Efficient Proof of Reserves Protocols for Cryptocurrency Exchanges

A Seminar Report of Dual Degree Project - Phase I
Submitted in partial fulfillment of
the requirements for the degree of
B.Tech + M.Tech with specialisation in Communication and Signal Processing
by

Suyash Bagad
(Roll No. 15D070007)

Supervisor:
Prof. Saravanan Vijayakumaran

Department of Electrical Engineering
Indian Institute of Technology Bombay
Mumbai - 400076 (India)
25 October 2019

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 cryptoexchanges, they are essentially businesses that allow customers to trade cryptocurrencies or digital currencies for other assets including conventional fiat money or different digital currencies. In the early days of cryptocurrency, crypto exchanges were few are far between, but before long their number increased dramatically forming an integral part of the industry and responsible for the price and volume of the vast majority of 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. 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.

Proving cryptocurrency reserves has become an important task for businesses seeking to retain the trust of their customers. These exchanges are increasingly coming under pressure from users and regulators to prove they are managing their users funds correctly. In this work, we develope cryptographic techniques to build proofs of reserves for cryptocurrency exchanges. Specifically, we present Revelio+ [†], an efficient proof of reserves protocol for cryptocurrencies built on MimbleWimble.

Revelio (CVCBT 2019) is a proof of reserves protocol for MimbleWimble-based cryptocurrencies which provides some privacy to a cryptocurrency exchange by hid-

[†]This work has been submitted to *Financial Cryptography and Data Security 2020*.

ing the exchange-owned outputs in a larger anonymity set of unspent outputs. A larger anonymity set naturally implies better privacy of the outputs owned by the exchange. A drawback of Revelio is that the proof size scales linearly in the size of the anonymity set. To tackle this issue, we develop Revelio+, a Bulletproofs-based proof of reserves protocol with proof sizes which scale logarithmically in the size of the anonymity set. This improvement allows us to use the set of all UTXOs as the anonymity set, resulting in ameliorated privacy for the exchange. By linking proof generation to the blockchain state, Revelio+ also has better guarantees against collusion between exchanges.

Furthermore, we aim to enhance the existing proofs of reserves for other cryptocurrencies using the techniques we have built in Revelio+ as a part of the second phase of the project.

Acknowledgements

I would like to express my deepest appreciation and gratitude to my guide Prof. Saravanan Vijayakumaran for his guidance and constant supervision as well as for providing necessary information regarding the project. His insightful suggestions and comments time and again have helped me get a much more in-depth understanding of the field. Thanks are also due to Arijit Dutta of IIT, Bombay for his many useful remarks and discussions about the project and otherwise. Finally, I would 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
25 October 2019

Table of Contents

Chapter 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 [1]). 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 [2, 3], Monero [4], and MimbleWimble [5]. In fact, the protocols proposed by Decker *et al* [2] and Dagher *et al* [3] 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 [2] relies on a trusted hardware assumption which can be difficult to verify. And the proof of liabilities protocol in [3] 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. An exchange can reduce its reported liabilities by censoring its customer list.

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 [3, 4, 5] 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 [6]. 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 [7, 8]. 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 [9]. Beam [10] and Grin [11] are two implementations of the MimbleWimble protocol which are available on several exchanges [12, 13]. Revelio [5] 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 a bottleneck in scaling the anonymity set to the set of all unspent transaction outputs (UTXOs).

1.1 Contributions

Revelio+ is a proof of reserves protocol for MimbleWimble with proof sizes which scale *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. At the time of writing this report, the number of UTXOs on the Grin blockchain is approximately 94,000 [14]. A Revelio proof of reserves for this anonymity set will have size 26 MB. Using Revelio+ instead will reduce the proof size to 0.37 MB.[1] The design of Revelio+ is heavily inspired by the recent Omniring proposal for Monero ring confidential transactions by Lai *et al* [15], which itself is based on Bulletproofs [16].

1.2 Related Work

The first known approach for generating a proof of solvency was given by Greg Maxwell and Peter Todd [**?**]. However, this approach revealed the total amount of assets held by an exchange and the addresses storing these assets. Decker *et al.* proposed a method for proving solvency of exchanges which uses a trusted platform module (TPM) to generate the proof [2]. This method does not leak any information about the amounts or addresses owned by the exchange and outputs a boolean result

[1] Under the conservative assumption that the exchange owns 10% of all UTXOs.

indicating solvency or insolvency. While ideal in terms of privacy, it requires trust in the security of the TPM implementation.

Depending on conventional cryptographic assumptions, Dagher *et al.* proposed a scheme called Provisions which generates privacy-preserving proofs of solvency for Bitcoin exchanges [17]. This work opened new avenues for proofs of solvency for not only Bitcoin but also newer cryptocurrencies. As noted earlier, a robust proof of liabilities for exchanges seems difficult since it relies of the data of customers provided by the exchange. Conventional as well as advanced cryptographic techniques notwithstanding, an exchange could still succeed in concealing customer data to reveal squashed liabilities. Developing robust proof of liabilities avoiding such chickanery is open to research. This work focusses only on building efficient and privacy-preserving proofs of reserves. More specifically, we concentrate our attention towards proofs of reserves for MimbleWimble-based cryptocurrencies for first part of this work.

The first proof of reserves for Grin - a MimbleWimble-based cryptocurrency given by Dutta *et al.* [5] is called as Revelio. An exchange is said to be solvent if the reserves it owns is greater than or equal to its liabilities to the customers. The Revelio protocol outputs a Pedersen commitment C_{res} to an amount which is equal to the number of coins owned by the exchange. Given a Pedersen commitment C_{liab} to the total liabilities of the exchange, it can prove solvency via a range proof which shows that the amount committed to in $C_{\text{res}}C_{\text{liab}}^{-1}$ is non-negative. Given the set of exchange-owned outputs $\mathcal{C}_{\text{own}}$, it hides the outputs in $\mathcal{C}_{\text{own}}$ in a larger anonymity set denoted by $\mathcal{C}_{\text{anon}}$. An observer who is given the set of outputs $\mathcal{C}_{\text{anon}}$ is not able to identify which outputs belong to $\mathcal{C}_{\text{own}}$. Revelio protocol has the following features:

1. *Inflation resistance*: An exchange should not be able to a commitment to an amount which is greater the total number of coins it owns.

2. *Privacy of amounts*: For an output $C_i \in \mathcal{C}_{\text{own}}$ such that $C_i = g^{k_i}h^{v_i}$, the amount v_i should not be revealed. This is to conform with the privacy-focused design of Grin. So the point k_iG should not appear as part of the proof. Otherwise, an adversary can try to exhaustively search through the possible values of v_i which will result in the commitment C_i.

3. *Proof of non-collusion between exchanges*: Exchanges may collude and share outputs to produce their respective proofs of reserves. The protocol should detect such collusion if it is used by all the exchanges.

Revelio protocol does a wonderful job in ensuring a privacy-preserving proof of reserves for exchanges. It, however, has the following two practical limitations:

1. Although Revelio protocol succeeds in providing privacy to the exchange-owned outputs in the sense that no observer could tell which outputs from $\mathcal{C}_{\text{anon}}$ the exchange owns. However, the fact that $\mathcal{C}_{\text{own}}$ being a proper subset of $\mathcal{C}_{\text{anon}}$ is still public and that everyone knows the exchange-owned outputs are some of the outputs in $\mathcal{C}_{\text{anon}}$, this level of privacy is certainly far from perfect. The problem is that the proof size in Revelio is linear in the anonymity set size. If we were to consider the entire set of UTXOs on the blockchain in the anonymity set, the massive proof size immediately makes the proof impractical.

2. Furthermore, the collusion-resistance property of Revelio works only if all the exchanges generate their proofs using the same blockchain state. This is again impractical from the point of view of participating exchanges. Even if all exchanges honestly generate their proofs of reserves at same blockchain states, there is no way to check it explicitly and naturally brings in the notion of trust. To solve this issue, enforcing exchanges to generate proofs of reserves corresponding to the same blockchain state using cryptographic techniques becomes necessary.

The novelty of Revelio+ protocol lies in successfully alleviating both of these practicality constraints using cryptographic primitives.

Chapter 2

Mathematical Background

Before delving into the details of Revelio+ protocol, let us equip ourselves with some background knowledge about the math of cryptocurrencies.

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 Assumptions

Definition 2.2.1 (Discrete Log Relation)**.** *For all PPT adversaries $\mathcal{A}$ and for all $n \geq 2$, $\exists$ a negligible function $\mu(\lambda)$ s.t*

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

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.

We use additional cryptographic assumptions such as Decisional Diffie-Hellman and its variants as described in [15].

2.3 Cryptographic Commitments

Definition 2.3.1 (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:*

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

Definition 2.3.2 (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 2.3.3 (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,*

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

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

Definition 2.3.4 (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,*

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

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

Definition 2.3.5 (Pedersen Commitment)**.** $\mathbf{M}_{pp}, \mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order p.*

- Setup*:* $g, h \leftarrow \mathbb{G}$
- $\text{Com}(x; r) = (g^x h^r)$

Definition 2.3.6 (Pedersen Vector Commitment)**.** $\mathbf{M}_{pp} = \mathbb{Z}_p^n$, $\mathbf{R}_{pp} = \mathbb{Z}_p$, $\mathbf{C}_{pp} = \mathbb{G}$ *of order p.*

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

The Pedersen vector commitment is perfectly hiding and computationally binding under the discrete logarithm assumption.

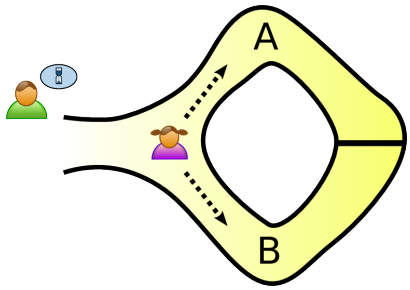
2.4 Zero-Knowledge Arguments of Knowledge

2.4.1 Zero-Knowledge Arguments

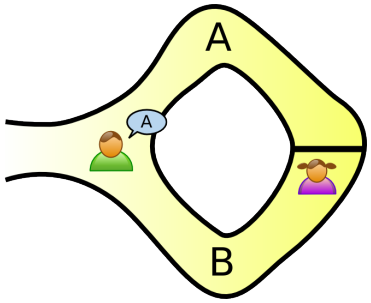
A protocol in which a prover convinces a verifier that a statement is true *without* revealing any information about why it holds is known as a Zero-knowledge

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.

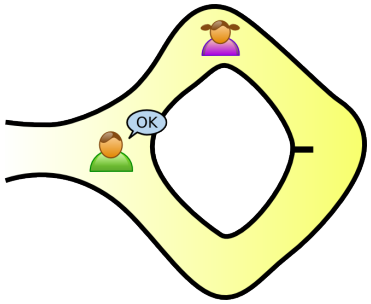
We illustrate the idea of zero-knowledge arguments of proof using the example of *Ali-Baba's secret cave* [18]. [1]



(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 2.1: Example of a zero knowledge proof

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 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.

[1] Figure courtesy: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`.

Peggy and Victor run the protocol described in figure 2.1. 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:

* Setup: $\sigma \leftarrow \text{Setup}(1^\lambda)$, σ is common reference string
* $\mathcal{P}$: prover, $\mathcal{V}$: verifier
* Transcript $tr \leftarrow \langle \mathcal{P}, \mathcal{V} \rangle$
* $\langle \mathcal{P}, \mathcal{V} \rangle = b$, $b = 0$ 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 $\mathcal{R}$.

2.4.2 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 2.4.1 (Argument of Knowledge)**.** *The triple $(\text{Setup}, \mathcal{P}, \mathcal{V})$ is called an argument of knowledge for relation $\mathcal{R}$ if it is perfectly complete and has computational witness-extended emulation.*

Definition 2.4.2 (Perfect completeness)**.** *$(\text{Setup}, \mathcal{P}, \mathcal{V})$ has perfect completeness if for all non-uniform polynomial time adversaries $\mathcal{A}$*

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

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 2.4.3 (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*

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

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 2.4.4 (Public coin)**.** *An argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ 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 2.4.5 (Zero Knowledge Argument of Knowledge)**.** *An argument of knowledge $(Setup, \mathcal{P}, \mathcal{V})$ 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 2.4.6 (Perfect Special Honest-Verifier Zero-Knowledge)**.** *A public coin argument of knowledge* $(Setup, \mathcal{P}, \mathcal{V})$ *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* $\mathcal{A}_1, \mathcal{A}_2$

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

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

PSHVZK AoK implies that even if an adversary chooses a distribution over statements and witnesses, it isnâĂŹt able to distinguish between simulated transcript and honestly generated transcript for $u \in \mathcal{L}_\sigma$.

Chapter 3

Cryptographic Preliminaries

3.1 Blockchain and Privacy

Modern day cryptography is based on the following primary functions [19]:

- *Privacy/confidentiality*: To ensure that no one can read or access the message except the intended receiver
- *Authentication*: Proving one's identity
- *Integrity*: Ensuring that the message intended to be received by the receiver is not altered in the path
- *Non-repudiation*: A protocol for checking if a message was actually generated by the sender
- *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 the *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 [20]. To motivate the need of *confidentiality* and *anonymity* in a digital transaction, we will consider an example in the context of Bitcoin and Blockchain.

In order to ensure that such a decentralized and public ledger is secure, Bitcoin uses a technology called *Blockchain*. Blockchain technology offers a way for untrusted parties to reach a consensus on a common digital history. The World Bank defines Blockchain as follows [21]:

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'.

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 withholds *anonymity* and *confidentiality*.

3.2 Bulletproofs

Bulletproofs [22] is a recently proposed zero-knowledge argument of knowledge system, to prove that a secret committed value lies in a given interval. Bulletproofs do not require a trusted setup. Further, in a distributed system where proofs are transmitted over a network or stored for a long time, short proofs reduce overall cost. Bulletproofs are short non-interactive zero-knowledge proofs which reduce the communication cost to the order of $\mathcal{O}(\log_2(n))$. They rely only on the discrete logarithm assumption, and are made non-interactive using the Fiat-Shamir heuristic. Bulletproofs support a simple and efficient multi-party computation (MPC) protocol that allows multiple parties with secret committed values to jointly generate an aggregated range proof for all their values, without revealing their secret values to each other. Bulletproofs heavily relies on the improved inner product argument proposed by Bünz *et al.* in [22]. This helps Bulletproofs reduce attain the state-of-the-art *logarithmic* proof size.

We now briefly describe inner-product argument. The inputs to the inner-product argument are independent generators $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, $P \in \mathbb{G}$ and a scalar $c \in \mathbb{Z}_q$. P is a binding vector commitment to $\mathbf{a}, \mathbf{b}$. The argument lets the prover convince a

verifier that the prover knows two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n$ such that

$$P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \qquad c = \langle \mathbf{a}, \mathbf{b} \rangle$$

The inner product argument is an efficient proof system for relation:

$$\{(\mathbf{g}, \mathbf{h} \in \mathbb{G}^n, P \in \mathbb{G}, c \in \mathbb{Z}_p;\ \mathbf{a}, \mathbf{b} \in \mathbb{Z}_p^n) :\ P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\} \tag{3.1}$$

To construct a range proof proving that an a lies in a range $\{0, 1, \ldots, 2^n - 1\}$, a can be written in its binary form $(a_1, a_2, \ldots, a_n) \in \{0, 1\}^n$. We let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$. Then, we formulate the range proof problem as an inner product problem by realising inner-product relationship between vectors $\mathbf{a}$ and $\mathbf{b}$. Lastly, using the improved inner-product argument, we achieve a logarithmic sized range proof.

3.3 Omniring

Omniring [15] is a Bulletproofs-based ring signature construction for Monero [23]. It leverages the Bulletproofs framework to achieve a proof size logarithmic in the ring size for Monero transactions. Our protocol takes inspiration from the construction of Omniring. To give an overview of the the idea behind Omniring construction, consider the ring $\mathcal{R} = \{R_1, \ldots, R_n\}$. Each of the addresses R_i is a public key with secret key x_i such that $R_i = g^{x_i}$. Let's say the prover owns some addresses $R_i \in \mathcal{R}$ and thus knows x_i and wishes to prove the knowledge of the same. Firstly, we rewrite the public-private key relation as $1 = g^{-x_i}\mathbf{R}^{\mathbf{e}_i}$ where $\mathbf{R} = (R_1, \ldots, R_n)$ and $\mathbf{e}_i$ is a unit vector with 1 in ith position and 0 everywhere else. Considering vector $\mathbf{a} = (-x_i, \mathbf{e}_i) \in \mathbb{Z}_q^{n+1}$ as a secret vector, we can formulate a Pedersen vector commitment with base $\mathbf{g} = (g, \mathbf{R})$. Concretely, for some $\mathbf{b} \in \mathbb{Z}_q$ as a function of $\mathbf{a}$ and generator $\mathbf{h} \in \mathbb{G}^{n+1}$ we can write $P = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$. Similar to the Bulletproofs formulation, we can now build some inner-product relation between $\mathbf{a}$ and $\mathbf{b}$ and consequently use improved inner-product argument.

However, we cannot directly use such a Bulletproofs-like argument here. This is because the extractability of the proof in Bulletproofs depends on the assumption that bases $\mathbf{g}, \mathbf{h} \in \mathbb{G}^{n+1}$ are uniformly independent over $\mathbb{G}^{n+1}$. That might not be the case here as the prover might know discrete logarithm relation between elements of base vector $\mathbf{g}$ or more specifically, public keys in the ring $\mathcal{R}$. To alleviate this issue, Lai *et al.* introduce the following base:

$$\mathbf{g}_w = (g, \mathbf{R})^{\circ w} \circ \mathbf{p}$$

for some $w \in \mathbb{Z}_q$ and a uniformly generated base vector $\mathbf{p} \xleftarrow{\$} \mathbb{G}^{n+1}$. Now, the newly constructed $\mathbf{g}_w$ is similar to that of a randomly generated base vector $\mathbf{g}$. Further, note that $\mathbf{g}_w^{\mathbf{a}} = \mathbf{g}_{w'}^{\mathbf{a}}$ for any $w, w' \in \mathbb{Z}_q$. Successfully running a Bulletproofs-like protocol twice with bases $\mathbf{g}_w, \mathbf{g}_{w'}$ solves the issue with extractability. The witness $(z\mathbf{a}, \mathbf{b})$ can be extracted by using the relation $P = \mathbf{g}_w^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} = \mathbf{g}_{w'}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$. Lai *et al.* extend this argument to construct a novel ring signature for Monero transactions.

3.4 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 a range like $\{0, 1, 2, \ldots, 2^{64} - 1\}$.

The group elements g and h are assumed to have an unknown discrete logarithm relationship. For example, in Grin $\mathbb{G}$ is equal to 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 [24]. 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 [25]. 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. Technically, ownership also requires knowledge of the amount a. But the amount is bounded by the number of coins which have been mined. For example, as of September 2019 the number of coins mined is approximately 36 million for Beam and 21 million for Grin. So the amount a can be found by brute force search given C and r.

3.5 From Omniring to Revelio+

In Monero, source addresses in a transaction are obfuscated using ring signatures and the amounts are hidden in Pedersen commitments [26]. The transaction structure, called *ring confidential transaction (RingCT)*, proves that (a) the transaction creator knows one of the private keys corresponding to the public keys in the ring, (b) the sum of the amounts in the input Pedersen commitments is equal to the sum of the

amounts in the output Pedersen commitments plus transaction fees, and (c) the output Pedersen commitments all commit to amounts in a valid range.

At present, the RingCT sizes in Monero scale linearly in the ring size. Omniring [15] is a recent proposal for RingCTs with proof sizes which scale logarithmically in the ring size. It relies on Bulletproofs [16] 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 $\mathsf{tag}_j = g^{x_j^{-1}}$ for $g \in \mathbb{G}$. This tag is used to detect double spending from a source address.

The design of Revelio+ 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 Revelio+ 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 $\mathsf{tag}_j = g_t^{r_j} h^{a_j}$ where $g_t \in \mathbb{G}$ is a randomly chosen group element. In Revelio+, 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.

Chapter 4

Revelio+Proof of Reserves Protocol

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 Revelio+ 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 $C_{\text{res}}C^{-1}_{\text{liab}}$ 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 Revelio+ does not reveal $\mathcal{C}^t_{\text{own}}$, it does reveal its cardinality $s_t = |\mathcal{C}^t_{\text{own}}|$. While this may seem like a privacy concern, an exchange can create some outputs which have zero amount in them for the purpose of *padding* the cardinality. We address this issue in a later section after describing the protocol.

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

- *Inflation resistance*: Using Revelio+, 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 Revelio+ proofs will be detected.
- *Output privacy*: A PPT adversary who observes a Revelio+ proof from an exchange cannot do any better than random guessing while identifying members of C^t_{own}.

The security proofs are given in Section 5.1.

4.1 Proof Generation

The Revelio+ protocol requires one randomly chosen group element $g_t \in \mathbb{G}$ per block. 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 Revelio+ proof of its reserves at the block with height t performs the following procedure:

1. From the UTXO set C^t_{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 $n = |C^t_{\text{utxo}}|$. 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 $C^t_{\text{own}} = \{C_{i_1}, C_{i_2}, \ldots, C_{i_s}\}$ of C^t_{utxo} where $1 \leq i_1 < i_2 < \cdots < i_s \leq n$. Once again, we suppress the dependence of s on t for notational simplicity. For each $C_{i_j} \in C^t_{\text{own}}$, 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 and 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 $\boldsymbol{\mathcal{E}} \in \{0,1\}^{s \times n}$ 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 of quantities $(\boldsymbol{\mathcal{E}}, \mathbf{a}, \mathbf{r})$ such that

$$\mathbf{C}^{\mathbf{e}_{i_j}} = g^{r_j} h^{a_j}, \tag{4.1}$$

$$I_j = g_t^{r_j} h^{a_j}, \tag{4.2}$$

   for all $j = 1, 2, \ldots, s$. Let $\Pi_{\text{Rev+}}$ denote this zero-knowledge argument.

4. The exchange publishes its Revelio+ proof as $(t, \mathbf{I}, \Pi_{\text{Rev+}})$ 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 Revelio+ 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 Revelio+ proofs at different block heights. If g_t were unchanged (as in Revelio [5]), then the appearance of the same tag in two Revelio+ 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.

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 Revelio+ proof. An exchange may choose to do this if its liabilities are much lower than its reserves.

4.2 Proof Verification

Given a Revelio+ proof of reserves $(t, \mathbf{I}, \Pi_{\text{Rev+}})$ 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 $\Pi_{\text{Rev+}}$ by checking that the verification equations described in Protocol 1 hold.

3. Finally, the verifier checks if any of the tags in the $\mathbf{I}$ vector appear in another exchange's Revelio+ proof. If the same tag I_j appears in the Revelio+ proofs of two different exchanges, then collusion is declared and the proofs of reserves is considered invalid.

4.3 Zero-Knowledge Argument of Knowledge $\Pi_{\text{Rev+}}$

For a positive integer s, let $[s]$ denote the set $\{1, 2, \ldots, s\}$. Let $\mathbf{C} = (C_1, \ldots, C_n)$ be the vector representation of the UTXO set $\mathcal{C}_{\text{utxo}}^t$ at block height t. Let $\mathbf{I} = (I_1, \ldots, I_s)$

be the tag vector published by the exchange as part of the Revelio+ proof. The objective of the argument of knowledge $\Pi_{\text{Rev+}}$ is to convince a verifier of the following statements:

(i) For each $j \in [s]$, the exchange knows r_j and a_j such that $I_j = g_t^{r_j} h^{a_j}$.

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

Note that while the existence of the indices i_j will be proved by $\Pi_{\text{Rev+}}$, the indices themselves are not revealed. Consider the language $\mathcal{L}_{\mathsf{Rev+}}$ given in equation (4.3) where $\mathbf{a} = (a_1, a_2, \ldots, a_s) \in \mathbb{Z}_q^s$, $\mathbf{r} = (r_1, r_2, \ldots, r_s) \in \mathbb{Z}_q^s$, and $\mathbf{e}_{i_j} \in \{0,1\}^n$ is a unit vector with a 1 at index i_j and 0s everywhere else. The language depends on the common reference string $\mathsf{crs} = \{\mathbb{G}, q, g, h, g_t\}$.

$$\mathcal{L}_{\text{Rev+}}(\mathsf{crs}) = \left\{ (\mathbf{C}, \mathbf{I}) \middle| \begin{array}{c} \exists(\mathbf{a}, \mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s}) \text{ such that } \forall j \in [s] \\ \mathbf{C}^{\mathbf{e}_{i_j}} = g^{r_j} h^{a_j} \text{ and } I_j = g_t^{r_j} h^{a_j} \end{array} \right\} \tag{4.3}$$

To leverage the Bulletproofs framework in the construction of an efficient argument of knowledge for the language $\mathcal{L}_{\text{Rev+}}$, we need to do the following:

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

2. 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 is necessary since the Bulletproofs technique helps one 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. This is necessary for the extractability of the proof. Lai *et al* [15] 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.

Concretely, let us concentrate on proving knowledge of only one unit vector $\mathbf{e}_{i_j}$ for some $j \in [s]$ and corresponding quantities a_j, r_j. From (4.3), we can write

$$g^{-r_j} h^{-a_j} \mathbf{C}^{\mathbf{e}_{i_j}} = 1 \text{ and } g_t^{-r_j} h^{-a_j} I_j = 1.$$

For $u, v \xleftarrow{\$} \mathbb{Z}_q$, we have

$$\left(g^{-r_j} h^{-a_j} \mathbf{C}^{\mathbf{e}_{i_j}}\right)^v \left(g_t^{-r_j} h^{-a_j} I_j\right)^u = g^{-vr_j} g_t^{-ur_j} h^{-(v+u)a_j} \mathbf{C}^{v\mathbf{e}_{i_j}} I_j^u = 1. \tag{4.4}$$

Given a vector $\mathbf{p} \in \mathbb{G}^{n+4}$ and a scalar $w \in \mathbb{Z}_q$, we construct a base

$$\mathbf{g}'_w = ((g \| g_t \| h \| \mathbf{C} \| I_j^u)^w \circ \mathbf{p}). \tag{4.5}$$

If the components of $\mathbf{p}$ are chosen uniformly and independently of $(\mathbf{C}, \mathbf{I})$, then $\mathbf{g}'_w$ is equivalent to a randomly generated base $\mathbf{g}$. Let the *compressed secrets* be $\xi_j = -vr_j,\ \xi' = -ur_j,\ \eta_j = -(v+u)a_j,\ \hat{\mathbf{e}}_j = v\mathbf{e}_{i_j}$ and let the corresponding vector be

$$\mathbf{a}' \coloneqq (\xi_j \| \xi'_j \| \eta_j \| \hat{\mathbf{e}}_j \| 1). \tag{4.6}$$

We can now construct a Pedersen vector commitment as $A = (h')^r (\mathbf{g}'_w)^{\mathbf{a}'} \mathbf{h}^{\mathbf{b}}$ for appropriately chosen $\mathbf{b} \in \mathbb{Z}_q^{n+4}, \mathbf{h} \xleftarrow{\$} \mathbb{G}^{n+4}$. Since $(\mathbf{g}'_w)^{\mathbf{a}'} = (\mathbf{g}'_{w'})^{\mathbf{a}'}$ for any $w, w' \in \mathbb{Z}_q$ (from equations (4.4), (4.5)), we note that 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 $A = (h')^r (\mathbf{g}'_w)^{\mathbf{a}'} \mathbf{h}^{\mathbf{b}} = (h')^r (\mathbf{g}'_{w'})^{\mathbf{a}'} \mathbf{h}^{\mathbf{b}}$.

To prove knowledge of the exact secrets $(r_j, a_j, \mathbf{e}_{i_j})$, we append the above base and the exponent vectors with $\mathbf{g}' \xleftarrow{\$} \mathbb{G}^{n+2}$ and $(\mathbf{e}_{i_j} \| a_j \| r_i)$ respectively.

$$\mathbf{g}_w \coloneqq \left[((g \| g_t \| h \| \mathbf{C} \| I_j^u)^w \circ \mathbf{p}) \| \mathbf{g}'\right] \tag{4.7}$$

$$\mathbf{a} \coloneqq \left[(\xi_j \| \xi'_j \| \eta_j \| \hat{\mathbf{e}}_j \| 1) \parallel (\mathbf{e}_{i_j} \| a_j \| r_j)\right] \tag{4.8}$$

Using the base vector $\mathbf{g}_w$ and secret vector $\mathbf{a}$, we can prove the knowledge of $(a_j, r_j, \mathbf{e}_{i_j})$ using Bulletproofs. In the $\Pi_{\text{Rev+}}$ protocol, for reasons of efficiency we use the aggregated Bulletproofs protocol to give a single Bulletproofs-like protocol proving the knowledge of $(\mathbf{a}, \mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$.

The interactive protocol $\Pi_{\text{Rev+}} = (\mathsf{Setup}, \langle \mathscr{P}, \mathscr{V} \rangle)$ for the language $\mathscr{L}_{\text{Rev+}}$ is shown in Protocol 1. Note that Setup, prover $\mathscr{P}$ and verifier $\mathscr{V}$ are PPT algorithms. The notation used in Protocol 1 is summarized in Figures 1, 2, 3, 4, 5.

Notation	*Description*
$\hat{I} = \hat{I}(u) \coloneqq \mathbf{I}^{\boldsymbol{u}^s}$	Compressed key-images with randomness $u \in \mathbb{Z}_q$
$\boldsymbol{\mathcal{E}} \in \mathbb{Z}_2^{s \times n}$	Matrix with row vectors $(\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$
$\hat{\mathbf{e}} = \boldsymbol{v}^s \boldsymbol{\mathcal{E}}$	Compressed vectors $(\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$ with randomness $v \in Z_q$
$\xi \coloneqq -\langle \boldsymbol{v}^s, \mathbf{r} \rangle$ $\xi' \coloneqq -\langle \boldsymbol{u}^s, \mathbf{r} \rangle$ $\eta \coloneqq -\langle \boldsymbol{v}^s + \boldsymbol{u}^s, \mathbf{a} \rangle$	Compressed secrets with randomness $u, v \in \mathbb{Z}_q$ such that $(\hat{\mathbf{e}}, \xi, \xi', \eta)$ satisfy $g^{\xi} \cdot g_t^{\xi'} \cdot h^{\eta} \cdot \mathbf{C}^{\hat{\mathbf{e}}} \cdot \hat{I} = 1$
$\mathbf{c}_L, \mathbf{c}_R$	Encoding of witness by an honest prover (Figure 2)
$N = sn + n + 2s + 4$	Size of the vectors $\mathbf{c}_L, \mathbf{c}_R$
$(\mathbf{v}_0, \ldots, \mathbf{v}_6)(u, v, y)$	Constraint vectors with randomness $u, v, y \in \mathbb{Z}_q$ See Figures 3 and 4
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \delta, \boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\theta}, \boldsymbol{\zeta}(u, v)$	Compressed constraint vectors with randomness $u, v \in \mathbb{Z}_q$ See Figures 3, 4, 5
$\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R)$	System of equations as defined in Figure 5

Fig. 1. Notation used in the argument of knowledge

$$
\begin{aligned}
\mathbf{c}_L &\coloneqq (\ \xi \ \|\ \xi' \ \|\ \eta \ \|\ \hat{\mathbf{e}} \ \|\ 1 \ \|\ \mathsf{vec}(\boldsymbol{\mathcal{E}}) \ \|\ \mathbf{a} \ \|\ \mathbf{r}\ ) \\
\mathbf{c}_R &\coloneqq (\ \mathbf{0}^{n+4} \ \|\ \mathsf{vec}(\boldsymbol{\mathcal{E}}) - \mathbf{1}^{sn} \ \|\ \mathbf{0}^{2s}\ )
\end{aligned}
$$

Fig. 2. Honest encoding of witness.

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

Fig. 3. Definitions of constraint vectors (Dots mean zero vectors).

$$\boldsymbol{\theta} \coloneqq \mathbf{v}_0, \qquad \boldsymbol{\zeta} \coloneqq \sum_{i=1}^{5} z^i \mathbf{v}_i,$$

$$\boldsymbol{\mu} \coloneqq \sum_{i=1}^{6} z^i \mathbf{v}_i, \qquad \boldsymbol{\nu} \coloneqq z^6 \mathbf{v}_6,$$

$$\boldsymbol{\alpha} \coloneqq \boldsymbol{\theta}^{\circ -1} \circ (-\boldsymbol{\nu}), \quad \boldsymbol{\beta} \coloneqq \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mu},$$

$$\delta \coloneqq z^5 \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle + \langle \mathbf{1}^t, \boldsymbol{\nu} \rangle$$

Fig. 4. Definitions of constraint vectors (continued).

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

$$\langle \boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \circ \mathbf{v}_0 \rangle = 0 \tag{4.9}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_1 \rangle = 0 \tag{4.10}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_2 \rangle = 0 \tag{4.11}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_3 \rangle = 0 \tag{4.12}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_4 \rangle = 0 \tag{4.13}$$

$$\langle \boldsymbol{\gamma}_L, \mathbf{v}_5 \rangle = \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle \tag{4.14}$$

$$\langle \boldsymbol{\gamma}_L - \boldsymbol{\gamma}_R - \mathbf{1}^t, \mathbf{v}_6 \rangle = 0 \tag{4.15}$$

Fig. 5. A system of equations guaranteeing integrity of encoding of witness.

Protocol 1. Argument of knowledge for language $\mathcal{L}_{\text{Rev+}}$.

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

$\mathcal{L}(\mathbb{G}, q, g, h, g_t)$ as defined in (4.3),
Generate: $h' \xleftarrow{\$} \mathbb{G},\ \mathbf{p} \xleftarrow{\$} \mathbb{G}^{n+4},\ \mathbf{g}' \xleftarrow{\$} \mathbb{G}^{N-n-4},\ \mathbf{h} \xleftarrow{\$} \mathbb{G}^N$
Output: $\mathsf{crs} = (\mathbb{G}, q, g, h, g_t, h', \mathbf{p}, \mathbf{g}', \mathbf{h})$

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

$\mathscr{V}$: $u, v \xleftarrow{\$} \mathbb{Z}$

$\mathscr{V} \longrightarrow \mathscr{P}$: u, v

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

1. $\hat{I} \coloneqq \mathbf{I}^{\boldsymbol{u}^s}$
2. For $w \in \mathbb{Z}_q$, write

$$\mathbf{g}_w \coloneqq \left[\left((g \| g_t \| h \| \mathbf{C} \| \hat{I})^{\circ w} \circ \mathbf{p} \right) \| \mathbf{g}' \right] \tag{4.16}$$

$\mathscr{P}$:

1. $r_A \xleftarrow{\$} \mathbb{Z}_q$
2. $A \coloneqq (h')^{r_A} \mathbf{g}_0^{\mathbf{c}_L} \mathbf{h}^{\mathbf{c}_R}$ *// commitment to* $\boldsymbol{c}_L,\ \boldsymbol{c}_R$

Note: $\mathbf{g}_w^{\mathbf{c}_L} = \mathbf{g}_{w'}^{\mathbf{c}_L}\ \forall w, w' \in \mathbb{Z}_q$ since $\left(g^{\xi} \cdot g_t^{\xi'} \cdot h^{\eta} \cdot \mathbf{C}^{\hat{\mathbf{e}}} \cdot \hat{I}\right) = 1$.

$\implies A = (h')^{r_A} \mathbf{g}_w^{\mathbf{c}_L} \mathbf{h}^{\mathbf{c}_R}\ \forall w \in \mathbb{Z}_q$

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

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

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

$\mathcal{P}$:

1. $r_S \xleftarrow{\$} \mathbb{Z}_q,\ \mathbf{s}_L, \mathbf{s}_R \xleftarrow{\$} \mathbb{Z}^N$ *// choose blinding factors $r_A,\ \boldsymbol{s}_L,\ \boldsymbol{s}_R$*
2. $S = (h')^{r_S} \mathbf{g}_w^{\mathbf{s}_L} \mathbf{h}^{\mathbf{s}_R}$ *// commitment to $\boldsymbol{s}_L,\ \boldsymbol{s}_R$*

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

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

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

$\mathcal{P}$:

1. Defining the following polynomials as a function of X in $\mathbb{Z}_q^N[X]$:

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

   for some $t_2, t_1, t_0 \in \mathbb{Z}_q$. Particularly, $t_0 = \delta(u, v, y)$.

2. $\tau_1, \tau_2 \xleftarrow{\$} \mathbb{Z}_q$ *// blinding factors for t_1, t_2*
3. $T_1 = g^{t_1} h^{\tau_1}, T_2 = g^{t_2} h^{\tau_2}$ *// commitments to t_1, t_2*

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

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

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

$\mathcal{P}$:

1. $\boldsymbol{\ell} \coloneqq l(x) = \mathbf{c}_L + \boldsymbol{\alpha} + \mathbf{s}_L \cdot x \in \mathbb{Z}_q^N$
2. $\boldsymbol{\imath} \coloneqq r(x) = \boldsymbol{\theta} \circ (\mathbf{c}_R + \mathbf{s}_R \cdot x) + \boldsymbol{\mu} \in \mathbb{Z}_q^N$
3. $\hat{t} \coloneqq \langle \boldsymbol{\ell}, \boldsymbol{\imath} \rangle \in \mathbb{Z}_q$
4. $\tau_x \coloneqq \tau_2 x^2 + \tau_1 x$ *// blinding factor for* $\hat{t}$
5. $r \coloneqq r_A + r_S x$

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

$\mathcal{V}$:

1. $\hat{t} \stackrel{?}{=} \langle \boldsymbol{\ell}, \boldsymbol{\imath} \rangle$ *// check if* $\hat{t}$ *is computed correctly*
2. $g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} g^{\delta} T_1^x T_2^{x^2}$ *// check that* $\hat{t}$ *satisfies* $\hat{t} = t_0 + t_1 x + t_2 x^2$
3. $(h')^r \mathbf{g}_w^{\boldsymbol{\ell}} \mathbf{h}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\imath}} \stackrel{?}{=} A S^x \mathbf{g}_w^{\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\beta}}$ *// check commitments to* $\boldsymbol{\ell}, \boldsymbol{\imath}$ *correct*

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

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

We use the definitions of a public-coin, perfectly complete and perfect special honest-verifier zero-knowledge argument of knowledge as given in [15]. The proofs for Theorem 1, 2 are given in Appendix A.1, A.2 respectively.

The $\Pi_{\text{Rev+}}$ protocol is public-coin as the challenges generated by $\mathcal{V}$ are uniformly randomly generated from $\mathbb{G}$. Thus, we can make it non-interactive using the Fiat-Shamir heuristic [27].

As the prover has to send vectors $\boldsymbol{\ell}, \boldsymbol{\imath} \in \mathbb{Z}_q^N$ in the last round, the $\Pi_{\text{Rev+}}$ 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 Bulletproofs technique [16].

Chapter 5

Results and Discussions

We now present the security analysis of the argument of knowledge $\Pi_{\text{Rev+}}$

5.1 Security Properties of Revelio+

In this section, we discuss the security properties of the Revelio+ protocol. We are concerned with inflation resistance, collusion detection, and output privacy (as defined in Section 4).

5.1.1 Inflation Resistance

The argument of knowledge $\Pi_{\text{Rev+}}$ given in Section 4.3 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 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 $\sum_{j=1}^{s} a_j$.

5.1.2 Collusion Resistance

Suppose two exchanges generate Revelio+ proofs at the same block height t. Ideally, each $C_i \in \mathcal{C}_{\text{utxo}}$ can be 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 $\Pi_{\text{Rev+}}$ 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 Revelio+ 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.

5.1.3 Output Privacy

Let λ be the security parameter such that $\mathsf{Setup}(1^\lambda)$ generates the group $(\mathbb{G}, q, g)$ with $\log_2 q = \lambda$. 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 $\mathscr{D}$, 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. The Revelio+ protocol provides output privacy under the following assumptions:

(i) The blinding factors of the exchange-owned outputs are chosen independently and 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 which is polynomial in λ.

If the first assumption does not hold, a PPT adversary could identify exchange-owned outputs given a Revelio+ 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 Revelio+ 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 group parameters are generated as $(\mathbb{G}, q, g) \leftarrow \mathsf{Setup}(1^\lambda)$ where $\log_2 q = \lambda$. 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 $f(\lambda)$ Revelio+ proofs.

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

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 $f(\lambda)$ Revelio+ 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 Revelio+ proof be $\Pi^l_{\text{Rev+}}$.

5. The $f(\lambda)$ Revelio+ proofs consisting of tags $I_1, I_2, \ldots, I_{f(\lambda)}$, arguments $\Pi^1_{\text{Rev+}}, \Pi^2_{\text{Rev+}}, \ldots, \Pi^{f(\lambda)}_{\text{Rev+}}$ along with the generators $g_1, g_2, \ldots, g_{f(\lambda)}$, outputs C_1, C_2, and amounts a_1, a_2 are given as input to a distinguisher $\mathcal{D}$ which outputs an integer $\mathfrak{b}' \in \{1, 2\}$, i.e.

$$\mathfrak{b}' = \mathcal{D}\left(I_1, \ldots, I_{f(\lambda)}, \Pi^1_{\text{Rev+}}, \ldots, \Pi^{f(\lambda)}_{\text{Rev+}}, g_1, \ldots, g_{f(\lambda)}, C_1, C_2, a_1, a_2\right) \tag{5.1}$$

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

The Revelio+ proof of reserves protocol provides output privacy if every PPT distinguisher $\mathcal{D}$ 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 Revelio+ 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 in $\mathcal{C}^t_{\text{utxo}}$. Suppose the number of adversary-owned outputs is n_a. Then by uniformly choosing from the set of UTXOs not owned by itself, 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 $\mathcal{D}$ 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 of output privacy captures the requirement that even entities with knowledge of the amounts in outputs should not be able to identify exchange-owned outputs from the Revelio+ proofs. We have the following theorem whose proof is given in Appendix B.

Theorem 3. *The* Revelio+ 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.

5.2 Performance

We compare the performance of our proof of reserves protocol with Revelio which was the first MimbleWimble proof of reserves protocol [5]. 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 Revelio+, 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 Revelio+ proof sizes are $\mathcal{O}(s + \log_2(sn))$. The logarithmic dependence of the Revelio+ proof size on the UTXO set size n is the main advantage of Revelio+ over Revelio. This is illustrated in Figure 6(a) where we compare the proof sizes of the Revelio and Revelio+ protocols as a function of n with $\mathbb{G}$ equal to the secp256k1 elliptic curve and $s = 50$. For a UTXO set size of 2^{21}, the Revelio+ proof is a mere 4.5 KB compared to a 580 MB Revelio proof.

We note that the main component of the Revelio+ proof generation algorithm is a Bulletproofs protocol [16] used to generate the argument $\Pi_{\text{Rev+}}$. The major difference is in the construction of the base vector used for computing Pedersen commitments to the secret vectors which results in more number of inner-product based constraint equations. However, since the basic framework remains the same, we can estimate the running time for generating the argument $\Pi_{\text{Rev+}}$ using Bulletproofs with appropriate input sizes.

We use an open-source Rust implementation of Bulletproofs [28] over the secp256k1 elliptic curve to estimate Revelio+ running times. The Revelio running times were estimated using the simulation code made available by Dutta *et al* [29]. All the experiments were run on a single core of a Intel i7-7700 3.6 GHz CPU. Figure 6(b) shows the Revelio+ and Revelio proof generation and verification times as a function of the UTXO set size for $s = 50$. For a UTXO set size equal to 10^5 and

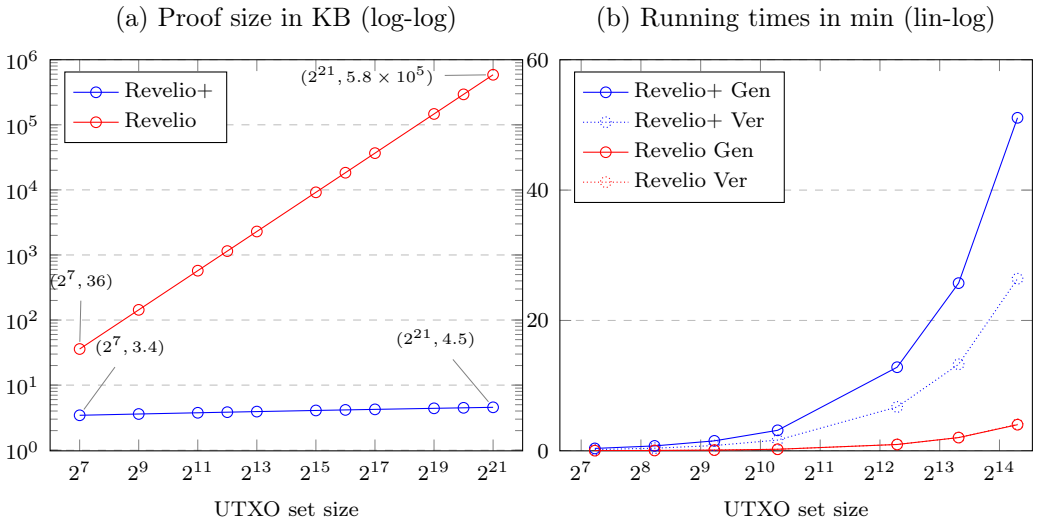


Fig. 6. Performance comparison of Revelio+ and Revelio

own output set size equal to 10^3, an exchange could take upto 80 hours to generate a Revelio+ proof while taking less than 20 minutes to generate a Revelio proof. Although Revelio+ proofs are much slower to generate, they are also much smaller in size. In situations where an exchange may need to store several historical proofs for audit purposes, proof size will be a bigger factor than generation time. Also, these generation times are based on unoptimized non-parallel code on commodity hardware. They can be decreased significantly using multicore CPUs or FPGAs.

5.3 Conclusion

We have presented an efficient proof of reserves protocol for MimbleWimble-based cryptocurrencies. For a fixed own output set size, the proof sizes in our protocol scale logarithmically in the size of the anonymity set, thus making it feasible to use the entire UTXO set as the anonymity set. In contrast, proof sizes in Revelio scale linearly in the size of the anonymity set. To keep the proof size small, Revelio proofs will have to compromise on output privacy by choosing small anonymity sets.

The proof generation time of our protocol is much larger than that of Revelio. Nevertheless, even in an unoptimized form it is small enough to allow the generation of a proofs every few days on commodity hardware. Proof generation times could be further reduced via Bulletproofs implementations on multicore CPUs or FPGAs.

Appendix A

Security Proofs for $\Pi_{\text{Rev+}}$

We present the complete security analysis of the argument of knowledge $\Pi_{\text{Rev+}}$ protocol in the following sections. Although the proofs have some similarities to the security proofs of the ring signature construction in Omniring [15], they are not trivial and necessary to describe the Revelio+ protocol completely.

A.1 Proof of Theorem 1 (Public-coin, Completeness & SHVZK)

The $\Pi_{\text{Rev+}}$ protocol is public-coin since all of the challenges from $\mathcal{V}$ are generated uniformly randomly from $\mathbb{Z}_q$ and $\mathbb{G}$. Let the language $\mathcal{L}_{\text{Rev+}}$ defined by equation (4.3). Given a $\mathsf{crs} = (\mathbb{G}, q, g, h, g_t)$ and an honest prover with knowledge of witness $\mathsf{wit} = (\mathbf{a}, \mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s})$ for a $\mathsf{stmt} = (\mathbf{C}, \mathbf{I}) \in \mathcal{L}_{\text{Rev+}}$, it is easy to see that the three verification conditions at the end of Protocol 1 hold. Thus, the protocol is perfectly complete. Next, we show that $\Pi_{\text{Rev+}}$ is perfect special honest-verifier zero-knowledge (SHVZK) by constructing an efficient simulator $\mathcal{S}$.

Let the verifier challenges (u, v, w, y, z, x) be provided by a PPT adversary $\mathcal{A}$. Simulator $\mathcal{S}$ computes $\hat{I}, \mathbf{g}_w$ as in $\Pi_{\text{Rev+}}$. That is,

$$\hat{I} = \mathbf{I}^{\boldsymbol{u}^s}, \tag{A.1}$$

$$\mathbf{g}_w = \left[((g \| g_t \| h \| \mathbf{C} \| \hat{I})^{\circ w} \circ \mathbf{p}) \| \mathbf{g}'\right]. \tag{A.2}$$

Now, $\mathcal{S}$ samples the following quantities uniformly from respective groups: $S, T_2 \xleftarrow{\$} \mathbb{G}$, $\boldsymbol{\ell}, \boldsymbol{\imath} \xleftarrow{\$} \mathbb{Z}_q^t$ $\tau_x, r \xleftarrow{\$} \mathbb{Z}_q$. It then computes the remaining quantities

corresponding to the ones which were sent by $\mathcal{P}$ to $\mathcal{V}$ in $\Pi_{\text{Rev+}}$ as follows:

$$\hat{t} = \langle \boldsymbol{\ell}, \boldsymbol{\imath} \rangle, \tag{A.3}$$

$$T_1 = (g^{\hat{t}-\delta} h^{\tau_x} T_2^{-x^2})^{x^{-1}}, \tag{A.4}$$

$$A = (h^r S^{-x} \mathbf{g}_w^{\boldsymbol{\ell}-\boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\theta}^{\circ-1} \circ \boldsymbol{\imath} - \boldsymbol{\beta}}). \tag{A.5}$$

Finally, $\mathcal{S}$ outputs $(A, S, T_1, T_2, \boldsymbol{\ell}, \boldsymbol{\imath}, \hat{t}, \tau_x, r)$. Note that since $\boldsymbol{\ell}, \boldsymbol{\imath}$ are uniformly sampled from $\mathbb{Z}_q^t$, $\hat{t}$ 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 sampled. Recall that $\mathbf{g}_w$ also is uniformly distributed in $\mathbb{G}^{n+4}$ as explained in Section 4.3. Thus, A is also uniformly distributed in $\mathbb{G}$ since the generators as well as exponents in the equation for computing A are uniformly sampled from respective groups. This implies that all the elements produced by $\mathcal{S}$ and those produced by $\Pi_{\text{Rev+}}$ are identically distributed and also satisfy the verification equations at the end of Protocol 1. Therefore, the protocol is perfect special honest-verifier zero knowledge. □

A.2 Proof of Theorem 2 (Soundness)

To prove that $\Pi_{\text{Rev+}}$ has witness-extended emulation, first we state a couple of useful lemmas and a corollary. Before we proceed, we define a few notations and a new system of constraint equations CS. Let $\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R \in \mathbb{Z}_q^N$ be defined as

$$\boldsymbol{\gamma}_L \coloneqq (\, \gamma_{L,1} \,\|\, \gamma_{L,2} \,\|\, \gamma_{L,3} \,\|\, \boldsymbol{\gamma}_{L,4} \,\|\, \gamma_{L,5} \,\|\, \boldsymbol{\gamma}_{L,6} \,\|\, \boldsymbol{\gamma}_{L,7} \,\|\, \boldsymbol{\gamma}_{L,8} \,),$$

$$\boldsymbol{\gamma}_R \coloneqq (\underbrace{\gamma_{R,1}}_{1} \| \underbrace{\gamma_{R,2}}_{1} \| \underbrace{\gamma_{R,3}}_{1} \| \underbrace{\boldsymbol{\gamma}_{R,4}}_{n} \| \underbrace{\gamma_{R,5}}_{1} \| \underbrace{\boldsymbol{\gamma}_{R,6}}_{sn} \| \underbrace{\boldsymbol{\gamma}_{R,7}}_{s} \| \underbrace{\boldsymbol{\gamma}_{R,8}}_{s}),$$

where for $i \in \{L, R\}$, $\gamma_{i,j} \in \mathbb{Z}_q$ for $j \in \{1, 2, 3, 5\}$, $\boldsymbol{\gamma}_{i,j} \in \mathbb{Z}_q^s$ for $j \in \{7, 8\}$, $\boldsymbol{\gamma}_{i,j} \in \mathbb{Z}_q^n$ for $j \in \{4\}$ and for some matrices $\Gamma_L, \Gamma_R \in \mathbb{Z}_q^{sn}$, $\boldsymbol{\gamma}_{i,6} = \mathsf{vec}(\Gamma_i) \in \mathbb{Z}_q^{sn}$. We now define the constraint system CS with parameters u, v.

$$\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0 \iff \begin{cases} \boldsymbol{\gamma}_{L,7} \circ \boldsymbol{\gamma}_{R,7} &= \mathbf{0}^{sn} & \text{(A.6)} \\ \gamma_{L,1} &= -\langle \boldsymbol{v}^s, \boldsymbol{\gamma}_{L,8} \rangle & \text{(A.7)} \\ \gamma_{L,2} &= -\langle \boldsymbol{u}^s, \boldsymbol{\gamma}_{L,8} \rangle & \text{(A.8)} \\ \gamma_{L,3} &= -\langle \boldsymbol{v}^s + \boldsymbol{u}^s, \boldsymbol{\gamma}_{L,7} \rangle & \text{(A.9)} \\ \boldsymbol{\gamma}_{L,4} &= \boldsymbol{v}^s \Gamma_L & \text{(A.10)} \\ \Gamma_L \mathbf{1}^n &= \mathbf{1}^s & \text{(A.11)} \\ \gamma_{L,5} &= 1 & \text{(A.12)} \\ \boldsymbol{\gamma}_{L,6} &= \boldsymbol{\gamma}_{R,6} + \mathbf{1}^{sn} & \text{(A.13)} \end{cases}$$

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

Proof: 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.

$$\langle \boldsymbol{\gamma}_L \circ \boldsymbol{\gamma}_L, \boldsymbol{y}^{sn} \rangle = 0 \qquad \text{by } Eq \text{ (4.9)}$$
$$\gamma_{L,1} + \langle \boldsymbol{\gamma}_{L,9}, \boldsymbol{v}^s \rangle = 0 \qquad \text{by } Eq \text{ (4.10)}$$
$$\gamma_{L,2} + \langle \boldsymbol{\gamma}_{L,9}, \boldsymbol{u}^s \rangle = 0 \qquad \text{by } Eq \text{ (4.11)}$$
$$\gamma_{L,3} + \langle \boldsymbol{\gamma}_{L,8}, \boldsymbol{v}^s + \boldsymbol{u}^s \rangle = 0 \qquad \text{by } Eq \text{ (4.12)}$$
$$\langle \boldsymbol{\gamma}_{L,4} - \boldsymbol{v}^s \Gamma_L, \boldsymbol{y}^n \rangle = 0 \qquad \text{by } Eq \text{ (4.13)}$$
$$((\gamma_{L,5} - 1)y^s + \langle \Gamma_L \mathbf{1}^n - \mathbf{1}^s, \boldsymbol{y}^s \rangle = 0 \qquad \text{by } Eq \text{ (4.14)}$$
$$\langle \boldsymbol{\gamma}_{L,7} - \boldsymbol{\gamma}_{R,7} - \mathbf{1}^{sn}, \boldsymbol{y}^{sn} \rangle = 0 \qquad \text{by } Eq \text{ (4.15)}$$

By comparing coefficients, we get $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$.

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

Proof: This follows trivially by observing $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R)$.

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

1. *On input $\mathbb{G}$, choose a vector of group elements $(\mathbf{C} \| \hat{I})$ and an integer w.*
2. *Receive a uniformly random vector $(h' \| \mathbf{p} \| \mathbf{g}' \| \mathbf{h})$ of correct dimensions.*
3. *Produce a vector $(r \| \mathbf{a} \| \mathbf{b})$ such that $1 = (h')^r \mathbf{g}_w^{\mathbf{a}} \mathbf{h}^{\mathbf{b}}$, where $\mathbf{g}_w = \left[\left((g \| g_t \| h \| \mathbf{C} \| \hat{I})^{\circ w} \circ \mathbf{p} \right) \| \mathbf{g}' \right]$*

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

Proof: The proof of this corollary very closely follows the proof of Corollary 1 in [15] and is therefore omitted.

With the above lemmas and the corollary, we proceed to construct an extractor $\mathscr{E}$. Let $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$ and $\mathsf{stmt}, \mathsf{wit} \leftarrow \mathcal{A}(\mathsf{pp})$. 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 $\langle \mathscr{P}^\star(\mathsf{pp}, \mathsf{stmt}; \mathsf{wit}), \mathcal{V}(\mathsf{pp}, \mathsf{stmt})\rangle$ 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 uniformly chosen challenge (u, v), 2 different values of w, sn different values of y, 7 different values of z, and 3 different values of x. This results in $42 \times sn$ transcripts. $\mathscr{E}$ fixes the values of (w, y, z) and runs $\mathscr{P}^\star$ for $x = (x_1, x_2, x_3)$. Let the transcripts for the respective x be $(A, S, T_1, T_2, \tau_{x_i}, r_{x_1}, \boldsymbol{\ell}_{x_i}, \boldsymbol{\mathfrak{r}}_{x_i}, \hat{t}_{x_i})$ 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 $\sum_{i=1}^{2} k_i x_i = 0$. Thus, we write from equation (A.5),

$$
\begin{aligned}
A^{k_i} &= h^{r_{x_i} k_i} S^{-x k_i} \mathbf{g}_w^{k_i \cdot (\boldsymbol{\ell}_{x_i} - \boldsymbol{\alpha})} \mathbf{h}^{k_i \cdot (\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i} - \boldsymbol{\beta})} \quad \forall i \in \{1, 2\} \quad \text{from (A.5)} \\
\implies \prod_{i=1}^{2} A^{k_i} &= h^{\sum_i r_{x_i} k_i} S^{\sum_i k_i x_i} \mathbf{g}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\alpha}(\sum_i k_i)} \mathbf{h}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}(\sum_i k_i)} \\
\implies A &= h^{\sum_i r_{x_i} k_i} \mathbf{g}_w^{(\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}) - \boldsymbol{\alpha}} \mathbf{h}^{(\sum_i k_i \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\mathfrak{r}}_{x_i}) - \boldsymbol{\beta}} \\
\implies A &= h^{r'_A} \mathbf{g}_w^{\mathbf{c}'_L} \mathbf{h}^{\mathbf{c}'_R}
\end{aligned}
$$

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 obtain the discrete logarithm representation of A, $\mathscr{E}$ 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 (probably) 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

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

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 $(g \| g_t \| h \| \mathbf{C} \| \hat{I})$, 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 $\mathbf{c}'_L = (\xi' \| \xi'' \| \eta' \| \hat{\mathbf{e}}' \| \psi'_1)$ we get

$$
\begin{aligned}
1 &= (g \| g_t \| h \| \mathbf{C} \| \hat{I})^{(w - w') \cdot \mathbf{c}'_L} \\
1 &= (g \| g_t \| h \| \mathbf{C} \| \hat{I})^{\mathbf{c}'_L} \text{ since } w \neq w' \\
1 &= \left(g^{\xi'} \cdot g_t^{\xi''} \cdot h^{\eta'} \cdot \mathbf{C}^{\hat{\mathbf{e}}'} \cdot \hat{I}^{\psi'} \right) \qquad \text{(A.14)}
\end{aligned}
$$

Extracting S: Similar to extraction of A, $\mathscr{E}$ samples some $k_1, k_2 \in \mathbb{Z}_q$ such that $k_1 + k_2 = 0$ and $k_1 x_1 + k_2 x_2 = 1$. We have

$$S^{x_i} = h^{r_{x_i}} A^{-1} \mathbf{g}_w^{\boldsymbol{\ell}_{x_i} - \boldsymbol{\alpha}} \mathbf{h}^{\boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\tau}_{x_i} - \boldsymbol{\beta}} \quad \forall i \in \{1,2\} \quad \text{from (A.5)}$$

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

$$\implies S = h^{r'_S} \mathbf{g}_w^{\sum_i k_i \cdot \boldsymbol{\ell}_{x_i}} \mathbf{h}^{\sum_i k_i \cdot \boldsymbol{\theta}^{\circ -1} \circ \boldsymbol{\tau}_{x_i}}$$

$$\implies S = h^{r'_S} \mathbf{g}_w^{\mathbf{s}'_L} \mathbf{h}^{\mathbf{s}'_R} \tag{A.15}$$

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

$$\boldsymbol{\ell}'_x = \mathbf{c}'_L + \boldsymbol{\alpha} + \mathbf{s}'_L \cdot x \in \mathbb{Z}_q^t$$
$$\boldsymbol{\tau}'_x = \boldsymbol{\theta} \circ (\mathbf{c}'_R + \mathbf{s}'_R \cdot x) + \boldsymbol{\mu} \in \mathbb{Z}_q^t$$

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: The verification equation for checking $\hat{t} = t_0 + t_1 x + t_2 x^2$ in the protocol can be rewritten as follows:

$$g^{\hat{t}} h^{\tau_x} \stackrel{?}{=} g^{\delta} T_1^x T_2^{x^2}$$
$$\iff T_1^x \stackrel{?}{=} g^{\hat{t} - \delta} h^{\tau_x} T_2^{-x^2} \quad \text{or} \quad T_2^{x^2} = g^{\hat{t} - \delta} h^{\tau_x} T_1^{-x}$$

$\mathscr{E}$ 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

$$\prod_{i=1}^{3} T_1^{k_i x_i} = g^{\sum_{i=1}^3 k_i \hat{t}_{x_i}} h^{\sum_{i=1}^3 k_i \tau_{x_i}}$$

$$\implies T_1 = g^{t'_1} h^{r'_1}$$

Similarly, to extract T_2, $\mathscr{E}$ chooses $k'_i \in \mathbb{Z}_q$ for $i \in \{1,2,3\}$ such that $\sum_{i=1}^3 k'_i = 0$, $\sum_{i=1}^3 k'_i x_i = 1$ and $\sum_{i=1}^3 k'_i x_i^2 = 0$.

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

$$\implies T_2 = g^{t'_2} h^{r'_2}$$

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.

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

$$\mathbf{c}'_L = (\xi'\|\xi''\|\eta'\|\hat{\mathbf{e}}'\|\psi'\|\mathsf{vec}(\mathfrak{E}')\|\mathbf{a}'\|\mathbf{r}')$$
$$\mathsf{wit}' = (\mathbf{a}', \mathbf{r}', \mathbf{e}'_{i_1}, \ldots \mathbf{e}'_{i_s})$$

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

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

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

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

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 $t'(X) = (t'_0 + t'_1 X + t'_2 X^2)\ \forall X \in \mathbb{Z}_q$ and particularly, $t'(0) = t'_0$. Thus, equating the following two polynomials in (y, z),

$$t'_0 = \delta(u, v) = z^5 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle + \langle \mathbf{1}^t, \boldsymbol{\nu} \rangle \tag{A.16}$$

$$\begin{aligned} t'(0) &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\mu} \rangle + \langle \mathbf{c}'_R, \boldsymbol{\theta} \circ \boldsymbol{\alpha} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle \\ &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \langle \mathbf{c}'_L, \boldsymbol{\nu} \rangle + \langle \mathbf{c}'_R, -\boldsymbol{\nu} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle \\ &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \langle \mathbf{c}'_L - \mathbf{c}'_R, \boldsymbol{\nu} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle \end{aligned} \tag{A.17}$$

Equations (A.16) and (A.17) imply

$$\begin{aligned} z^5 \cdot \langle \mathbf{1}^{s+1}, \boldsymbol{y}^{s+1} \rangle &= \langle \mathbf{c}'_L, \boldsymbol{\theta} \circ \mathbf{c}'_R \rangle + \langle \mathbf{c}'_L, \boldsymbol{\zeta} \rangle + \\ &\qquad \langle \mathbf{c}'_L - \mathbf{c}'_R - \mathbf{1}^t, \boldsymbol{\nu} \rangle + \langle \boldsymbol{\alpha}, \boldsymbol{\mu} \rangle \\ &= z^0 \langle \mathbf{c}'_L, \mathbf{v}_0 \circ \mathbf{c}'_R \rangle + \sum_{i=1}^{5} \langle \mathbf{c}'_L, \mathbf{v}_i \rangle \\ &\qquad + z^6 \langle \mathbf{c}'_L - \mathbf{c}'_R - \mathbf{1}^t, \boldsymbol{\nu} \rangle \end{aligned}$$

The above equation holds for 7 different values of z. Hence, the system of equations $\mathsf{EQ}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$ (Fig. 5) is satisfied for sn different values of y. By Lemma 1, constraint equations given by $\mathsf{CS}(\boldsymbol{\gamma}_L, \boldsymbol{\gamma}_R) = 0$. Further, Lemma 2 implies that each row vector of $\mathfrak{E}'$ is a unit vector of length n. Let $\mathsf{vec}(\mathfrak{E}') = (\mathbf{e}'_{i_1}, \ldots, \mathbf{e}'_{i_s})$ and write

$$\begin{aligned}
\xi' &= -\langle \boldsymbol{v}^s + u_1 \cdot \mathbf{1}^s, \mathbf{r}' \rangle \\
\xi'' &= \langle \boldsymbol{u}_2^s, \mathbf{r}' \rangle \\
\psi' &= 1 \\
\eta' &= -\langle \boldsymbol{v}^s + u_1 \cdot \mathbf{1}^s, \mathbf{a}' \rangle \\
\hat{\mathbf{e}}' &= \boldsymbol{v}^s \mathfrak{E}'
\end{aligned}$$

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 $(\mathbf{a}', \mathbf{r}', \mathbf{e}'_{i_1}, \mathbf{e}'_{i_2}, \ldots, \mathbf{e}'_{i_s})$ are correct. By equation (A.14),

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

The final equality could be interpreted as an evaluation of a s-degree polynomial in exponents at a random point (u, v) to 0. Hence, the probability of this happening when the polynomial is non-zero 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 $i \in [s]$

$$\begin{aligned}
\mathbf{C}^{\mathbf{e}'_{i_j}} &= g^{r'_j} h^{a'_j} \\
I_j &= g_t^{r'_j h^{a'_j}}
\end{aligned}$$

Hence, wit' is a valid witness corresponding to stmt for language $\mathcal{L}_{\mathrm{Rev}+}$.

□

Appendix B

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 [30] 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 $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$. Specifically, $\mathscr{E}$ wants to distinguish between the following two cases:

- In the tuple $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$, $g_l \xleftarrow{\$} \mathbb{G}$, $u_l \xleftarrow{\$} \mathbb{G}$ for all $l = 0, 1, 2, \ldots, f(\lambda)$.
- In the tuple $(g_0, g_1, \ldots, g_{f(\lambda)}, u_0, u_1, \ldots, u_{f(\lambda)}) \in \mathbb{G}^{2f(\lambda)+2}$, $g_l \xleftarrow{\$} \mathbb{G}$ and $u_l = g_l^r$ for all $l = 0, 1, 2, \ldots, f(\lambda)$ 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 an h randomly from $\mathbb{G}$, i.e. $h \xleftarrow{\$} \mathbb{G}$. It also chooses amounts a_1, a_2 randomly from the allowed range set V.
2. $\mathscr{E}$ chooses an integer $\mathfrak{b}$ uniformly from the set $\{1, 2\}$.
   (a) If $\mathfrak{b} = 1$, $\mathscr{E}$ sets $C_1 = u_0 h^{a_1}$ and chooses C_2 randomly from $\mathbb{G}$. For $l = 1, 2, \ldots, f(\lambda)$, $\mathscr{E}$ sets the tags $I_l = u_l h^{a_1}$.

(b) If $\mathfrak{b} = 2$, $\mathscr{E}$ sets $C_2 = u_0 h^{a_2}$ and chooses C_1 randomly from $\mathbb{G}$. For $l = 1, 2, \ldots, f(\lambda)$, $\mathscr{E}$ sets the tags $I_l = u_l h^{a_2}$.

3. For $l = 1, 2, \ldots, f(\lambda)$, $\mathscr{E}$ creates the lth argument $\Pi^l_{\text{Rev+}}$ using the PPT simulator $\mathcal{S}$ in Appendix A.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

$$\mathfrak{b}' = \mathscr{D}\left(I_1, \ldots, I_{f(\lambda)}, \Pi^1_{\text{Rev+}}, \ldots, \Pi^{f(\lambda)}_{\text{Rev+}}, g_1, \ldots, g_{f(\lambda)}, C_1, C_2, a_1, a_2\right).$$

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

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 $\Pi^1_{\text{Rev+}}, \ldots, \Pi^{f(\lambda)}_{\text{Rev+}}$ identical for both values of $\mathfrak{b}$. Thus $\mathscr{D}$ can only estimate $\mathfrak{b}$ with a success probability of $\frac{1}{2}$. So if $\mathscr{D}$ is wrong in its estimate of $\mathfrak{b}$, the $\mathscr{E}$ concludes that the input to $\mathscr{D}$ has no structure.

- 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

$$\Pr\left[\mathfrak{b}' = \mathfrak{b} \mid \mathfrak{d} = 2\right] \geq \frac{1}{2} + \frac{1}{\mathfrak{p}(\lambda)}. \tag{B.1}$$

We have already argued that

$$\Pr\left[\mathfrak{b}' = \mathfrak{b} \mid \mathfrak{d} = 1\right] = \frac{1}{2}. \tag{B.2}$$

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, $\mathscr{E}$ is 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 who can solve the classical DDH problem [30], we get a contradiction. It follows that any PPT distinguisher $\mathscr{D}$ in the `OutputPriv` experiment can only succeed with a probability which is negligibly close to $\frac{1}{2}$. □

References

[1] Steven Roose. Standardizing Bitcoin Proof of Reserves. Blockstream Blog Post, February 2018. https://blockstream.com/2019/02/04/standardizing-bitcoin-proof-of-reserves/.

[2] Christian Decker, James Guthrie, Jochen Seidel, and Roger Wattenhofer. Making bitcoin exchanges transparent. In *20th European Symposium on Research in Computer Security (ESORICS)*, pages 561–576, 2015.

[3] Dagher, Gaby G. and Bünz, Benedikt and Bonneau, Joseph and Clark, Jeremy and Boneh, Dan. Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, pages 720–731, New York, NY, USA, 2015.

[4] 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)*, pages 330–339, June 2019.

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

[6] CoinMarketCap Bitcoin Markets. https://coinmarketcap.com/currencies/bitcoin/#markets.

[7] Ernst & Young Inc — Third Report of the Monitor, March 2019. 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.

[8] Aaron Wood. QuadrigaCX Wallets Have Been Empty, Unused Since April 2018, Ernst and Young Finds, March 2019. https://cointelegraph.com/news/report-quadrigacx-wallets-have-been-empty-unused-since-april.

[9] Tom Elvis Jedusor. Mimblewimble, 2016. https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt.

[10] Beam project website. https://www.beam.mw/.

[11] Grin project website. https://grin-tech.org/.

[12] CoinMarketCap Beam Markets. https://coinmarketcap.com/currencies/bitcoin/#markets.

[13] CoinMarketCap Grin Markets. https://coinmarketcap.com/currencies/bitcoin/#markets.

[14] Grinscan website. https://grinscan.net/charts.

[15] Russell W. F. Lai and Viktoria Ronge and Tim Ruffing and Dominique Schröder and Sri Aravinda Krishnan Thyagarajan and Jiafan Wang. Omniring: Scaling up private payments without trusted setup. Cryptology ePrint Archive, Report 2019/580, 2019. https://eprint.iacr.org/2019/580.

[16] 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)*, pages 315–334, May 2018.

[17] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan Boneh. Provisions: Privacy-preserving proofs of solvency for bitcoin exchanges. *IACR Cryptology ePrint Archive*, 2015:1008, 2015.

[18] Jean-Jacques Quisquater, Myriam Quisquater, Muriel Quisquater, Michaël Quisquater, Louis Guillou, Marie Annick Guillou, Gaïd Guillou, Anna Guillou, Gwenolé Guillou, and Soazig Guillou. How to explain zero-knowledge protocols to your children. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 628–631, New York, NY, 1990. Springer New York.

[19] Kessler Gary C. An Overview of Cryptography. *Handbook on Local Area Networks*, September 1998. https://www.garykessler.net/library/crypto.html.

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

[21] Harish Natarajan, Solvej Karla Krause, and Helen Luskin Gradstein. Distributed Ledger Technology (DLT) and blockchain (English). *FinTech note; no. 1. Washington, D.C. : World Bank Group*, 2017. http://documents.worldbank.org/curated/en/177911513714062215/Distributed-Ledger-Technology-DLT-and-blockchain.

[22] Benedikt BÃijnz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short Proofs for Confidential Transactions and More. *Cryptology ePrint Archive, Report 2017/1066*, 2017. https://eprint.iacr.org/2017/1066.

[23] Monero website. https://getmonero.org/.

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

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

[26] Shen Noether and Adam Mackenzie. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[27] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[28] KZen Networks Bulletproofs github repository. https://github.com/KZen-networks/bulletproofs.

[29] Revelio simulation code. https://github.com/avras/revelio.

[30] Feng Bao, Robert H. Deng, and HuaFei Zhu. Variations of Diffie-Hellman problem. In *Information and Communications Security*, pages 301–312, 2003.