Revelio: A MimbleWimble Proof of Reserves Protocol

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

***Abstract*—We reveal Revelio, a new privacy-preserving proof of reserves protocol for Grin exchanges. By design, Revelio allows the detection of collusion between exchanges while hiding the identities of the outputs owned by the exchange in a larger anonymity set of outputs.**

***Index Terms*—Cryptocurrency, MimbleWimble, Grin, proof of reserves**

I. Introduction

Grin is a MimbleWimble-based cryptocurrency which became operational on January 15th, 2019 [1]. Despite being a relatively new project, Grin has been listed on several cryptocurrency exchanges [2]. Such exchanges enable their customers to acquire Grin coins without mining them. As customers do not control the private keys of the coins they hold on the exchange, it is possible for dishonest exchanges to sell more coins to their customers than the total reserves they hold. To alleviate such concerns, proof of reserves protocols have been proposed for Bitcoin [3]–[5] and Monero [6]. But the design of Grin prevents the usage of these previously proposed protocols for generating proofs of Grin reserves.

Our contribution: In this paper, we present Revelio, the first privacy-preserving proof of reserves protocol for Grin exchanges. The unspent outputs owned by an exchange are hidden in a larger anonymity set of unspent outputs. While the cryptographic primitives used in Revelio are not novel, we believe that the protocol construction itself is not obvious and will be of interest to the Grin community.

II. Revelio Proof of Reserves Protocol

Being an implementation of the MimbleWimble protocol, Grin does not have addresses [7], [8]. To transfer coins, the sender and receiver have to interactively build a transaction [9]. Coins are stored in *outputs* which contain a Pedersen commitment $C = kG + vH$ where k and v are scalars in the prime field $\mathbb{F}_n$ with n equal to the order of the group in the secp256k1 curve. Knowledge of both these scalars implies ownership of the output. The scalar v represents the amount stored in the output. The scalar k is chosen randomly to conceal the amount and is called the *blinding factor*. G is the base point of the secp256k1 curve and H is another point on the secp256k1 curve whose discrete logarithm with respect to G is not known. The output also has a range proof which proves that the value v stored in the commitment C is in the range $\{0, 1, \ldots, 2^{64} - 1\}$.

Like the proof of reserves protocols in Provisions [5] and MProve [6], the Revelio protocol will output a Pedersen commitment C_{assets} to an amount which is equal to the number of coins owned by the exchange. Given a Pedersen commitment $C_{\text{liabilities}}$ to the total liabilities of the exchange, it can prove solvency via a range proof which shows that the amount committed to in $C_{\text{assets}} - C_{\text{liabilities}}$ is non-negative.

Let $\mathcal{C}_{\text{unspent}}$ be the set of all unspent outputs on the Grin blockchain. Let $\mathcal{C}_{\text{own}}$ be the set of unspent outputs owned by the exchange. The exchange will choose a subset $\mathcal{C}_{\text{anon}}$ of $\mathcal{C}_{\text{unspent}}$ such that $\mathcal{C}_{\text{own}} \subseteq \mathcal{C}_{\text{anon}}$. The set $\mathcal{C}_{\text{anon}}$ represents the *anonymity set* of unspent outputs which contains the unspent outputs actually owned by the exchange. This anonymity set will be revealed in the Revelio protocol. But an observer will not be able to distinguish between members of $\mathcal{C}_{\text{anon}}$ which belong to $\mathcal{C}_{\text{own}}$ from those which do not.

If an exchange does not care about revealing the identity of the unspent outputs it owns, then there is no need for the Revelio protocol. The exchange can simply reveal $\mathcal{C}_{\text{own}}$ and generate C_{assets} as

$$C_{\text{assets}} = \sum_{C \in \mathcal{C}_{\text{own}}} C. \tag{1}$$

It can then prove knowledge of scalars k and v such that $C_{\text{assets}} = kG + vH$ by giving a proof of knowledge of the representation of C_{assets} [10]. We compare the performance of Revelio to this simple non-private protocol in Section II-E.

Any proof of reserves protocol for Grin should satisfy the following requirements:

- *Inflation resistance:* An exchange should not be able to generate a commitment to an amount which is greater than the total number of coins it owns.
- *Privacy of amounts:* For an output $C_i \in \mathcal{C}_{\text{own}}$ such that $C_i = k_i G + v_i H$, the amount v_i should not be revealed. This is to conform with the privacy-focused design of Grin. So the point $k_i G$ 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.
- *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.

The Revelio protocol satisfies the above requirements. Additionally, Revelio provides *output privacy* in the sense that an observer who is given the set of outputs $\mathcal{C}_{\text{anon}}$ is not able to identify which outputs belong to $\mathcal{C}_{\text{own}}$.

A. Proving Statements About Discrete Logarithms

In this section, we recall constructions of non-interactive zero-knowledge (NIZK) proofs of knowledge for three statements involving discrete logarithms using the notation proposed by Camenisch and Stadler [10], [11]. We use additive notation to represent the group operation for consistency with the protocol construction presented in the next section.

Let $\mathcal{G}$ be a cyclic group of prime order n. Let G, G', H each be generators of the group $\mathcal{G}$. Suppose that computing discrete logarithms of random group elements in $\mathcal{G}$ is infeasible with respect to any of these generators. Also suppose that the discrete logarithms of each of these three generators with respect to the other two are not known. Let $\mathcal{H} : \{0,1\}^* \mapsto \mathbb{Z}_n$ be a cryptographic hash function which is modelled as a random oracle. For notational convenience, we will use a comma to represent the concatenation operator $\|$ in the arguments of $\mathcal{H}$, i.e. we will write $\mathcal{H}(X, Y, Z)$ instead of $\mathcal{H}(X\|Y\|Z)$.

Definition 1. *A pair of scalars $(c, s) \in \mathbb{Z}_n^2$ is a NIZK proof of knowledge (PoK) of the discrete logarithm of an element $X \in \mathcal{G}$ with respect to a generator G if they satisfy*

$$c = \mathcal{H}\left(G, X, sG + cX\right).$$

Such a pair will be denoted by $\mathit{PoK}\{\alpha \mid X = \alpha G\}$.

This proof is a Schnorr signature which can be generated by choosing r randomly from $\mathbb{Z}_n$, calculating $c = \mathcal{H}(G, X, rG)$, and setting $s = r - c\alpha \bmod n$.

By the *representation* of an element $X \in \mathcal{G}$ with respect to generators G, H, we mean the scalars $(\alpha, \beta) \in \mathbb{Z}_n^2$ such that $X = \alpha G + \beta H$.

Definition 2. *A triple of scalars $(c, s_1, s_2) \in \mathbb{Z}_n^3$ is a NIZK proof of the knowledge and equality of the representations of elements $X, Y \in \mathcal{G}$ with respect to the generator pairs G, H and G', H respectively, if they satisfy*

$$c = \mathcal{H}\left(S, s_1G + s_2H + cX, s_1G' + s_2H + cY\right)$$

where $S = G\|G'\|H\|X\|Y$. Such a triple will be denoted by

$$\mathit{PoK}\{(\alpha, \beta) \mid X = \alpha G + \beta H \ \wedge \ Y = \alpha G' + \beta H\}.$$

A prover with knowledge of α, β can generate such a proof as follows:

- She chooses r_1, r_2 randomly from $\mathbb{Z}_n$ and calculates
$$c = \mathcal{H}\left(S, r_1G + r_2H, r_1G' + r_2H\right).$$
- She sets s_1 and s_2 as
$$\begin{aligned} s_1 &= r_1 - c\alpha, \\ s_2 &= r_2 - c\beta. \end{aligned}$$

In the Revelio protocol, we requires proofs of statements which are disjunctions of the two types of statements given in the above definitions. The main idea behind the following construction was first proposed by Cramer et al. [12].

Definition 3. *A 5-tuple of scalars $(c_1, c_2, s_1, s_2, s_3) \in \mathbb{Z}_n^5$ is a NIZK proof of* ***either***

(i) the knowledge and equality of the representations of elements $X, Y \in \mathcal{G}$ with respect to the generator pairs G, H and G', H respectively, ***or***

(ii) knowledge of the discrete logarithm of the element $Y \in \mathcal{G}$ with respect to the generator G',

if they satisfy

$$c_1 + c_2 = \mathcal{H}\left(S, V_1, V_2, V_3\right)$$

where $S = G\|G'\|H\|X\|Y$ and

$$\begin{aligned} V_1 &= s_1G + s_2H + c_1X, \\ V_2 &= s_1G' + s_2H + c_1Y, \\ V_3 &= s_3G' + c_2Y. \end{aligned} \tag{2}$$

Such a 5-tuple will be denoted by

$$\mathit{PoK}\{(\alpha, \beta, \gamma) \mid (X = \alpha G + \beta H \ \wedge \ Y = \alpha G' + \beta H) \vee (Y = \gamma G')\}.$$

Suppose the prover knows γ such that $Y = \gamma G'$. Then she can generate a proof as follows:

- She chooses r_3, c_1, s_1, s_2 randomly from $\mathbb{Z}_n$ and calculates c_2 as
$$c_2 = \mathcal{H}(S, V_1, V_2, r_3G') - c_1, \tag{3}$$
where $S = G\|G'\|H\|X\|Y$, $V_1 = s_1G + s_2H + c_1X$, and $V_2 = s_1G' + s_2H + c_1Y$.
- She sets s_3 as
$$s_3 = r_3 - c_2\gamma. \tag{4}$$

Note the similarity with the proof generation for the statement in Definition 1.

Now consider the case when the prover knows α, β such that $X = \alpha G + \beta H$ and $Y = \alpha G' + \beta H$. She can generate a proof as follows:

- She chooses r_1, r_2, c_2, s_3 randomly from $\mathbb{Z}_n$ and calculates c_1 as
$$c_1 = \mathcal{H}(S, r_1G + r_2H, r_1G' + r_2H, V_3) - c_2, \tag{5}$$
where $S = G\|G'\|H\|X\|Y$ and $V_3 = s_3G' + c_2Y$.
- She sets s_1 and s_2 as
$$\begin{aligned} s_1 &= r_1 - c_1\alpha, \\ s_2 &= r_2 - c_1\beta. \end{aligned} \tag{6}$$

This is similar to the proof generation for the statement in Definition 2.

B. Proof Generation

Let G be the base point of the secp256k1 elliptic curve. Let G' be another generator of the secp256k1 curve whose discrete logarithms with respect to G and H are not known. Let $\mathcal{H}$ be equal to the SHA256 hash function.

The Revelio proof of reserves protocol proceeds as follows:

1. The exchange chooses a long-term secret key k_{exch} uniformly from $\mathbb{Z}_n$. This key must remain the same in all the Revelio proofs generated by the exchange, i.e. this key is chosen when the exchange generates a Revelio proof for the first time and subsequently remains unchanged.
2. The exchange chooses a list of unspent outputs $\mathcal{C}_{\text{anon}} = (C_1, C_2, \ldots, C_N)$ from the Grin blockchain such that it owns the outputs in a subset $\mathcal{C}_{\text{own}}$ of $\mathcal{C}_{\text{anon}}$. The list $\mathcal{C}_{\text{anon}}$ is made public by the exchange. Ownership of an output $C_i \in \mathcal{C}_{\text{own}}$ is equivalent to knowledge of scalars k_i and v_i such that $C_i = k_i G + v_i H$. For $C_i \in \mathcal{C}_{\text{anon}} \setminus \mathcal{C}_{\text{own}}$, the exchange may know the value v_i if it was the sender in the transaction which created C_i. But it does not know the blinding factor k_i for such an output.
3. For each $C_i \in \mathcal{C}_{\text{anon}}$ such that $C_i = k_i G + v_i H$, the exchange generates a curve point I_i as
$$I_i = \begin{cases} k_i G' + v_i H & \text{if } C_i \in \mathcal{C}_{\text{own}}, \\ y_i G' & \text{if } C_i \notin \mathcal{C}_{\text{own}}, \end{cases} \tag{7}$$
where $y_i = \mathcal{H}(k_{\text{exch}}, C_i)$. The points $(I_1, I_2, \ldots, I_N)$ are published by the exchange. Note that the I_is are a deterministic function of the respective C_is (for a fixed long-term secret key k_{exch}).
4. For each $i = 1, 2, \ldots, N$, the exchange uses the method described in Section II-A to generate a NIZK PoK $\sigma_i = (c_1^i, c_2^i, s_1^i, s_2^i, s_3^i)$ of the form
$$\text{PoK}\{(\alpha, \beta, \gamma) \mid (C_i = \alpha G + \beta H \ \wedge \ I_i = \alpha G' + \beta H) \vee (I_i = \gamma G')\}.$$
The proofs $(\sigma_1, \sigma_2, \ldots, \sigma_N)$ are published by the exchange.
5. The exchange claims that the commitment C_{assets} given by
$$C_{\text{assets}} = \sum_{i=1}^{N} I_i \tag{8}$$
is a Pedersen commitment of the form $x_{\text{tot}} G' + v_{\text{tot}} H$ where v_{tot} is the total amount of Grin it owns.

Note that the blinding factor x_{tot} is multiplying G' and not G in the commitment C_{assets}. The proof of solvency needs to take this into account by generating the $C_{\text{liabilities}}$ commitment to have the form $yG' + vH$.

The intuition behind the protocol construction is as follows. The NIZK PoK σ_i proves that if the exchange does not own the output C_i then I_i is a commitment to the zero amount. Furthermore, it proves that in case the exchange does own the output C_i it can generate I_i as a commitment *only* to the amount v_i which is committed to by C_i. These two properties of I_i force the C_{assets} calculated in (8) to be a commitment to an amount which is *at most* equal to the total amount of Grin owned by the exchange. C_{assets} is not necessarily a commitment to an amount equal to the total amount owned by the exchange because the exchange can choose I_i to be a commitment to the zero amount in spite of owning C_i.

The above argument does not clarify the need for introducing the generator G' or the need for setting y_i equal to $\mathcal{H}(k_{\text{exch}}, C_i)$. The former is needed to detect collusion between exchanges while the latter is needed to prevent identification of outputs belonging to the exchange.

Let us consider the need for G' first. One can get the same guarantees regarding C_{assets} by defining I_i as follows for $C_i = k_i G + v_i H$.
$$I_i = \begin{cases} x_i G + v_i H & \text{if } C_i \in \mathcal{C}_{\text{own}}, \\ y_i G & \text{if } C_i \notin \mathcal{C}_{\text{own}}, \end{cases} \tag{9}$$
where the x_is are chosen uniformly from $\mathbb{Z}_n$ and $y_i \in \mathbb{Z}_n$. Note that for $C_i \in \mathcal{C}_{\text{own}}$ the blinding factor in I_i is different from the blinding factor in C_i but the amounts in both commitments are the same. One can prove that I_i has this structure by giving a NIZK PoK of the form
$$\text{PoK}\{(\alpha, \beta, \gamma, \delta) \mid (C_i = \alpha G + \beta H \ \wedge \ I_i = \delta G + \beta H) \vee (I_i = \gamma G)\}.$$

While this definition of I_i guarantees that the C_{assets} calculated in (8) has the right properties, it does not prevent collusion between exchanges. Two exchanges could share an output C_i to generate their respective asset proofs without being detected as the blinding factors x_i in (9) can be chosen freely. By forcing I_i to have the structure given in (7), we are ensuring that I_i is a deterministic function of C_i for $C_i \in \mathcal{C}_{\text{own}}$. Consequently, it plays the role of a *key image* of C_i which enables detection of collusion between exchanges. In case the same I_i appears in the proofs of reserves of two different exchanges, collusion between them is revealed. This structure of I_i is the main innovation in the Revelio protocol.

Finally, the reason for setting $y_i = \mathcal{H}(k_{\text{exch}}, C_i)$ is to make I_i a deterministic function of C_i even when $C_i \notin \mathcal{C}_{\text{own}}$, without revealing the nonmembership of C_i in $\mathcal{C}_{\text{own}}$. Suppose I_i was not a deterministic function of C_i for $C_i \notin \mathcal{C}_{\text{own}}$. For example, the y_is could simply be chosen uniformly from $\mathbb{Z}_n$ every time an exchange generated a Revelio proof (the y_i values in the current Revelio proof will be independent of the y_i values in the previous Revelio proofs). It is realistic to assume that a proof of reserves protocol will need to be executed multiple times by an exchange. In this scenario, the I_i points corresponding to $C_i \in C_{\text{anon}} \setminus C_{\text{own}}$ will keep changing in each Revelio proof while the I_i points corresponding to $C_i \in C_{\text{own}}$ will remain the same. Outputs C_i which appear in multiple Revelio proofs of an exchange with the same I_i will be identified as outputs belonging to C_{own}. Outputs C_i which appear in multiple Revelio proofs with different I_i will be identified as outputs not belonging to C_{own}.[1] Of course, the method of making the y_is a deterministic function of C_i using

[1] Our earlier proposal had this flaw. One of the anonymous reviewers of our submission identified this flaw and suggested the fix.

$\mathcal{H}$ and k_{exch} is largely a matter of convenience. An exchange could also implement such a function using a lookup table where the y_is are chosen uniformly and independently from $\mathbb{Z}_n$ once and reused in all subsequent Revelio proofs.

C. Proof Verification

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

- The list $\mathcal{C}_{\text{anon}} = (C_1, C_2, \ldots, C_N)$.
- The key image commitment list $(I_1, I_2, \ldots, I_N)$.
- The proofs $\sigma_i = (c_1^i, c_2^i, s_1^i, s_2^i, s_3^i)$ for $i = 1, 2, \ldots, N$.

Verification involves the following operations:

1. The verifier checks that the list $\mathcal{C}_{\text{anon}}$ consists of only unspent outputs.
2. For each $i = 1, 2, \ldots, N$, the verifier verifies the proof σ_i using the pair (C_i, I_i).
3. The verifier also checks that none of the key image commitments I_i published by the exchange appear in the proofs of reserves published by other exchanges. If a key image commitment is common to the proofs published by two different exchanges, collusion is declared.

D. Security Properties

The security of the Revelio protocol against *asset inflation* by an exchange and *undetected collusion* between exchanges follows from the unforgeability of the NIZK proofs σ_i.

- In order to generate a C_{assets} commitment to an amount which is greater than the total amount it owns, an exchange would have to create an I_i which is either a commitment to the amount in C_i when it does not own C_i or a commitment to an amount larger than the amount in C_i. But this would be a forgery of the NIZK proof σ_i. So a probabilistic polynomial time (PPT) exchange can possibly achieve asset inflation with only a negligible probability of success.
- When I_i is a commitment to the amount in C_i, the NIZK proof σ_i guarantees that I_i is a deterministic function of C_i except with negligible probability. This ensures that collusion between exchanges (via output sharing) is detected.

Revelio also provides *output privacy* in the sense that a PPT adversary cannot distinguish between those outputs in the anonymity set which belong to the exchange and those which do not. To make this notion precise, consider the following experiment (which we call `OutputPriv`).

Let the anonymity set $\mathcal{C}_{\text{anon}}$ consist of a single unspent output $C_1 = k_1 G + v_1 H$ owned by the exchange. Let $\mathcal{A}$ be a PPT adversary whose running time is bounded by a polynomial in the security parameter λ.

1. The exchange chooses a bit b uniformly from $\{0, 1\}$ and a long-term secret key k_{exch} uniformly from $\mathbb{Z}_n$.
2. The adversary specifies a polynomial $p(\lambda)$ number of proofs to be generated by the exchange.
3. If $b = 0$, the exchange generates a list of $p(\lambda)$ Revelio proofs (C_1, I_1, σ_1^i) where $i = 1, 2, \ldots, p(\lambda)$, $I_1 = y_1 G$, and $y_1 = \mathcal{H}(k_{\text{exch}}, C_1)$.
4. If $b = 1$, the exchange generates a list of $p(\lambda)$ Revelio proofs (C_1, I_1, σ_1^i) where $i = 1, 2, \ldots, p(\lambda)$ and I_1 has the same representation as C_1 with respect to generators G' and H, i.e. $I_1 = k_1 G' + v_1 H$.
5. Given the proofs (C_1, I_1, σ_1^i), $i = 1, 2, \ldots, p(\lambda)$, and the amount v_1, an adversary $\mathcal{A}$ outputs a bit b', i.e.
$$b' = \mathcal{A}\left(C_1, I_1, \sigma_1^1, \sigma_1^2, \ldots, \sigma_1^{p(\lambda)}, v_1\right). \tag{10}$$
6. The adversary succeeds if $b' = b$. Otherwise, it fails.

We give the amount v_1 as an input to $\mathcal{A}$ to model the pessimistic scenario of the adversary having knowledge of the amount in C_1 (due to being the sender in the transaction which created C_1). Not knowing v_1 only makes it harder for $\mathcal{A}$ to succeed. If $\mathcal{A}$ succeeds in estimating the bit b with a probability which is non-negligibly better than $\frac{1}{2}$, then the proof of reserves protocol leaks information about output ownership. As the I_is and σ_is corresponding to different outputs are independent, an adversary who succeeds in detecting output ownership when the anonymity set has a size larger than one must be able to succeed in the above experiment.

Definition 4. *The Revelio protocol provides output privacy if every PPT adversary $\mathcal{A}$ succeeds in the `OutputPriv` experiment with a probability which is negligibly close to $\frac{1}{2}$.*

If we model the hash function $\mathcal{H}$ as a random oracle, each $\sigma_1^i = (c_1^i, c_2^i, s_1^i, s_2^i, s_3^i)$ is uniformly distributed on $\mathbb{Z}_n^5$ irrespective of the bit b (see the calculations in (3), (4), (5), and (6)). So an adversary can only hope to estimate b via (C_1, I_1, v_1). The adversary can obtain the point $k_1 G$ by calculating $C_1 - v_1 H$. It can also calculate $I_1 - v_1 H$ which is given by

$$I_1 - v_1 H = \begin{cases} y_1 G' - v_1 H & \text{if } b = 0, \\ k_1 G' & \text{if } b = 1. \end{cases} \tag{11}$$

We make the following observations:

- For $b = 1$, the triple $(k_1 G, G', I_1 - v_1 H)$ is a Diffie-Hellman triple. To see this, let $\alpha = k_1$ and $G' = \beta G$ for some $\beta \in \mathbb{Z}_n$, and observe that $I_1 - v_1 H = k_1 G' = \alpha\beta G$. Thus
$$(k_1 G, G', I_1 - v_1 H) = (\alpha G, \beta G, \alpha\beta G). \tag{12}$$
- For $b = 0$, $I_1 = y_1 G$ is uniformly distributed over the group $\mathcal{G}$ as $y_1 = \mathcal{H}(k_{\text{exch}}, C_1)$ and $\mathcal{H}$ is a random oracle. This implies that $I_1 - v_1 H$ is uniformly distributed over the group $\mathcal{G}$. Thus
$$(k_1 G, G', I_1 - v_1 H) = (\alpha G, \beta G, \gamma G), \tag{13}$$
where γ is uniformly distributed over $\mathbb{Z}_n$ and independent of α and β.

So the task of estimating b is as hard as the task of identifying Diffie-Hellman triples. As long as the decisional Diffie-Hellman assumption holds in the group $\mathcal{G}$, no PPT adversary can estimate b with a probability which is non-negligibly better than $\frac{1}{2}$. This argument essentially proves the following

TABLE I
PROOF GENERATION AND VERIFICATION PERFORMANCE

$\mathcal{C}_{\text{anon}}$ Size	$\mathcal{C}_{\text{own}}$ Size	Revelio Proof Size	Revelio Gen. Time	Revelio Ver. Time	Simple Proof Size	Simple Gen. Time	Simple Ver. Time
100	25	0.02 MB	1.13 s	1.14 s	0.92 KB	21.96 ms	22.00 ms
100	50	0.02 MB	1.14 s	1.15 s	1.74 KB	22.12 ms	22.17 ms
100	75	0.02 MB	1.15 s	1.16 s	2.57 KB	22.38 ms	22.40 ms
1000	250	0.22 MB	11.98 s	11.98 s	8.34 KB	23.52 ms	23.56 ms
1000	500	0.22 MB	11.80 s	11.86 s	0.01 MB	25.42 ms	25.42 ms
1000	750	0.22 MB	11.73 s	11.84 s	0.02 MB	27.46 ms	27.46 ms
10000	2500	2.26 MB	109.57 s	110.23 s	0.08 MB	41.36 ms	41.28 ms
10000	5000	2.26 MB	109.87 s	110.78 s	0.16 MB	60.36 ms	60.18 ms
10000	7500	2.26 MB	109.57 s	110.46 s	0.24 MB	79.10 ms	78.73 ms

theorem. We defer the formal proof to an extended version of this paper.

Theorem 1. *The Revelio protocol provides output privacy in the random oracle model under the decisional Diffie-Hellman assumption.*

E. Performance

To the best of our knowledge, Revelio is the first proof of reserves protocol for Grin exchanges which provides output privacy. So there is no existing benchmark which can be used to evaluate the relative performance of Revelio. Instead, we compare Revelio to the simple non-private protocol described in (1). The simulation code was implemented in Rust using the rust-secp256k1-zkp library [13]. It is available at [14].

The performance of the Revelio proof generation and verification algorithms is given in Table I for anonymity list $\mathcal{C}_{\text{anon}}$ having sizes 100, 1000, and 10000. For each case, the percentage of known addresses is either 25%, 50%, or 75%. The table also shows the performance of the simple non-private protocol as a function of $\mathcal{C}_{\text{own}}$ size (the $\mathcal{C}_{\text{anon}}$ parameter is irrelevant for this protocol). The execution times were measured on a single core of an Intel i7-7700 3.6 GHz CPU. The Revelio protocol is orders of magnitude slower and its proof size an order of magnitude larger compared to the simple non-private protocol. Nevertheless the proof sizes and running times of Revelio are practical and the higher values are justified by the privacy it provides. As the NIZK proof generation and verification for different outputs can proceed in parallel, running times can be reduced by parallel execution.

III. CONCLUSION

We have presented the design of the first privacy-preserving collusion-resistant proof of reserves protocol for Grin exchanges. Our simulations show that the proof generation and verification times are practical. Here are two potential areas for improvement.

- The collusion-resistance property of Revelio works only if all the exchanges generate their proofs using the same blockchain state. If an exchange generates a Revelio proof and then transfers some coins to another exchange before the latter generates its Revelio proof, then these two exchanges end up effectively sharing some coins. Enforcing simultaneous proof generation is an interesting direction for future research.
- Revelio provides output privacy in the sense that a PPT adversary cannot tell where an output in C_{anon} belongs to C_{own} or $C_{\text{anon}} \setminus C_{\text{own}}$. But this level of privacy is far from perfect as the adversary knows that the exchange owns some outputs in C_{anon}. Since the proof sizes and generation/verification times in Revelio increase linearly with the size of C_{anon}, simply setting C_{anon} to be equal to the whole unspent output set C_{unspent} is not a scalable strategy. Development of more efficient proof of reserves protocols which will enable perfect output privacy is another interesting direction for future research.

ACKNOWLEDGEMENTS

We would like to thank the anonymous reviewers for their comments which have improved this paper and informed our discussion of the potential areas for improvement in the concluding section. Special thanks to the reviewer who found a flaw in our earlier design and suggested a fix for it. We also acknowledge the support of the Bharti Centre for Communication at IIT Bombay.

REFERENCES

[1] Grin project website. [Online]. Available: https://grin-tech.org/
[2] Exchanges information sharing for Grin. [Online]. Available: www.grin-forum.org/t/exchanges-information-sharing-for-grin/2343
[3] Z. Wilcox, "Proving your Bitcoin reserves," Bitcoin Talk Forum Post, May 2014. [Online]. Available: https://bitcointalk.org/index.php?topic=595180.0
[4] C. Decker, J. Guthrie, J. Seidel, and R. Wattenhofer, "Making bitcoin exchanges transparent," in *20th European Symposium on Research in Computer Security (ESORICS)*, 2015, pp. 561–576.
[5] G. G. Dagher, B. Bünz, J. Bonneau, J. Clark, and D. Boneh, "Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (ACM CCS)*, New York, NY, USA, 2015, pp. 720–731.
[6] A. Dutta and S. Vijayakumaran, "MProve: A proof of reserves protocol for Monero exchanges," Cryptology ePrint Archive, Report 2018/1210, 2018, https://eprint.iacr.org/2018/1210.
[7] T. E. Jedusor, "Mimblewimble," 2016. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.txt
[8] A. Poelstra, "Mimblewimble," 2016. [Online]. Available: https://download.wpsoftware.net/bitcoin/wizardry/mimblewimble.pdf
[9] Introduction to MimbleWimble and Grin. [Online]. Available: https://github.com/mimblewimble/grin/blob/master/doc/intro.md
[10] J. Camenisch, "Group signature schemes and payment systems based on the discrete logarithm problem," Ph.D. dissertation, ETH Zurich, 1998.
[11] J. Camenisch and M. Stadler, "Proof systems for general statements about discrete logarithms," Dept. of Computer Science, ETH Zürich, Tech. Rep. 260, Mar 1997.
[12] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology — CRYPTO '94*, 1994, pp. 174–187.
[13] Grin rust-secp256k1-zkp github repository. [Online]. Available: https://github.com/mimblewimble/secp256k1-zkp/
[14] Revelio simulation code. [Online]. Available: https://github.com/avras/revelio