm relation between elements of the base vectors.

The first requirement seems natural since the Bulletproofs technique helps us prove the knowledge of exponents in a Pedersen vector commitment satisfying some inner product relations. The second one is a more technical requirement. In Bulletproofs, the elements in the base vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$ are uniformly chosen from the group $\mathbb{G}$ to ensure that a discrete logarithm relation between them is not known to a PPT prover. The soundness of the Bulletproofs protocol relies on this assumption. Lai *et al* [18] noted that if base vector components are chosen from a blockchain a prover might know the discrete logarithm relation between them. To solve this problem, they proposed using a base vector which is the Hadamard product of the vectors taken from the blockchain (with a random exponent) and a randomly chosen base vector.

To construct a base vector satisfying the above requirements, we write the statement of $\mathcal{L}_{\mathsf{RevBP}}$ from (3.3) as

$$g^{-r_j} g_t^{r_j} \mathbf{C}^{\mathbf{e}_{i_j}} I_j^{-1} = 1 \ \forall j \in [s]. \tag{3.4}$$

For $u \xleftarrow{\$} \mathbb{Z}_q$, combining the above constraints, we have

$$\begin{aligned} & \prod_{j \in [s]} \left(g^{-r_j} g_t^{r_j} \mathbf{C}^{\mathbf{e}_{i_j}} I_j^{-1} \right)^{u^{j-1}} = 1, \\ \implies \quad & g^{-\langle \boldsymbol{u}^s, \boldsymbol{r} \rangle} g_t^{\langle \boldsymbol{u}^s, \boldsymbol{r} \rangle} \mathbf{C}^{\boldsymbol{u}^s \boldsymbol{\mathcal{E}}} \mathbf{I}^{-\boldsymbol{u}^s} = 1, \end{aligned} \tag{3.5}$$

where $\boldsymbol{\mathcal{E}}$ is a $s \times n$ matrix having the $\mathbf{e}_{i_j}$ vectors as rows. We write the exponents in (3.5) as *compressed secrets*, namely $\xi = -\langle \boldsymbol{u}^s, \boldsymbol{r} \rangle$, $\xi' = \langle \boldsymbol{u}^s, \boldsymbol{r} \rangle$, $\hat{\mathbf{e}} = \boldsymbol{u}^s \boldsymbol{\mathcal{E}}$ and let $\hat{I} = \mathbf{I}^{-\boldsymbol{u}^s}$. Given a vector $\mathbf{p} \in \mathbb{G}^{n+3}$ and a scalar $w \in \mathbb{Z}_q$, we construct the base and exponent vectors as follows

$$\mathbf{g}'_w \coloneqq \left((g \| g_t \| \mathbf{C} \| \hat{I})^{\circ w} \circ \mathbf{p} \right), \tag{3.6}$$

$$\mathbf{a}' \coloneqq (\xi \| \xi' \| \hat{\mathbf{e}} \| 1). \tag{3.7}$$

Note that the compressed secrets are a linear combination of the actual secrets. We need to append the actual secrets $(\mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$ for completeness to (3.7). Thus,

we have

$$\mathbf{g}_w \coloneqq \big[\big((g\|g_t\|\mathbf{C}\|\hat{I})^{\circ w} \circ \mathbf{p}\big)\|\mathbf{g}'\big], \tag{3.8}$$

$$\mathbf{a} \coloneqq \big[(\xi\|\xi'\|\hat{\mathbf{e}}\|1)\|(\mathbf{e}_{i_1}\|\ldots\|\mathbf{e}_{i_s}\|\mathbf{r})\big]. \tag{3.9}$$

where $\mathbf{g}' \xleftarrow{\$} \mathbb{G}^{sn+s}$. We now state a lemma in regards to the non-trivial discrete-log relation between the components of the base vector $\mathbf{g}_w$.

Lemma 1 *If the components of $\boldsymbol{p}$ are chosen uniformly from $\mathbb{G}$ and independent of $(\boldsymbol{C}, \boldsymbol{I})$, then a PPT adversary cannot find a non-trival discrete logarithm relation between components of $\boldsymbol{g}_w$.*

Proof: As the components of $\mathbf{p}$ and $\mathbf{g}'$ are uniformly chosen from $\mathbb{G}$, the components of $\mathbf{g}_w$ are iid with a uniform distribution in $\mathbb{G}$. Hence a PPT adversary cannot find a non-trivial discrete logarithm relation between these components. ■

We can now construct a Pedersen vector commitment as $A = (h')^r\mathbf{g}_w^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$ for appropriately chosen $\mathbf{b} \in \mathbb{Z}_q^N, \mathbf{h} \xleftarrow{\$} \mathbb{G}^N$ where $N = |\mathbf{a}|$, satisfying the first requirement in using the Bulletproofs framework. Owing to (3.5), (3.8), (3.9), we have $\mathbf{g}_w^{\mathbf{a}} = \mathbf{g}_{w'}^{\mathbf{a}}$ for any $w, w' \in \mathbb{Z}_q$. Thus, the above vector commitment to $\mathbf{a}$ remains the same for any $w \in \mathbb{Z}_q$. By successfully running the Bulletproofs protocol twice on A with respect to two different bases $(h'\|\mathbf{g}_w\|\mathbf{h})$ and $(h'\|\mathbf{g}_{w'}\|\mathbf{h})$ we can extract the secret vector $\mathbf{a}$ such that $A = (h')^r\mathbf{g}_w^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} = (h')^r\mathbf{g}_{w'}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$. This solves the problem of extractability (soundness) of the protocol. In summary, we are now ready to construct a Bulletproofs-based argument of knowledge proving that the exchange some outputs from the entire set of UTXOs.

The interactive protocol $\Pi_{\mathsf{RevBP}} = (\mathit{Setup}, \langle\mathscr{P}, \mathcal{V}\rangle)$ for the language $\mathscr{L}_{\mathsf{RevBP}}$ is shown in Figure 3.1. Note that *Setup*, prover $\mathscr{P}$ and verifier $\mathcal{V}$ are PPT algorithms. The notation used in the protocol is given in Fig. 3.2, 3.3, 3.4, 3.5, 3.6.

Figure 3.1: Argument of knowledge for $\mathscr{L}_{\mathsf{RevBP}}$

$\mathit{Setup}(\lambda, \mathscr{L})$:

$\mathscr{L}(\mathbb{G}, q, g, h, g_t)$ as defined in (3.3),

Generate following elements randomly from $\mathbb{G}$

$h' \xleftarrow{\$} \mathbb{G}, \mathbf{p} \xleftarrow{\$} \mathbb{G}^{n+3}, \mathbf{g}' \xleftarrow{\$} \mathbb{G}^{sn+s}, \mathbf{h} \xleftarrow{\$} \mathbb{G}^N$

Output: $crs = (\mathbb{G}, q, g, h, g_t, h', \mathbf{p}, \mathbf{g}', \mathbf{h})$

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

$\mathscr{P}$:

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

(ii) $\mathbf{g}_0 = (\mathbf{p} \parallel \mathbf{g}')$

(iii) $A \coloneqq (h')^{r_A} \mathbf{g}_0^{\mathbf{c}_L} \mathbf{h}^{\mathbf{c}_R}$

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

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

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

(i) $\hat{I} \coloneqq \mathbf{I}^{-\boldsymbol{u}^s}$

(ii) $\mathbf{g}_w \coloneqq \left[\left((g \| g_t \| \mathbf{C} \| \hat{I})^{\circ w} \circ \mathbf{p}\right) \| \mathbf{g}'\right]$

$\mathscr{P}$:

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

$$\mathbf{s}_R[j] = \begin{cases} s_j \xleftarrow{\$} \mathbb{Z}_q & \text{if } n+4 \leq j \leq N-s, \\ 0 & \text{otherwise} \end{cases}$$

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

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

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

$\mathscr{P}$:

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

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

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

(ii) $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$

(iii) $T_1 = g^{t_1} h^{\tau_1}, T_2 = g^{t_2} h^{\tau_2}$

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

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

$\mathcal{P}$:

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

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

(iii) $\hat{t} \coloneqq \langle \boldsymbol{\ell}, \boldsymbol{\imath} \rangle \ \in \mathbb{Z}_q$

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

(v) $r \coloneqq r_A + r_S x$

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

$\mathcal{V}$:

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

(ii) $g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} g^{\delta} T_1^x T_2^{x^2}$ // *$\hat{t}$ satisfies $t_0 + t_1 x + t_2 x^2$*

(iii) $(h')^r \mathbf{g}_w^{\boldsymbol{\ell}} \mathbf{h}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\imath}} \stackrel{?}{=} A S^x \mathbf{g}_w^{\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\beta}}$ // *Check if $\boldsymbol{\ell} = l(x)$ and $\boldsymbol{\imath} = r(x)$*

Notation	*Description*
$\hat{I} = \hat{I}(u) \coloneqq \mathbf{I}^{-\boldsymbol{u}^s}$	Compressed key-images $I_1, I_2, \ldots, I_s$
$\boldsymbol{\mathcal{E}} \in \mathbb{Z}_2^{s \times n}$	Secret indices matrix, $vec(\boldsymbol{\mathcal{E}}) = (\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$
$\hat{\mathbf{e}} = \boldsymbol{u}^s \boldsymbol{\mathcal{E}}$ $\xi \coloneqq -\langle \boldsymbol{u}^s, \mathbf{r} \rangle$ $\xi' \coloneqq \langle \boldsymbol{u}^s, \mathbf{r} \rangle$	Compressed secrets $(\hat{\mathbf{e}}, \xi, \xi')$ satisfying $g^{\xi} g_t^{\xi'} \mathbf{C}^{\hat{\mathbf{e}}} \hat{I} = 1$
$\mathbf{c}_L, \mathbf{c}_R$	Honest encoding of witness (Fig. 3.3)
$N = sn + n + s + 3$	Size of the vectors $\mathbf{c}_L, \mathbf{c}_R$
$(\mathbf{v}_0, \ldots, \mathbf{v}_4)(u, v, y)$	Constraint vectors (Fig. 3.4, 3.5)
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\theta}(u, v, y, z)$	Compressed constraint vectors (Fig. 3.4, 3.5, 3.6)
$\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R)$	Relationship between valid witnesses (Fig. 3.6)

Figure 3.2: Notation used in the argument of knowledge Π_{RevBP}

$$\mathbf{c}_L \coloneqq (\ \xi \parallel \xi' \parallel\ \hat{\mathbf{e}}\ \parallel 1 \parallel \quad vec(\mathfrak{E}) \quad \parallel\ \mathbf{r}\ )$$
$$\mathbf{c}_R \coloneqq (\quad \mathbf{0}^{n+3} \quad \parallel\ \mathbf{1}^{sn} - vec(\mathfrak{E})\ \parallel\ \mathbf{0}^{s}\ )$$

Figure 3.3: Honest encoding of witness vectors

$$\begin{bmatrix}\mathbf{v}_0\\ \mathbf{v}_1\\ \mathbf{v}_2\\ \mathbf{v}_3\\ \mathbf{v}_4\end{bmatrix} \coloneqq \begin{bmatrix} \cdot & \cdot & \cdot & \cdot & \boldsymbol{y}^{sn} & \cdot \\ v & 1 & \cdot & \cdot & \cdot & (v-1)\boldsymbol{u}^s \\ \cdot & \cdot & -\boldsymbol{y}^n & \cdot & \boldsymbol{u}^s \otimes \boldsymbol{y}^n & \cdot \\ \cdot & \cdot & \cdot & y^s & \boldsymbol{y}^s \otimes \mathbf{1}^n & \cdot \\ \cdot & \cdot & \cdot & \cdot & \boldsymbol{y}^{sn} & \cdot \end{bmatrix}$$

Figure 3.4: Definitions of constraint vectors where dots mean zero scalars or vectors

$$\boldsymbol{\theta} \coloneqq \mathbf{v}_0, \qquad \boldsymbol{\mu} \coloneqq \sum_{i=1}^{4} z^i \mathbf{v}_i, \qquad \boldsymbol{\nu} \coloneqq z^4 \mathbf{v}_4,$$
$$\boldsymbol{\theta}^{\circ -1}[j] = \begin{cases} (\boldsymbol{\theta}[j])^{-1} & \text{if } \boldsymbol{\theta}[j] \neq 0, \\ 0 & \text{otherwise,} \end{cases}$$
$$\boldsymbol{\alpha} \coloneqq \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\nu}, \qquad \boldsymbol{\beta} \coloneqq \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mu},$$
$$\delta \coloneqq z^3 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle + \langle \mathbf{1}^N, \boldsymbol{\nu} \rangle.$$

Figure 3.5: Definitions of compressed constraint vectors

As the prover has to send vectors $\boldsymbol{\ell}, \boldsymbol{\imath} \in \mathbb{Z}_q^N$ in the last round, the Π_{RevBP} protocol results in a communication cost of $\mathcal{O}(N)$ for the prover, where N is the length of the secret vectors. We reduce this to $\mathcal{O}(\log_2(N))$ using the inner-product argument in [3]. Concretely, the language for the inner-product argument is expressed as

$$\mathcal{L}_{\mathsf{IP}} = \left\{ P \in \mathbb{G}, c \in \mathbb{Z}_q \;\middle|\; \begin{array}{c} \exists (\mathbf{a}, \mathbf{b}) \text{ such that} \\ P = u^c \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle \end{array} \right\} \tag{3.15}$$

where $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_q^{|\mathbf{a}|}$, $\mathbf{g}, \mathbf{h} \xleftarrow{\$} \mathbb{G}^{|\mathbf{a}|}$, $u \xleftarrow{\$} \mathbb{G}$. Thus, in our case, we construct a Pedersen vector commitment to $(\boldsymbol{\ell}, \boldsymbol{\imath})$

$$P = u^{\hat{t}} \mathbf{g}_w^{\boldsymbol{\ell}} (\mathbf{h}')^{\boldsymbol{\imath}} = u^{\hat{t}} (h')^{-r} A S^x \mathbf{g}_w^{\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\beta}},$$

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

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_0 \rangle = 0, \tag{3.10}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_1 \rangle = 0, \tag{3.11}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_2 \rangle = 0, \tag{3.12}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_3 \rangle = \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle, \tag{3.13}$$

$$\langle \boldsymbol{\gamma}_L + \boldsymbol{\gamma}_R - \mathbf{1}^t, \mathbf{v}_4 \rangle = 0. \tag{3.14}$$

Figure 3.6: A system of equations guaranteeing the integrity of the encoding of the witnesses

where $\mathbf{h}' = \mathbf{h}^{\boldsymbol{\theta}^{\circ -1}}$. Note that P, as shown above, could be computed by verifier. Thus, running the inner-product argument with input $(P, \hat{t})$ proves the knowledge of $(\boldsymbol{\ell}, \boldsymbol{r})$ in $\mathcal{O}(\log_2 N)$ communication. Furthermore, since the Π_{RevBP} protocol is public-coin, we can make it non-interactive using the Fiat-Shamir heuristic [33].

Theorem 1 *The argument presented in Figure 3.1 is public-coin, constant-round, perfectly complete and perfect special honest-verifier zero-knowledge.*

Proof sketch: The Π_{RevBP} protocol is public-coin since all of the challenges from $\mathcal{V}$ are generated uniformly randomly from $\mathbb{Z}_q$. Given a $crs = (\mathbb{G}, q, g, h, g_t)$ and an honest prover with knowledge of witness $wit = (\mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$ for a $stmt = (\mathbf{C}, \mathbf{I}) \in \mathcal{L}_{\mathsf{RevBP}}$, it is easy to see that the three verification conditions at the end of Figure 3.1 hold. Thus, the protocol is perfectly complete. Next, we need show that Π_{RevBP} is perfect special honest-verifier zero-knowledge (SHVZK) by constructing an efficient simulator $\mathcal{S}$, which can simulate the transcript of Π_{RevBP} without knowing the witness. Note the difference between a Special HVZK and HVZK is that in the former, the simulator $\mathcal{S}$ is given the challenges chosen by the verifier while in the latter, simulator $\mathcal{S}$ is allowed to choose the challenges by itself. Perfect SHVZK implies that no adversary, even if it is computationally unbounded, would be able to distinguish between the simulated and the honestly generated transcripts for valid statements and witnesses. For the protocol to be perfect SHVZK, the simulated transcript needs to be identically distributed to the transcript of Π_{RevBP}. Detailed construction of $\mathcal{S}$ is shown in Appendix 3.8.1.

Theorem 2 *Assuming the discrete logarithm assumption holds over* $\mathbb{G}$*,* Π_{RevBP} *has computational witness-extended-emulation for extracting a valid witness* wit*.*

Proof sketch: Proving that Π_{RevBP} has computational witness-extended emulation implies showing that it is *computationally sound*. To do so, we need to construct a PPT extractor $\mathscr{E}$, which when given enough number of transcripts of Π_{RevBP} via rewinding, succeeds in construction of a valid witness as defined in the language $\mathcal{L}_{\mathsf{RevBP}}$. We show the detailed construction of $\mathscr{E}$ in Appendix 3.8.2.

3.5 Security Properties of RevelioBP

In this section, we discuss the security properties of the Π_{RevBP} protocol, namely inflation resistance, collusion detection, and output privacy (as defined in Section 3.3). We defer the rigorous treatment of the security properties to an extended version of this paper.

3.5.1 Inflation Resistance

Theorem 2 proves that a PPT exchange can include the tag $I_j = g_t^{r_j} h^{a_j}$ in the vector $\mathbf{I}$ *only if* it knows the blinding factor r_j and amount a_j corresponding to the output $C_{i_j} = g^{r_j} h^{a_j}$. Thus an exchange can only create tags corresponding to outputs it owns. Furthermore, since each tag I_j is forced to be a Pedersen commitment to the *same amount* as the output C_{i_j}, the exchange cannot inflate the amount being contributed by I_j to C_{res}. Thus C_{res} is a Pedersen commitment to the actual reserves $\sum_{j=1}^{s} a_j$.

3.5.2 Collusion Resistance

Suppose two exchanges generate RevelioBP proofs at the same block height t. Ideally, each $C_i \in \mathcal{C}_{\text{utxo}}$ can contribute to the reserves of at most one of the exchanges. If exchange 1 who owns an output $C_i = g^{r_i} h^{a_i}$ reveals r_i and a_i to exchange 2, both of them can try using C_i as a contributing output in their proofs of reserves. Then while creating their respective arguments Π_{RevBP} both exchanges will be forced to include the tag $I_j = g_t^{r_i} h^{a_i}$ in their tag vectors, revealing the collusion. This technique will work only if all exchanges agree to use the same sequence of g_ts in their RevelioBP proofs. If exchanges 1 and 2 were to use different bases g_t and g_t' to generate their proofs, then collusion cannot be detected. As of now, pressure from customers and regulators seems to be the only way to ensure that all exchanges use the same g_t.

3.5.3 Output Privacy

Let λ be the security parameter such that $Setup(1^\lambda)$ generates the group $(\mathbb{G}, q, g)$ with $\log_2 q \approx \lambda$. Suppose an exchange publishes a polynomial number of proofs of reserves at block heights $t_1, t_2, \ldots, t_{f(\lambda)}$ where f is a polynomial. Output privacy requires that a PPT distinguisher $\mathcal{D}$, which is given the $f(\lambda)$ proofs of reserves as input, cannot do better than random guessing while classifying an output as owned by the exchange. Note that the Π_{RevBP} protocol itself does not reveal anything about the secrets. Here, we intend to analyse privacy of outputs due to the revelation of the tag vector.

The RevelioBP protocol provides output privacy under the following assumptions:

(i) The blinding factors of the exchange-owned outputs are chosen independently, uniformly from $\mathbb{Z}_q$.

(ii) The DDH problem is hard in the group $\mathbb{G}$, i.e. there is no algorithm which can solve the DDH problem in $\mathbb{G}$ with a running time polynomial in λ.

If the first assumption does not hold, a PPT adversary could identify exchange-owned outputs given a RevelioBP proof. For example, consider the case when two exchange-owned outputs C_i and C_j have the same blinding factor r but different amounts a_i and a_j, i.e. $C_i = g^r h^{a_i}$ and $C_j = g^r h^{a_j}$. If the exchange uses both outputs in a RevelioBP proof at block height t, then the tags corresponding to the outputs will be $I_j = g_t^r h^{a_i}$ and $I_j = g_t^r h^{a_j}$. An adversary can figure out that these two tags have the same blinding factor by checking if the equality $I_j h^{-a_1} = I_j h^{-a_2}$ holds for some $(a_1, a_2) \in V^2$ where V is the range of possible amounts. As the size of the set V is usually small, such a search is feasible. Once the amounts a_i and a_j have been found, the adversary could iterate through all possible output pairs (C, C') in $\mathcal{C}^t_{\text{utxo}} \times \mathcal{C}^t_{\text{utxo}}$ and check if $Ch^{-a_i} = C'h^{a_j}$. If a pair of outputs satisfying this equality is found, then the adversary concludes that both of them belong to the exchange. By requiring that the blinding factors are randomly chosen, such attacks become infeasible.

To precisely define output privacy, we use an experiment called `OutputPriv` which proceeds as follows:

1. For security parameter λ, the generate group parameters $(\mathbb{G}, q, g) \leftarrow Setup(1^\lambda)$ where $q \approx 2^\lambda$. A sequence of generators $g_1, g_2, \ldots, g_{f(\lambda)}$ are chosen uniformly from $\mathbb{G}$. These generators will be used to instantiate g_t in each of the $f(\lambda)$ RevelioBP proofs.

2. The exchange creates two outputs C_1, C_2 with amounts a_1, a_2 and blinding factors $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_q$, i.e. $C_1 = g^{r_1}h^{a_1}$ and $C_2 = g^{r_2}h^{a_2}$.

3. The exchange chooses an integer $\mathfrak{b} \xleftarrow{\$} \{1, 2\}$. Note that $C_{\mathfrak{b}} = g^{r_{\mathfrak{b}}}h^{a_{\mathfrak{b}}}$.

4. The exchange now generates $f(\lambda)$ RevelioBP proofs where the UTXO vector is $\mathbf{C} = (C_1, C_2)$ and the number of exchange-owned outputs is 1 in all of them. The lth proof reveals a single tag $I_l = g_l^{r_{\mathfrak{b}}}h^{a_{\mathfrak{b}}}$ for $l = 1, 2, \ldots, f(\lambda)$, i.e. the same exchange-owned output is used to generate each of the l proofs. Let the argument of knowledge corresponding to the lth RevelioBP proof be Π^l_{RevBP}.

5. The $f(\lambda)$ RevelioBP proofs consisting of tags $\bar{I} = (I_1, \ldots, I_{f(\lambda)})$, arguments $\bar{\Pi} = (\Pi^1_{\mathsf{RevBP}}, \ldots, \Pi^{f(\lambda)}_{\mathsf{RevBP}})$ along with the generators $\bar{g} = (g_1, g_2, \ldots, g_{f(\lambda)})$, outputs C_1, C_2, and amounts a_1, a_2 are given as input to a distinguisher $\mathscr{D}$ which outputs $\mathfrak{b}' \in \{1, 2\}$, i.e.

$$\mathfrak{b}' = \mathscr{D}\left(\bar{I}, \bar{\Pi}, \bar{g}, C_1, C_2, a_1, a_2\right) \tag{3.16}$$

6. $\mathscr{D}$ succeeds if $\mathfrak{b}' = \mathfrak{b}$. Otherwise it fails.

Definition 1 *The* RevelioBP *proof of reserves protocol provides output privacy if every PPT distinguisher $\mathscr{D}$ in the* `OutputPriv` *experiment succeeds with a probability which is negligibly close to $\frac{1}{2}$.*

To motivate the above definition, consider an adversary who observes a RevelioBP proof generated by an exchange for UTXO set $\mathcal{C}^t_{\text{utxo}}$ having size n. The length of the tag vector $\mathbf{I}$ reveals the number of outputs s owned by the exchange. Suppose the adversary is asked to identify an exchange-owned output from $\mathcal{C}^t_{\text{utxo}}$. If it chooses an output uniformly from $\mathcal{C}^t_{\text{utxo}}$, then it succeeds with probability $\frac{s}{n}$. But the adversary may itself own some outputs (say, n_a) in $\mathcal{C}^t_{\text{utxo}}$. Then, the success probability can be increased to $\frac{s}{n-n_a}$. The above definition models the extreme case when $n - n_a = 2$ and $s = 1$. The definition states that a PPT adversary can only do negligibly better than a random guessing strategy.

The justification for revealing the amounts a_1, a_2 to the distinguisher $\mathscr{D}$ is that the amount in a MimbleWimble output may be known to an entity other than the output owner. For instance, a MimbleWimble transaction where Alice sends some coins to Bob will result in a new output whose blinding factor is known only to Bob but the amount in this output is known to Alice. The above definition captures the requirement that even entities with knowledge of the amounts in outputs should not

be able to identify exchange-owned outputs from the RevelioBP proofs. We have the following theorem whose proof is given in Appendix 3.8.3.

Theorem 3 *The* RevelioBP *proof of reserves protocol provides output privacy in the random oracle model under the DDH assumption provided that the exchange chooses the blinding factors in its outputs uniformly and independently of each other.*

3.6 Performance

We compare the performance of our proof of reserves protocol with Revelio which was the first MimbleWimble proof of reserves protocol [24]. In Revelio, an exchange publishes an anonymity set $\mathcal{C}_{\text{anon}} \subseteq \mathcal{C}_{\text{utxo}}$ such that $\mathcal{C}_{\text{own}} \subseteq \mathcal{C}_{\text{anon}}$. In RevelioBP, the anonymity set is always equal to $\mathcal{C}_{\text{utxo}}$. For a fair comparison, we set $\mathcal{C}_{\text{anon}} = \mathcal{C}_{\text{utxo}}$ in Revelio. Let $n = |\mathcal{C}_{\text{utxo}}|$ and $s = |\mathcal{C}_{\text{own}}|$. Revelio proof sizes are $\mathcal{O}(n)$ while RevelioBP proof sizes are $\mathcal{O}(s + \log_2(sn))$. The logarithmic dependence of the RevelioBP proof size on the UTXO set size n is the main advantage of RevelioBP over Revelio. This is illustrated in Figure 3.7a where we compare the proof sizes of the Revelio and RevelioBP protocols as a function of n for $s = 20$. For a UTXO set size of 2^{17}, the RevelioBP proof is a mere 2.9 KB compared to a 300 MB Revelio proof.

We have implemented RevelioBP in Rust over the secp256k1 elliptic curve. The Revelio running times were estimated using the simulation code made available by Dutta *et al* [34]. All the experiments were run on an Intel Core i7 2.6 GHz CPU. Our simulation code is open-sourced on GitHub [35]. Figure 3.7c shows the RevelioBP and Revelio proof generation and verification times as a function of the UTXO set size for $s = 20$. For a constant own set size $s = 10$, we observe the following:

(i) A linear growth in the running times of both RevelioBP and Revelio as a function of n.

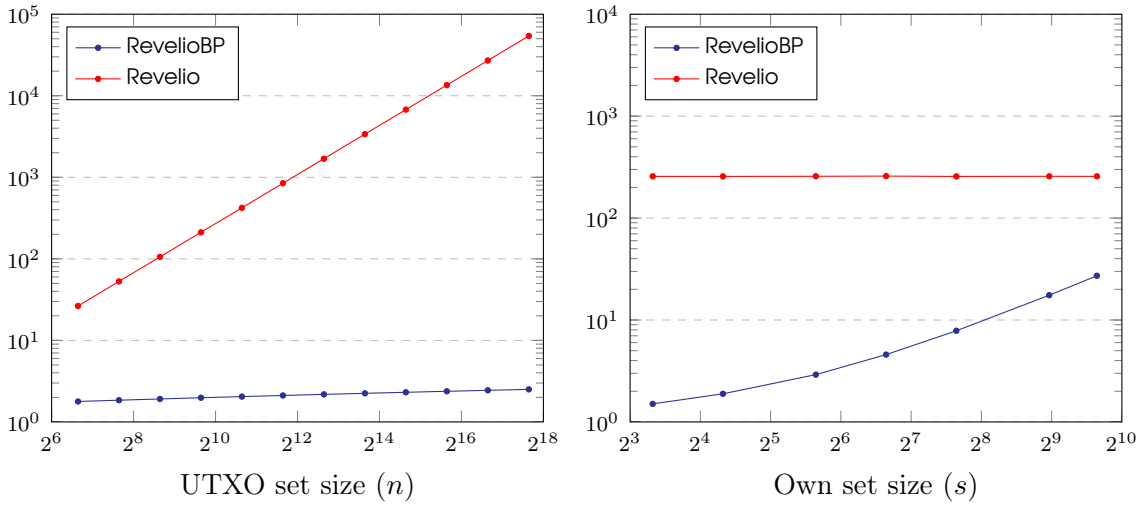
(ii) The RevelioBP proof generation is typically 2X slower than that of Revelio proof generation.

(iii) Although both the generation and verification of RevelioBP proofs if linear in n for a constant s, the verification is around 2.5X faster than its generation because of a single multi-exponentiation check in the verification of inner product protocol.

(iv) The RevelioBP verification is 20% faster than the verification of a Revelio proof.

(a) Proof size in KB for $s = 20$

(b) Proof size in KB for $n = 10^3$



(c) Running times in min, $s = 20$

(d) Running times in min, $n = 10^3$

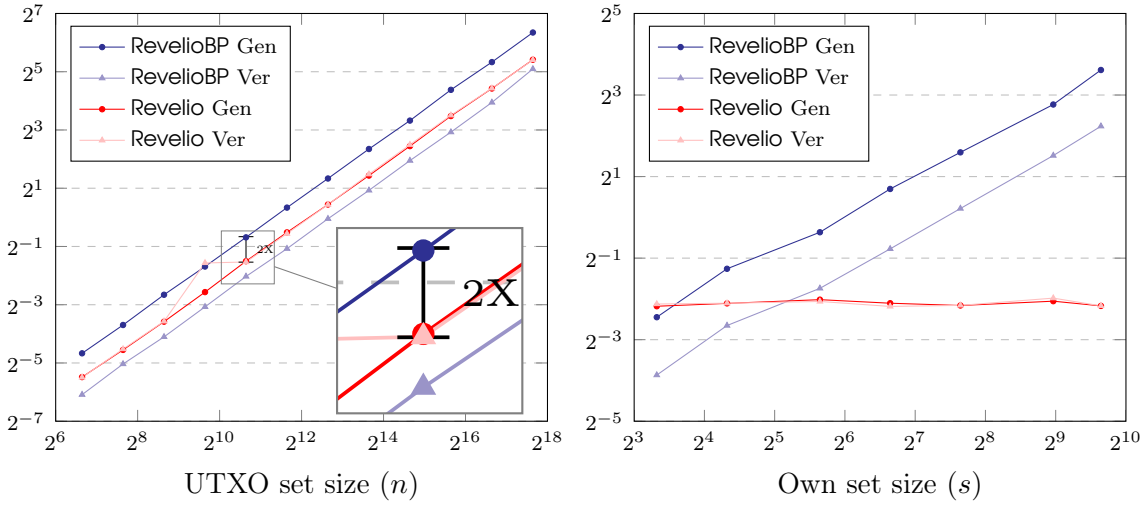


Figure 3.7: Performance comparison of RevelioBP and Revelio for $\mathbb{G} =$ `secp256k1` elliptic curve. All the plots are in log-log scale.

Furthermore, while the running time of Revelio is independent of s, it scales as $\mathcal{O}(sn)$ for RevelioBP. Figure 3.7d shows the generation and verification times respectively as a function of s for $n = 10^3$. Also, the proof size of RevelioBP proofs grow linearly with s while that of Revelio remains constant. This is shown in Figure 3.7b for a constant anonymity set size $n = 10^3$. This implies that for a constant UTXO set n, the verification in RevelioBP is faster than Revelio only upto a particular s. Similarly, the difference between RevelioBP and Revelio proof generation widens as s increases. To illustrate this in practical terms, the size of the current UTXO set in the Grin blockchain as of June 7, 2020 is 161,000 [29]. For this UTXO set size and own output set size equal to 100, an exchange could take

upto 160 minutes to generate a RevelioBP proof while taking less than 34 minutes to generate a Revelio proof. The customers of the exchange can verify RevelioBP and Revelio proofs in 68 and 34 minutes respectively. For an exchange which owns 200 outputs in the current UTXO set, the RevelioBP generation and verification times would rise to 320 minutes and 130 minutes respectively, while the timings of Revelio remain unchanged.

3.6.1 Scalability and Performance Trade-off

The smaller proof sizes of RevelioBP would enable exchanges to store several historical proofs for audit purposes. Further, if proofs of reserves are to be uploaded on the blockchain for public verifiability, smaller proof size becomes crucial. The benefits in scalability due to RevelioBP comes at the price of performance. From the customer point of view too, larger verification times are undesirable given their limited computational resources.

The simpler formulation of Revelio, on the other hand, allows faster generation and verification. Therefore, if the proof size is not a concern for an exchange, using Revelio can reduce computational cost for the exchanges as well as the customers. Moreover, the design of Revelio allows parallelization of its proof generation as well as verification. On the contrary, RevelioBP relies on the recursive inner product protocol, preventing parallelization of proof generation. Also, since the base vector used in RevelioBP depends on the tag vector published by an exchange, RevelioBP proofs of multiple exchanges on the same blockchain state cannot be aggregated. Lastly, if exchanges are required to generate proofs of reserves after every K blocks are mined, the proof generation time needs to be less than K minutes[†]. In such cases, proofs with smaller running times would be preferred.

3.7 Conclusion

To avoid proof sizes which are linear in the size of anonymity set, we have presented Bulletproofs-based proof of reserves protocol RevelioBP with proof size scaling logarithmically in the size of the anonymity set. Having implemented RevelioBP, we conclude that the smaller proof sizes it offers comes with the cost of larger proof generation and verification times. Revelio, the first proof of reserves for MimbleWimble, does better in terms of generation and verification times. An exchange has to make

[†]The average inter-block time is one minute in both Grin and Beam (the two most popular MimbleWimble-based cryptocurrencies).

a trade-off between scalability and performance and choose which protocol suits their needs better: RevelioBP with smaller proof size and large generation times or Revelio with larger proof sizes and faster generation times.

Acknowledgements

We would like to thank the anonymous reviewers for their insightful comments which helped improve this paper. We also acknowledge the support of the Bharti Centre for Communication at IIT Bombay. We thank Piyush Bagad (Wadhwani AI) for help with running the simulations.

3.8 Appendix

3.8.1 Proof of Theorem 1 (Perfect SHVZK)

Let the verifier challenges be (u, v, w, y, z, x). Simulator $\mathcal{S}$ computes $\hat{I}(u), \mathbf{g}_w(u, w)$ as described in Π_{RevBP}. We will use $\delta(u, v, y, z)$ to make the dependence of δ (as defined in Fig. 3.5) on the challenges explicit. Now, $\mathcal{S}$ samples the following quantities uniformly from respective groups: $S, T_2 \xleftarrow{\$} \mathbb{G}$, $\boldsymbol{\ell}, \boldsymbol{\imath} \xleftarrow{\$} \mathbb{Z}_q^N$ $\tau_x, r \xleftarrow{\$} \mathbb{Z}_q$. It then computes the remaining quantities corresponding to the ones which were sent by $\mathcal{P}$ to $\mathcal{V}$ in Π_{RevBP} as follows:

$$\hat{t} = \langle \boldsymbol{\ell}, \boldsymbol{\imath} \rangle, \tag{3.17}$$

$$T_1 = (g^{\hat{t}-\delta} h^{\tau_x} T_2^{-x^2})^{x^{-1}}, \tag{3.18}$$

$$A = (h')^r S^{-x} \mathbf{g}_w^{\boldsymbol{\ell}-\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\imath} - \boldsymbol{\beta}}. \tag{3.19}$$

Finally, $\mathcal{S}$ outputs the simulated transcript $(A, S, T_1, T_2, \boldsymbol{\ell}, \boldsymbol{\imath}, \hat{t}, \tau_x, r)$. Note that since $\boldsymbol{\ell}, \boldsymbol{\imath}$ are uniformly sampled from $\mathbb{Z}_q^N$, $\hat{t}$ is uniformly distributed in $\mathbb{Z}_q$. T_1 is also uniformly distributed in $\mathbb{G}$ as g, h, T_2 are uniformly sampled from $\mathbb{G}$ and the corresponding exponents are also uniformly distributed in $\mathbb{Z}_q$. From Lemma 1, recall that $\mathbf{g}_w$ can also be considered as uniformly distributed in $\mathbb{G}^{n+3}$. Thus, A is also uniformly distributed in $\mathbb{G}$ since the generators as well as exponents in the equation for computing A are uniformly distributed in the respective groups. This implies that all the elements produced by $\mathcal{S}$ and those produced by Π_{RevBP} are identically distributed and also satisfy the verification equations at the end of Figure 3.1. Therefore, the protocol is perfect special honest-verifier zero knowledge. ■

3.8.2 Proof of Theorem 2 (Soundness)

To prove that Π_{RevBP} has witness-extended emulation, first we state a couple of useful lemmas and a corollary. Before we proceed, we define some notation and a new system of constraint equations CS. Let $\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \in \mathbb{Z}_q^N$ be

$$\begin{aligned}\boldsymbol{\gamma}_L &\coloneqq (\gamma_{L,1} \| \gamma_{L,2} \| \boldsymbol{\gamma}_{L,3} \| \gamma_{L,4} \| \quad \boldsymbol{\gamma}_{L,5} \quad \| \boldsymbol{\gamma}_{L,6}),\\ \boldsymbol{\gamma}_R &\coloneqq (\underbrace{\gamma_{R,1}}_{1} \| \underbrace{\gamma_{R,2}}_{1} \| \underbrace{\boldsymbol{\gamma}_{R,3}}_{n} \| \underbrace{\gamma_{R,4}}_{1} \| \underbrace{\boldsymbol{\gamma}_{R,5}}_{sn} \| \underbrace{\boldsymbol{\gamma}_{R,6}}_{s}),\end{aligned}$$

where for $i \in \{L, R\}$, $\gamma_{i,j} \in \mathbb{Z}_q$ for $j \in \{1, 2, 4\}$, $\boldsymbol{\gamma}_{i,3} \in \mathbb{Z}_q^n$, $\boldsymbol{\gamma}_{i,6} \in \mathbb{Z}_q^s$ and for some matrices $\Gamma_L, \Gamma_R \in \mathbb{Z}_q^{s \times n}$, $\boldsymbol{\gamma}_{i,5} = vec(\Gamma_i) \in \mathbb{Z}_q^{sn}$. Define the constraint system CS with parameter u such that $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0 \iff$

$$\begin{cases} \boldsymbol{\gamma}_{L,5} \circ \boldsymbol{\gamma}_{R,5} &= \mathbf{0}^{sn}, & (3.20)\\ \gamma_{L,1} &= -\langle \boldsymbol{u}^s, \boldsymbol{\gamma}_{L,6} \rangle, & (3.21)\\ \gamma_{L,2} &= \langle \boldsymbol{u}^s, \boldsymbol{\gamma}_{L,6} \rangle, & (3.22)\\ \boldsymbol{\gamma}_{L,3} &= \boldsymbol{u}^s \Gamma_L, & (3.23)\\ \Gamma_L \mathbf{1}^n &= \mathbf{1}^s, & (3.24)\\ \gamma_{L,4} &= 1, & (3.25)\\ \boldsymbol{\gamma}_{L,5} &= -\boldsymbol{\gamma}_{R,5} + \mathbf{1}^{sn}. & (3.26) \end{cases}$$

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

Proof: Since $\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$ for sn different values of y, the following polynomials in y of degree at most $sn - 1$ have sn different roots. Thus, all of them must be equal to *zero* polynomials.

$$\begin{aligned}\langle \boldsymbol{\gamma}_{L,5} \circ \boldsymbol{\gamma}_{R,5}, \boldsymbol{y}^{sn} \rangle = 0 \qquad & \text{by (3.10)},\\ v \cdot \gamma_{L,1} + \gamma_{L,2} + (v-1)\langle \boldsymbol{\gamma}_{L,6}, \boldsymbol{u}^s \rangle = 0 \qquad & \text{by (3.11)},\\ \langle \boldsymbol{\gamma}_{L,3} - \boldsymbol{u}^s \Gamma_L, \boldsymbol{y}^n \rangle = 0 \qquad & \text{by (3.12)},\\ (\gamma_{L,4} - 1) y^s + \langle \Gamma_L \mathbf{1}^n - \mathbf{1}^s, \boldsymbol{y}^s \rangle = 0 \qquad & \text{by (3.13)},\\ \langle \boldsymbol{\gamma}_{L,5} + \boldsymbol{\gamma}_{R,5} - \mathbf{1}^{sn}, \boldsymbol{y}^{sn} \rangle = 0 \qquad & \text{by (3.14)}.\end{aligned}$$

Furthermore, since $\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$ for two different values of v too, the coefficient of v and the constant term in the second equation both must be *zero*. By comparing coefficients, we get $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$. ■

Lemma 3 *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 equations (3.20), (3.24) and (3.26). ■

Corollary 1 *Assuming the discrete logarithm assumption holds over* $\mathbb{G}$*, a* PPT *adversary cannot find a non-trivial discrete logarithm relation between the components of the base* $(h' \| \boldsymbol{g}_w \| \boldsymbol{h})$

Proof: Since $(h', \mathbf{h})$ are generated uniformly from $\mathbb{G}$, it is infeasible for a PPT adversary to compute its discrete log relation with base vector $\mathbf{g}_w$. Proving that a PPT adversary cannot find a non-trivial discrete log relation between components of $\mathbf{g}_w$ follows from Lemma 1. ■

With the above lemmas and the corollary, we proceed to construct an extractor $\mathscr{E}$. Let $pp \leftarrow \mathit{Setup}(\lambda)$ and $\mathit{stmt}, \mathit{wit} \leftarrow \mathscr{A}(pp)$. The aim of $\mathscr{E}$ is to produce a *valid* transcript and consequently the witness wit' corresponding to that transcript. Since $\mathscr{E}$ has oracle access to $\langle \mathscr{P}^\star(pp, \mathit{stmt}; \mathit{wit}), \mathcal{V}(pp, \mathit{stmt}) \rangle$ for any prover $\mathscr{P}^\star$, producing a valid transcript is trivial for $\mathscr{E}$. We hence focus on how $\mathscr{E}$ could extract a valid witness.

Extractor $\mathscr{E}$ runs $\mathscr{P}^\star$ on one value each of u, v, 2 different values of w, sn different values of y, 5 different values of z and 3 different values of x. This results in $30 \times sn$ 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.

Extracting A: Choose $k_i \in \mathbb{Z}_q$ for $i = 1, 2$ such that $\sum_{i=1}^{2} k_i = 1$ and $\sum_{i=1}^{2} k_i x_i = 0$. From (3.19), we have

$$\prod_{i=1}^{2} A^{k_i} = h^{\sum_i r_{x_i} k_i} S^{-\sum_i k_i x_i} \mathbf{g}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\alpha}(\sum_i k_i)} \mathbf{h}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}(\sum_i k_i)},$$

$$\implies A = h^{\sum_i r_{x_i} k_i} \mathbf{g}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\alpha}} \mathbf{h}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}},$$

$$\implies A = h^{r'_A} \mathbf{g}_w^{\mathbf{c}'_L} \mathbf{h}^{\mathbf{c}'_R}.$$

where $r'_A = \sum_i r_{x_i} k_i$, $\mathbf{c}'_L = (\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\alpha}$ and $\mathbf{c}'_R = \left(\sum_i k_i \cdot (\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i})\right) - \boldsymbol{\beta}$. 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. Write $\mathbf{c}'_L = (\mathbf{c}'_{L,1}\|\mathbf{c}'_{L,2})$ and $\mathbf{c}''_L = (\mathbf{c}''_{L,1}\|\mathbf{c}''_{L,2})$ of appropriate dimensions. We have

$$h^{r'_A}\mathbf{g}_w^{\mathbf{c}'_L}\mathbf{h}^{\mathbf{c}'_R} = h^{r''_A}\mathbf{g}_{w'}^{\mathbf{c}''_L}\mathbf{h}^{\mathbf{c}''_R} \implies$$
$$1 = h^{r'_A - r''_A}(g\|g_t\|\mathbf{C}\|\hat{I})^{w\cdot\mathbf{c}'_{L,1} - w'\cdot\mathbf{c}''_{L,1}}(\mathbf{p}\|\mathbf{g}')^{\mathbf{c}'_L - \mathbf{c}''_L}\mathbf{h}^{\mathbf{c}'_R - \mathbf{c}''_R}.$$

Now, if $r'_A \neq r''_A,\ \mathbf{c}'_L \neq \mathbf{c}''_L,\ \mathbf{c}'_R \neq \mathbf{c}''_R$, since $(\mathbf{p}\|\mathbf{g}'\|\mathbf{h})$ is uniformly chosen after fixing $(g\|g_t\|\mathbf{C}\|\hat{I})$, we would have violated the discrete logarithm assumption. Thus, $r'_A = r''_A,\ \mathbf{c}'_L = \mathbf{c}''_L,\ \mathbf{c}'_R = \mathbf{c}''_R$ and letting $\mathbf{c}'_L = (\xi'\|\xi''\|\hat{\mathbf{e}}'\|\psi')$, we get

$$\begin{aligned} 1 &= (g\|g_t\|\mathbf{C}\|\hat{I})^{(w-w')\cdot\mathbf{c}'_L}, \\ \implies \quad 1 &= (g\|g_t\|\mathbf{C}\|\hat{I})^{\mathbf{c}'_L} \text{ since } w \neq w', \\ \implies \quad 1 &= g^{\xi'}\cdot g_t^{\xi''}\cdot\mathbf{C}^{\hat{\mathbf{e}}'}\cdot\hat{I}^{\psi'}. \end{aligned} \tag{3.27}$$

We will use equation (3.27) in the last part of the proof.

Extracting S: $\mathscr{E}$ samples some $k_1, k_2 \in \mathbb{Z}_q$ such that $\sum_i k_i = 0$ and $\sum_i k_i x_i = 1$. From (3.19), we have

$$S = h^{r'_S}\mathbf{g}_w^{\sum_i k_i\cdot\boldsymbol{\ell}_{x_i}}\mathbf{h}^{\sum_i k_i\cdot\boldsymbol{\theta}^{\circ-1}\circ\boldsymbol{\tau}_{x_i}} = h^{r'_S}\mathbf{g}_w^{\mathbf{s}'_L}\mathbf{h}^{\mathbf{s}'_R}. \tag{3.28}$$

For a fixed w, the extracted A, S hold for all possible (x, y, z) because otherwise, the discrete log assumption would be violated owing to Corollary 1 as a non-trivial discrete logarithm representation of 1 with respect to the base $(h'\|\mathbf{g}_w\|\mathbf{h})$ would be known.

Substituting these expressions of A, S in the expressions for $\boldsymbol{\ell}, \boldsymbol{\tau}$ from the protocol, we get

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

These vectors must also hold for all (x, y, z) because if that was not the case, then we would know a non-trivial discrete logarithm representation of 1 with respect to the base $(h'\|\mathbf{g}_w\|\mathbf{h})$ due to Corollary 1.

Extracting T_1, T_2: $\mathscr{E}$ chooses $k_i \in \mathbb{Z}_q$ for $i \in \{1, 2, 3\}$ such that $\sum_{i=1}^3 k_i = 0$, $\sum_{i=1}^3 k_i x_i = 1$ and $\sum_{i=1}^3 k_i x_i^2 = 0$. Thus, we have

$$T_1 = \prod_{i=1}^3 T_1^{k_i x_i} = g^{\sum_{i=1}^3 k_i\hat{t}_{x_i}}h^{\sum_{i=1}^3 k_i\tau_{x_i}} = g^{t'_1}h^{r'_1}.$$

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 $\sum_{i=1}^{3} k'_i x_i^2 = 1$. Thus, we have

$$T_2 = \prod_{i=1}^{3} T_2^{k'_i x_i^2} = g^{\sum_{i=1}^{3} k'_i \hat{t}_{x_i}} h^{\sum_{i=1}^{3} k'_i \tau_{x_i}} = g^{t'_2} h^{r'_2}.$$

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 base $(g\|h)$ violating the discrete logarithm assumption. Therefore, we have obtained $t'_1, t'_2 \in \mathbb{Z}_q$ as the exponents of g in the extracted T_1 and T_2 respectively.

Extracting witness: $\mathscr{E}$ parses $\mathbf{c}'_L$ as below and outputs the witness wit'

$$\begin{aligned} \mathbf{c}'_L &= (\xi' \| \xi'' \| \hat{\mathbf{e}}' \| \psi' \| vec(\mathfrak{E}') \| \mathbf{r}'), \\ wit' &= (\mathbf{r}', \mathbf{e}'_{i_1}, \dots \mathbf{e}'_{i_s}). \end{aligned}$$

Finally, what remains to show is that the extracted witness is a valid witness to the statement $stmt$. Using the extracted t'_1, t'_2 we have

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

for all (x, y, z) or else we would violate the DL assumption by having a DL relation between g, h. Let

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

Now, the following polynomial, for all (y, z), has at least 3 roots and hence must be a zero polynomial.

$$t'(X) - (t'_0 + t'_1 X + t'_2 X^2).$$

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

$$t'_0 = z^3 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle + \langle \mathbf{1}^N, \boldsymbol{\nu} \rangle, \tag{3.29}$$

$$\begin{aligned} t'(0) &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\mu} \rangle + \langle \mathbf{c}'_R, \boldsymbol{\theta} \circ \boldsymbol{\alpha} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle, \\ &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \\ &\qquad \langle \mathbf{c}'_L + \mathbf{c}'_R, \boldsymbol{\nu} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle, \end{aligned} \tag{3.30}$$

where $\boldsymbol{\zeta} = \sum_{i=1}^{3} z^i \mathbf{v}_i$. Equations (3.29) and (3.30) imply

$$z^3\langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1}\rangle = \langle \mathbf{c}'_L, \mathbf{v}_0 \circ \mathbf{c}'_R\rangle + \sum_{i=1}^{3} z^i \langle \mathbf{c}'_L, \mathbf{v}_i\rangle + z^4\langle \mathbf{c}'_L + \mathbf{c}'_R - \mathbf{1}^N, \mathbf{v}_4\rangle.$$

The above equation holds for 5 different values of z. As the equation involves a degree 4 polynomial, the coefficients on both sides must be equal. This implies that $\mathsf{EQ}(\mathbf{c}'_L, \mathbf{c}'_R) = 0$ for sn different values of y and 2 values of v. By Lemma 2, we have $\mathsf{CS}(\mathbf{c}'_L, \mathbf{c}'_R) = 0$. Further, Lemma 3 implies that each row vector of $\boldsymbol{\mathcal{E}}'$ is a unit vector of length n. Let $vec(\boldsymbol{\mathcal{E}}') = (\mathbf{e}'_{i_1}, \ldots, \mathbf{e}'_{i_s})$ and write

$$\xi' = -\langle \boldsymbol{u}^s, \mathbf{r}'\rangle,\ \xi'' = \langle \boldsymbol{u}^s, \mathbf{r}'\rangle,\ \psi' = 1,\ \hat{\mathbf{e}}' = \boldsymbol{v}^s \boldsymbol{\mathcal{E}}'.$$

Also, let $i'_1, i'_2, \ldots, i'_s$ be the indices of the non-zero numbers in vector $\hat{\mathbf{e}}'$. We now show that these exponents computed from the extracted witness $(\mathbf{r}', \mathbf{e}'_{i_1}, \mathbf{e}'_{i_2}, \ldots, \mathbf{e}'_{i_s})$ is correct. From (3.27), we have

$$\begin{aligned} 1 &= g^{\xi'} \cdot g_t^{\xi''} \cdot \mathbf{C}^{\hat{\mathbf{e}}'} \cdot \hat{I}^{\psi'}, \\ &= g^{-\langle \boldsymbol{u}^s, \mathbf{r}'\rangle} \cdot g_t^{\langle \boldsymbol{u}^s, \mathbf{r}'\rangle} \cdot \left(\prod_{j=1}^{s} \mathbf{C}^{u^{j-1}\cdot \mathbf{e}'_{i_j}}\right) \cdot \left(\prod_{j=1}^{s} I_j^{-u^{j-1}}\right), \\ &= \prod_{j=1}^{s} \left(g^{-r'_j} g_t^{r'_j} \mathbf{C}^{\mathbf{e}'_{i_j}} I_j^{-1}\right)^{u^{j-1}}. \end{aligned}$$

The final equality can be interpreted as an evaluation of an s-degree polynomial in the exponent at a random point u. The probability of such an evaluation being zero for a non-zero polynomial is bounded by $\frac{s+1}{q}$, which is negligible since $q > 2^\lambda$ by Schwartz-Zippel lemma. Thus, we assume that the polynomial is always *zero*. This implies that for all $j \in [s]$, $g^{-r'_j} g_t^{r'_j} \mathbf{C}^{\mathbf{e}'_{i_j}} I_j^{-1} = 1$. Now the amount a'_j can be calculated (after extracting $(r'_j, \mathbf{e}'_{i_j})$) by an honest PPT prover (or extractor) since the amount lies in the finite range $\{0, 1, \ldots, 2^{64} - 1\}$. Therefore, wit' is a valid witness corresponding to the statement $stmt$ for the language $\mathcal{L}_{\mathsf{RevBP}}$. ■

3.8.3 Proof of Theorem 3

We will prove Theorem 3 by contradiction. We will prove that if there is a PPT distinguisher $\mathscr{D}$ who can succeed in the `OutputPriv` experiment with probability at least $\frac{1}{2} + \frac{1}{\mathfrak{p}(\lambda)}$ for a polynomial $\mathfrak{p}$, then we can construct a PPT adversary $\mathscr{E}$ who can solve the generalized DDH problem [36] with success probability at least $\frac{1}{2} + \frac{1}{2\mathfrak{p}(\lambda)}$.

This is a contradiction as the generalized DDH problem is equivalent to the DDH problem and the latter is assumed to be hard in the group $\mathbb{G}$.

Let $\mathscr{E}$ be an adversary who is tasked with solving the generalized DDH problem given a tuple $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$. $\mathscr{E}$ wants to distinguish between the following two cases:

- In the tuple $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$, $g_l \xleftarrow{\$} \mathbb{G}$, $u_l \xleftarrow{\$} \mathbb{G}$ $\forall l = 0, 1, 2, \ldots, f(\lambda)$.
- In the tuple $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$, $g_l \xleftarrow{\$} \mathbb{G}$, $u_l = g_l^r$ $\forall l = 0, 1, 2, \ldots, f(\lambda)$ where $r \xleftarrow{\$} \mathbb{Z}_q$.

Let $\mathfrak{d} = 1$ and $\mathfrak{d} = 2$ denote the above two cases, which are assumed to be equally likely. $\mathscr{E}$ needs to output its estimate $\mathfrak{d}'$ of $\mathfrak{d}$. To estimate $\mathfrak{d}$ correctly, $\mathscr{E}$ constructs a valid input to the `OutputPriv` distinguisher $\mathscr{D}$ as follows:

1. $\mathscr{E}$ sets $g = g_0$ and chooses $h \xleftarrow{\$} \mathbb{G}$. It also chooses amounts $a_1, a_2 \xleftarrow{\$} V$.
2. $\mathscr{E}$ chooses an integer $\mathfrak{b}$ uniformly from $\{1, 2\}$. It sets $C_{\mathfrak{b}} = u_0 h^{a_{\mathfrak{b}}}$ and chooses the other output uniformly from $\mathbb{G}$. For $l = 1, 2, \ldots, f(\lambda)$, $\mathscr{E}$ sets the tags $I_l = u_l h^{a_{\mathfrak{b}}}$.
3. For $l = 1, 2, \ldots, f(\lambda)$, $\mathscr{E}$ creates the lth argument Π_{RevBP} using the PPT simulator $\mathcal{S}$ in Appendix 3.8.2 with $g_t = g_l$, $\mathbf{C} = (C_1, C_2)$, and $\mathbf{I} = (I_l)$.
4. $\mathscr{E}$ feeds the computed quantities to $\mathscr{D}$ and gets

$$\mathfrak{b}' = \mathscr{D}\left(\left\{I_j, \Pi^j_{\mathsf{RevBP}}, g_j\right\}_{j=1}^{f(\lambda)}, C_1, C_2, a_1, a_2\right)$$

5. If $\mathfrak{b}' = \mathfrak{b}$, then $\mathscr{E}$ sets $\mathfrak{d}' = 2$. Otherwise, $\mathfrak{d}' = 1$.

The motivation behind this construction is that when $\mathscr{D}$ estimates $\mathfrak{b}$ correctly it could be exploiting some structure in the inputs given to it.

- When $\mathfrak{d} = 1$, the $u_0, u_1, \ldots, u_{f(\lambda)}$ components of the tuple given to $\mathscr{E}$ are uniformly distributed. This makes the distribution of $(I_1, I_2, \ldots, I_{f(\lambda)}, C_1, C_2)$ identical for both $\mathfrak{b} = 1$ and $\mathfrak{b} = 2$. This in turn makes the distributions of the simulated arguments $\Pi^1_{\mathsf{RevBP}}, \ldots, \Pi^{f(\lambda)}_{\mathsf{RevBP}}$ identical for both values of $\mathfrak{b}$. Thus $\mathscr{D}$ can only estimate $\mathfrak{b}$ with a success probability of $\frac{1}{2}$. Thus $\Pr[\mathfrak{b}' = \mathfrak{b} \mid \mathfrak{d} = 1] = \frac{1}{2}$.

- When $\mathfrak{d} = 2$, the $u_l = g_l^r$ for all $l = 0, 1, \ldots, f(\lambda)$. By construction, the vector $(I_1, I_2, \ldots, I_{f(\lambda)}, C_1, C_2)$ has a distribution which is different for $\mathfrak{b} = 1$ and $\mathfrak{b} = 2$. More importantly, the input $\mathscr{E}$ feeds to $\mathscr{D}$ is identically distributed to the input $\mathscr{D}$ receives in the `OutputPriv` experiment. If $\mathscr{D}$ can estimate $\mathfrak{b}$ correctly, then $\mathscr{E}$ bets on the distinguisher $\mathscr{D}$'s ability to win in the `OutputPriv` experiment and concludes that the tuple it received is a generalized DDH tuple.

Clearly, if adversary $\mathscr{D}$ is PPT then so is $\mathscr{E}$. Suppose there is a PPT distinguisher $\mathscr{D}$ which succeeds in the `OutputPriv` experiment with probability of success which is lower bounded by $\frac{1}{2} + \frac{1}{\mathfrak{p}(\lambda)}$ where $\mathfrak{p}$ is a polynomial. Thus we have $\Pr[\mathfrak{b}' = \mathfrak{b} \mid \mathfrak{d} = 2] \geq \frac{1}{2} + \frac{1}{\mathfrak{p}(\lambda)}$.

The success probability of $\mathscr{E}$ is given by

$$\begin{aligned}
\Pr[\mathfrak{d}' = \mathfrak{d}] &= \frac{1}{2}\Pr[\mathfrak{d}' = 1 | \mathfrak{d} = 1] + \frac{1}{2}\Pr[\mathfrak{d}' = 2 | \mathfrak{d} = 2], \\
&= \frac{1}{2}\Pr[\mathfrak{b}' \neq \mathfrak{b} \mid \mathfrak{d} = 1] + \frac{1}{2}\Pr[\mathfrak{b}' = \mathfrak{b} \mid \mathfrak{d} = 2], \\
&\geq \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \left(\frac{1}{2} + \frac{1}{\mathfrak{p}(\lambda)}\right) = \frac{1}{2} + \frac{1}{2\mathfrak{p}(\lambda)}.
\end{aligned}$$

Thus, $\mathscr{E}$ succeeds in solving the generalized DDH problem with a probability non-negligibly larger than $\frac{1}{2}$. As a PPT adversary who can solve the generalized DDH problem is equivalent to a PPT adversary solving the classical DDH problem [36], we get a contradiction. ■

Chapter 4

Literature Survey

Chapter 5

Materials and Methods

5.1 Including Figures

Chapter 6

Results and Discussions

6.1 Including Tables

Tables are to be used in a special environment so that they have a Number, caption and appear in the list of tables. Table 6.1 is a sample table. In the case of tables, it is a convention to write the caption above the table. Note that in the case of figures the caption appears below the figure.

Table 6.1: Physical properties of the materials used.

Property	Value
Particle Density, ρ_{p}	2500 kg/m^3
Viscosity, η_{s}	1 $\times 10^{-3}$ Pa-s

Appendix A

Supporting Material

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 10-JUNE-2020]. [Online]. Available: https://en.bitcoin.it/wiki/Mt._Gox

[3] 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.

[4] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://bitcoin.org/bitcoin.pdf

[5] M. Conti, E. S. Kumar, C. Lal, and S. Ruj, "A survey on security and privacy issues of bitcoin," *IEEE Communications Surveys & Tutorials*, vol. 20, pp. 3416–3452, 2018.

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

[7] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *2014 IEEE Symposium on Security and Privacy*, 2014, pp. 459–474.

[8] "Grin project website." [Online]. Available: https://grin-tech.org/

[9] "Beam project website." [Online]. Available: https://www.beam.mw/

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

[11] H. Natarajan, S. K. Krause, and H. L. Gradstein, "Distributed Ledger Technology (DLT) and blockchain (English)," *FinTech note;*

no. 1. Washington, D.C. : World Bank Group, 2017. [Online]. Available: http://documents.worldbank.org/curated/en/177911513714062215/Distributed-Ledger-Technology-DLT-and-blockchain

[12] K. Gary C., "An Overview of Cryptography," *Handbook on Local Area Networks*, September 1998. [Online]. Available: https://www.garykessler.net/library/crypto.html

[13] C. David, "Blind Signatures for Payments," *Springer-Verlag*, 1982. [Online]. Available: http://www.hit.bme.hu/~buttyan/courses/BMEVIHIM219/2009/Chaum.BlindSigForPayment.1982.PDF

[14] A. H. Koblitz, N. Koblitz, and A. Menezes, "Elliptic curve cryptography: The serpentine course of a paradigm shift," *Journal of Number Theory*, vol. 131, no. 5, pp. 781 – 814, 2011, elliptic Curve Cryptography. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022314X09000481

[15] S. Vijayakumaran. (2017) An introduction to Bitcoin, 2017. [Online]. Available: https://www.ee.iitb.ac.in/~sarva/bitcoin/bitcoin-notes-v0.1.pdf

[16] P. Hess, "Sec 2: Recommended elliptic curve domain parameters," 2000.

[17] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120âĂŞ126, Feb. 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[18] R. W. F. Lai, V. Ronge, T. Ruffing, D. Schröder, S. A. K. Thyagarajan, and J. Wang, "Omniring: Scaling private payments without trusted setup," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, November 2019, p. 31âĂŞ48.

[19] J.-J. Quisquater, M. Quisquater, M. Quisquater, M. Quisquater, L. Guillou, M. A. Guillou, G. Guillou, A. Guillou, G. Guillou, and S. Guillou, "How to explain zero-knowledge protocols to your children," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, pp. 628–631.

[20] S. Roose, "Standardizing Bitcoin Proof of Reserves," Blockstream Blog Post, Feb. 2018. [Online]. Available: https://blockstream.com/2019/02/04/standardizing-bitcoin-proof-of-reserves/

[21] 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.

[22] 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.

[23] 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.

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

[25] "CoinMarketCap Markets." [Online]. Available: https://coinmarketcap.com/#markets

[26] A. Wood, "QuadrigaCX Wallets Have Been Empty, Unused Since April 2018, Ernst and Young Finds," Mar. 2019. [Online]. Available: https://cointelegraph.com/news/report-quadrigacx-wallets-have-been-empty-unused-since-april

[27] "Ernst & Young Inc — Third Report of the Monitor," Mar. 2019. [Online]. Available: https://documentcentre.eycan.com/eycm_library/Quadriga%20Fintech%20Solutions%20Corp/English/CCAA/1.%20Monitor%27s%20Reports/4.%20Third%20Report%20of%20the%20Monitor/Third%20Report%20of%20the%20Monitor%20dated%20March%201,%202019.pdf

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

[29] Grinscan website. Date accessed: June 7, 2020. [Online]. Available: https://grinscan.net/charts

[30] "constants.rs — Grin rust-secp256k1-zkp GitHub repository." [Online]. Available: https://github.com/mimblewimble/rust-secp256k1-zkp/blob/master/src/constants.rs

[31] "Introduction to MimbleWimble and Grin." [Online]. Available: https://github.com/mimblewimble/grin/blob/master/doc/intro.md

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

[33] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology — CRYPTO' 86*, A. M. Odlyzko, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1987, pp. 186–194.

[34] Revelio simulation code. [Online]. Available: https://github.com/avras/revelio

[35] RevelioBP simulation code. [Online]. Available: https://github.com/suyash67/RevelioBP

[36] F. Bao, R. H. Deng, and H. Zhu, "Variations of Diffie-Hellman problem," in *Information and Communications Security*, 2003, pp. 301–312.

List of Publications

[1] S. Bagad and S. Vijayakumaran, "On the confidentiality of amounts in grin," in *Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2020.

[2] S. Bagad and S. Vijayakumaran, "Performance trade-offs in design of mimblewimble proofs of reserves," in *IEEE European Symposium on Security and Privacy Workshops (EuroS PW)*, 2020.

Acknowledgements

I would like to express my deepest appreciation and gratitude to my guide Prof. Saravanan Vijayakumaran for his guidance and constant supervision and help throughout the last 16 months. His insightful suggestions and comments have time and again helped me get a more in-depth understanding of the field. His most valuable lesson for me, from amongst many others, was to be persistent especially in times when things are not going as planned. Not only did he offer consistent assistance with regards to the academic work but also gave invaluable suggestions in helping me carve out my career path. Thanks are also due to Arijit Dutta of IIT, Bombay for his many useful remarks and discussions about the project and otherwise. I also acknowledge the help offered by my brother Piyush Bagad (Wadhwani AI) in understanding Python as well as in running simulations. Finally, I would also like to thank my dear friend Madhura Pawar for her constant encouragement which helped me work towards my goal even when things weren't working out.

Suyash Bagad
IIT Bombay
20 June 2020