Efficient Proof of Reserves for Cryptocurrency Exchanges

Dual Degree Project - Phase I

Suyash Bagad

Guide: Prof. Saravanan Vijayakumaran

Department of Electrical Engineering,
Indian Institute of Technology, Bombay

21st October 2019

Pedersen Commitment Scheme

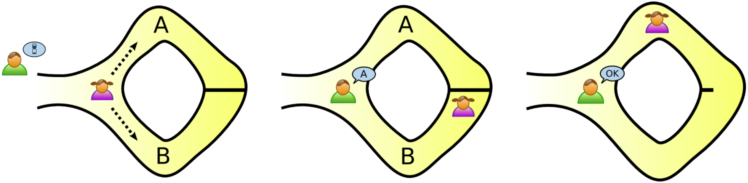
Figure: Digital equivalent of a sealed box

- Given a group $\mathbb{G}$ of prime order q and $g, h \xleftarrow{\$} \mathbb{G}$, define $\mathsf{pk} = (g, h)$
- A commitment to a message m, with $r \xleftarrow{\$} \mathbb{Z}_q$, is defined as

$$\mathsf{Com}_{\mathsf{pk}}(m) = g^r \cdot h^m$$

- Secret (m, r) forms an *opening* to the above Pedersen commitment
- Perfectly Hiding: Given a Pedersen commitment P, no adversary can determine (m, r)
- Computationally Binding: Assuming the discrete-log problem is hard, it is infeasible to output two correct openings $(m, r), (m', r')$ with $m \neq m'$ to P

Zero Knowledge Proofs of Knowledge



Courtesy: `https://en.wikipedia.org/wiki/Zero-knowledge_proof`

- Proofs that yield nothing beyond the validity of an assertion
- An interactive proof system is a ZKPoK if it satisfies:
  - Completeness: Honest prover convinces honest verifier
  - Zero-Knowledge: Malicious verifiers learn nothing more than statement validity
  - Soundness: Dishonest prover cannot convince a verifier

Crypto Exchanges

Figure: A crypto exchange is like a virtual bank for cryptocurrencies in exchange with fiat currencies

- ✓ Enable anyone to own cryptocurrencies without mining them
- ✓ Free the customer from storing secret information (private keys)
- ✓ Provide custodial wallets and trading services to customers
- ✗ Loss of customer money in cases of hack (MtGox, '14)
- ✗ Exit scam or internal fraud by exchange owners (QuadrigaCX, '18)

Can we design a cryptographic system to avoid or predict occurence of such undesirable cases?

Proof of Solvency

- ✓ Proves that crypto reserves of an exchange exceed its liabilities
- ✓ Prevents exchanges from hiding loss of funds due to cyberattacks
- ✓ Disallows them to sell crypto assets without actually owning them
- ✗ Not a fool-proof method to counter hacks and exit scams

- An exchange generates two Pedersen commitments:

$$C_{\text{res}} = g^{r_{\text{res}}} h^{a_{\text{res}}} \qquad C_{\text{liab}} = g^{r_{\text{liab}}} h^{a_{\text{liab}}}$$

To the total reserves amount a_{res} To the total liability amount a_{liab}

- An exchange is solvent if it proves that $C_{\text{res}} C_{\text{liab}}^{-1}$ is a commitment to a non-negative amount

We focus on the design of proof of reserves.

Our contribution

- We present Revelio+ , an efficient and privacy-preserving proof of reserves for Mimblewimble-based cryptocurrencies.[1]
- We alleviate the drawbacks of Revelio, the first proof of reserves for Mimblewimble cryptocurrencies

	Revelio+	Revelio
Proof size	$\mathcal{O}(\log(sn)+s)$	$\mathcal{O}(n)$
Scalability	✓	✗
Blockchain state	✓	✗
Output privacy	✓	✓
Non-collusion	✓	✓
Inflation resistance	✓	✓

[1]Work submitted to *Finacial Cryptography 2020*.

Mimblewimble

- MimbleWimble is a blockchain-based ledger with strong privacy and confidentiality guarantees
- Transactions are scriptless and consist only of *inputs*, *outputs* and *excess* $\implies$ Scalability!
- A MimbleWimble output is a Pedersen commitment of the form $C = g^r h^a$ where $r, a \in \mathbb{Z}_q$ are blinding factor and amount resp.
- Knowledge of r, a implies ownership of output $C = g^r h^a$
- Grin blockchain consists of all unspent outputs

$$\boxed{C_1},\ C_2,\ \boxed{C_3},\ C_4,\ C_5 \ldots,\ C_{n-3},\ C_{n-2},\ \boxed{C_{n-1}},\ \boxed{C_n}$$

Revelio

- Let UTXO set be $\mathcal{C}_{\text{unspent}}$, set of exchange-owned outputs be $\mathcal{C}_{\text{own}}$
- Exchange reveals anonymity set $\mathcal{C}_{\text{anon}} = \{C_1, C_2, \ldots, C_n\}$ so that

$$\mathcal{C}_{\text{own}} \subset \mathcal{C}_{\text{anon}} \subset \mathcal{C}_{\text{unspent}}$$

- For each $C_i = g^{r_i} h^{a_i} \in \mathcal{C}_{\text{anon}}$, exchange defines key-images I_i

$$I_i = \begin{cases} (g')^{x_i} h^{a_i} & \text{if } C_i \in \mathcal{C}_{\text{own}} \\ (g')^{y_i} & \text{if } C_i \notin \mathcal{C}_{\text{own}} \end{cases}$$

  where $x_i \xleftarrow{\$} \mathbb{Z}_q$ and $y_i = \mathcal{H}(k_{\text{exch}}, C_i)$
- For each $C_i \in C_{\text{anon}}$, exchange gives ZKPoK of the form

$$\sigma_i = \text{PoK}\{(\alpha, \beta, \gamma, \delta) \mid (C_i = g^{\alpha} h^{\beta} \wedge I_i = (g')^{\delta} h^{\beta}) \vee (I_i = (g')^{\gamma})\}$$

- Setting $C_{\text{res}} = \prod_{i=1}^{n} I_i$, a Revelio proof is of the form

$$\Pi_{\text{Rev}} = \{\mathcal{C}_{\text{anon}}, I_1, \ldots, I_n, \sigma_1, \ldots, \sigma_n\}$$

Courtesy: `https://hackmd.io/@sarva/revelio/`

Revelio

- Let UTXO set be $\mathcal{C}_{\text{unspent}}$, set of exchange-owned outputs be $\mathcal{C}_{\text{own}}$
- Exchange reveals anonymity set $\mathcal{C}_{\text{anon}} = \{C_1, C_2, \ldots, C_n\}$ so that

$$\mathcal{C}_{\text{own}} \subset \mathcal{C}_{\text{anon}} \subset \mathcal{C}_{\text{unspent}}$$

- For each $C_i = g^{r_i} h^{a_i} \in \mathcal{C}_{\text{anon}}$, exchange defines key-images I_i

$$I_i = \begin{cases} \boxed{(g')}^{x_i} h^{a_i} & \text{if } C_i \in \mathcal{C}_{\text{own}} \\ (g')^{y_i} & \text{if } C_i \notin \mathcal{C}_{\text{own}} \end{cases}$$

  where $x_i \xleftarrow{\$} \mathbb{Z}_q$ and $y_i = \mathcal{H}(k_{\text{exch}}, C_i)$
- For each $C_i \in C_{\text{anon}}$, exchange gives ZKPoK of the form

$$\sigma_i = \text{PoK}\{(\alpha, \beta, \gamma, \delta) \mid (C_i = g^\alpha h^\beta \wedge I_i = (g')^\delta h^\beta) \vee (I_i = (g')^\gamma)\}$$

- Setting $C_{\text{res}} = \prod_{i=1}^{n} I_i$, a Revelio proof is of the form

$$\Pi_{\text{Rev}} = \{\mathcal{C}_{\text{anon}}, I_1, \ldots, I_n, \sigma_1, \ldots, \sigma_n\}$$

Revelio

- Let UTXO set be $\mathcal{C}_{\text{unspent}}$, set of exchange-owned outputs be $\mathcal{C}_{\text{own}}$
- Exchange reveals anonymity set $\mathcal{C}_{\text{anon}} = \{C_1, C_2, \dots, C_n\}$ so that

$$\mathcal{C}_{\text{own}} \subset \mathcal{C}_{\text{anon}} \subset \mathcal{C}_{\text{unspent}}$$

- For each $C_i = g^{r_i} h^{a_i} \in \mathcal{C}_{\text{anon}}$, exchange defines key-images I_i

$$I_i = \begin{cases} (g')^{x_i} h^{a_i} & \text{if } C_i \in \mathcal{C}_{\text{own}} \\ (g')^{y_i} & \text{if } C_i \notin \mathcal{C}_{\text{own}} \end{cases}$$

  where $x_i \xleftarrow{\$} \mathbb{Z}_q$ and $\boxed{y_i = \mathcal{H}(k_{\text{exch}}, C_i)}$
- For each $C_i \in C_{\text{anon}}$, exchange gives ZKPoK of the form

$$\sigma_i = \text{PoK}\{(\alpha, \beta, \gamma, \delta) \mid (C_i = g^{\alpha} h^{\beta} \wedge I_i = (g')^{\delta} h^{\beta}) \vee (I_i = (g')^{\gamma})\}$$

- Setting $C_{\text{res}} = \prod_{i=1}^{n} I_i$, a Revelio proof is of the form

$$\Pi_{\text{Rev}} = \{\mathcal{C}_{\text{anon}}, I_1, \dots, I_n, \sigma_1, \dots, \sigma_n\}$$

Revelio - The Good and The Bad

- ✓ Exchange cannot inflate reserves
- ✓ Multiple exchanges cannot collude by sharing UTXOs
- ✓ Output-privacy of an exchange is conserved
- ✗ Collusion can be detected *only* if all the exchanges generate proofs from same blockchain state (i.e same $\mathcal{C}_{\text{unspent}}$)
- ✗ Proof sizes are linear in anonymity set size
- ✗ An adversary succeeds in guessing an exchange-owned output is

$$\frac{|\mathcal{C}_{\text{own}}|}{|\mathcal{C}_{\text{anon}}|}$$

For maximum privacy of the outputs, exchanges would want to enlarge the set $\mathcal{C}_{\text{anon}}$ to $\mathcal{C}_{\text{unspent}}$. But the linearly growing proof sizes make it impractical.

Motivating Revelio+

- Let the set of unspent outputs at block height t be

$$\{C_1, \underbrace{\boxed{C_2}}_{i_1}, \underbrace{\boxed{C_3}}_{i_2}, C_4, \ldots, \underbrace{\boxed{C_{n-2}}}_{i_{s-1}}, C_{n-1}, \underbrace{\boxed{C_n}}_{i_s}\}$$

In the order of appearance on the blockchain

Motivating Revelio+

- Let the set of unspent outputs at block height t be

$$\{C_1, \underbrace{\boxed{C_2}}_{i_1}, \underbrace{\boxed{C_3}}_{i_2}, C_4, \ldots, \underbrace{\boxed{C_{n-2}}}_{i_{s-1}}, C_{n-1}, \underbrace{\boxed{C_n}}_{i_s}\}$$

- Let the exchange-owned outputs be $(C_{i_1}, \ldots, C_{i_s})$ such that $C_{i_j} = g^{r_j} h^{a_j}$ for all $j \in [s]$

In the order of appearance on the blockchain

Motivating Revelio+

- Let the set of unspent outputs at block height t be

$$\{C_1, \underbrace{\boxed{C_2}}_{i_1}, \underbrace{\boxed{C_3}}_{i_2}, C_4, \ldots, \underbrace{\boxed{C_{n-2}}}_{i_{s-1}}, C_{n-1}, \underbrace{\boxed{C_n}}_{i_s}\}$$

- Let the exchange-owned outputs be $(C_{i_1}, \ldots, C_{i_s})$ such that $C_{i_j} = g^{r_j} h^{a_j}$ for all $j \in [s]$
- The corresponding key-images be defined as

$$I_j = g_t^{r_j} h^{a_j}$$

In the order of appearance on the blockchain

Motivating Revelio+

- Let the set of unspent outputs at block height t be

$$\{C_1, \underbrace{\boxed{C_2}}_{i_1}, \underbrace{\boxed{C_3}}_{i_2}, C_4, \ldots, \underbrace{\boxed{C_{n-2}}}_{i_{s-1}}, C_{n-1}, \underbrace{\boxed{C_n}}_{i_s}\}$$

- Let the exchange-owned outputs be $(C_{i_1}, \ldots, C_{i_s})$ such that $C_{i_j} = g^{r_j} h^{a_j}$ for all $j \in [s]$
- The corresponding key-images be defined as

$$I_j = g_t^{r_j} h^{a_j}$$

- We wish to design a ZKPoK of the form

$$\mathrm{PoK}\{(i_1, \ldots, i_s, \mathbf{r}, \mathbf{a}) \mid C_{i_j} = g^{r_j} h^{a_j} \wedge I_j = g_t^{r_j} h^{a_j} \ \forall j \in [s]\}$$

where $\mathbf{r} = (r_1, \ldots, r_s)$ and $\mathbf{a} = (a_1, \ldots, a_s)$

In the order of appearance on the blockchain

Motivating Revelio+

- Let the set of unspent outputs at block height t be

$$\{C_1, \underbrace{\boxed{C_2}}_{i_1}, \underbrace{\boxed{C_3}}_{i_2}, C_4, \dots, \underbrace{\boxed{C_{n-2}}}_{i_{s-1}}, C_{n-1}, \underbrace{\boxed{C_n}}_{i_s}\}$$

- Let the exchange-owned outputs be $(C_{i_1}, \dots, C_{i_s})$ such that $C_{i_j} = g^{r_j} h^{a_j}$ for all $j \in [s]$
- The corresponding key-images be defined as

$$I_j = g_t^{r_j} h^{a_j}$$

- We wish to design a ZKPoK of the form

$$\text{PoK}\{(i_1, \dots, i_s, \mathbf{r}, \mathbf{a}) \mid C_{i_j} = g^{r_j} h^{a_j} \wedge I_j = g_t^{r_j} h^{a_j} \ \forall j \in [s]\}$$

where $\mathbf{r} = (r_1, \dots, r_s)$ and $\mathbf{a} = (a_1, \dots, a_s)$
- We need a protocol which aggregates proofs of discrete-log based statements!

In the order of appearance on the blockchain

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$
- Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$ as $A = g^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$
- Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$ as $A = g^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$
- Proving $\mathbf{a} \circ \mathbf{b} = \mathbf{0}^n$, $\mathbf{a} - \mathbf{b} = \mathbf{1}^n$, $\langle \mathbf{a}, \mathbf{2}^n \rangle = a$ would suffice

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$
- Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$ as $A = g^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$
- Proving $\mathbf{a} \circ \mathbf{b} = \mathbf{0}^n$, $\mathbf{a} - \mathbf{b} = \mathbf{1}^n$, $\langle \mathbf{a}, \mathbf{2}^n \rangle = a$ would suffice
- To do so, form an inner-product relation given $y, z \xleftarrow{\$} \mathbb{Z}_q$

$$\langle \mathbf{a}, \mathbf{2}^n \rangle = a \ \wedge \ \langle \mathbf{a}, \mathbf{b} \circ \mathbf{y}^n \rangle = 0 \ \wedge \ \langle \mathbf{a} - \mathbf{b} - \mathbf{1}^n, \mathbf{y}^n \rangle = 0$$

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\text{PoK}\{(\mathbf{a}, \mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a}, \mathbf{b} \rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$
- Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$ as $A = g^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$
- Proving $\mathbf{a} \circ \mathbf{b} = \mathbf{0}^n$, $\mathbf{a} - \mathbf{b} = \mathbf{1}^n$, $\langle \mathbf{a}, \mathbf{2}^n \rangle = a$ would suffice
- To do so, form an inner-product relation given $y, z \xleftarrow{\$} \mathbb{Z}_q$

$$\langle \mathbf{a}, \mathbf{2}^n \rangle = a \;\wedge\; \langle \mathbf{a}, \mathbf{b} \circ \mathbf{y}^n \rangle = 0 \;\wedge\; \langle \mathbf{a} - \mathbf{b} - \mathbf{1}^n, \mathbf{y}^n \rangle = 0$$

- Proof size is $\mathcal{O}(\log(n))$

Bulletproofs (BP)

- A ZKPoK to prove that a committed value lies in a given interval
- BP uses improved inner-product argument, statement of which is

$$\mathrm{PoK}\{(\mathbf{a},\mathbf{b}) \mid A = \mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}} \wedge c = \langle \mathbf{a},\mathbf{b}\rangle\}$$

- Encodes an integer $a \in [0, 2^n)$ in binary form as $\mathbf{a}$, set $\mathbf{b} = \mathbf{a} - \mathbf{1}^n$
- Pedersen vector commitment to $\mathbf{a}, \mathbf{b}$ as $A = g^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$
- Proving $\mathbf{a} \circ \mathbf{b} = \mathbf{0}^n$, $\mathbf{a} - \mathbf{b} = \mathbf{1}^n$, $\langle \mathbf{a}, \mathbf{2}^n\rangle = a$ would suffice
- To do so, form an inner-product relation given $y, z \xleftarrow{\$} \mathbb{Z}_q$

$$\langle \mathbf{a}, \mathbf{2}^n\rangle = a \ \wedge \ \langle \mathbf{a}, \mathbf{b} \circ \mathbf{y}^n\rangle = 0 \ \wedge \ \langle \mathbf{a} - \mathbf{b} - \mathbf{1}^n, \mathbf{y}^n\rangle = 0$$

- Proof size is $\mathcal{O}(\log(n))$
- Extractability of bulletproofs depends on the assumption that discrete-log relation between $(g, \mathbf{g}, \mathbf{h})$ is unknown to the prover

Towards Revelio+

- Consider the problem of proving knowledge of a *single* private key (say, for instance, $R_i = g^{x_i}$) from a given set of public keys

$$\mathbf{R} = (R_1, R_2, \ldots, R_i, \ldots, R_n)$$

- Let $\mathbf{e}_i$ be unit vector with 1 in ith position, we can now write

$$1 = g^{-x_i}\mathbf{R}^{\mathbf{e}_i} = (g, \mathbf{R})^{(-x_i, \mathbf{e}_i)}$$

- With secrets $\mathbf{a} = (-x_i, \mathbf{e}_i)$, $\mathbf{b} = (x_i^{-1}, \mathbf{e}_i - \mathbf{1}^n)$, base vectors $\mathbf{g} = (g, \mathbf{R})$, $\mathbf{h} \xleftarrow{\$} \mathbb{G}^{n+1}$, we form the Pedersen vector commitment

$$P = (g')^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$$

- Forming an appropriate inner-product relation between $\mathbf{a}, \mathbf{b}$, we can design a Bulletproofs-like protocol!

Towards Revelio+

- Consider the problem of proving knowledge of a *single* private key (say, for instance, $R_i = g^{x_i}$) from a given set of public keys

$$\mathbf{R} = (R_1, R_2, \ldots, R_i, \ldots, R_n)$$

- Let $\mathbf{e}_i$ be unit vector with 1 in ith position, we can now write

$$1 = g^{-x_i}\mathbf{R}^{\mathbf{e}_i} = (g, \mathbf{R})^{(-x_i, \mathbf{e}_i)}$$

- With secrets $\mathbf{a} = (-x_i, \mathbf{e}_i)$, $\mathbf{b} = (x_i^{-1}, \mathbf{e}_i - \mathbf{1}^n)$, base vectors $\mathbf{g} = (g, \mathbf{R})$, $\mathbf{h} \xleftarrow{\$} \mathbb{G}^{n+1}$, we form the Pedersen vector commitment

$$P = (g')^{\alpha}\mathbf{g}^{\mathbf{a}}\mathbf{h}^{\mathbf{b}}$$

- ~~Forming an appropriate inner-product relation between **a**, **b**, we can design a Bulletproofs-like protocol!~~

Towards Revelio+

- Consider the problem of proving knowledge of a *single* private key (say, for instance, $R_i = g^{x_i}$) from a given set of public keys

$$\mathbf{R} = (R_1, R_2, \ldots, R_i, \ldots, R_n)$$

- Let $\mathbf{e}_i$ be unit vector with 1 in ith position, we can now write

$$1 = g^{-x_i} \mathbf{R}^{\mathbf{e}_i} = (g, \mathbf{R})^{(-x_i, \mathbf{e}_i)}$$

- With secrets $\mathbf{a} = (-x_i, \mathbf{e}_i)$, $\mathbf{b} = (x_i^{-1}, \mathbf{e}_i - \mathbf{1}^n)$, base vectors $\mathbf{g} = (g, \mathbf{R})$, $\mathbf{h} \xleftarrow{\$} \mathbb{G}^{n+1}$, we form the Pedersen vector commitment

$$P = g^{\alpha} \mathbf{g}^{\mathbf{a}} \mathbf{h}^{\mathbf{b}}$$

- ~~Forming an appropriate inner-product relation between **a**, **b**, we can design a Bulletproofs-like protocol!~~
- Extractability of Bulletproofs!

Towards Revelio+

- Lai *et al* fixed this by defining a new generator, $w \xleftarrow{\$} \mathbb{Z}_q, \mathbf{p} \xleftarrow{\$} \mathbb{G}^{n+1}$

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

- Further, $\mathbf{g}_w^{\mathbf{a}} = \mathbf{g}_{w'}^{\mathbf{a}}$ for any $w' \in \mathbb{Z}_q$
- Run BP-like protocol twice with bases $(g', \mathbf{g}_w, \mathbf{h})$ and $(g', \mathbf{g}_{w'}, \mathbf{h})$
- Extract secret vector $(\alpha, \mathbf{a}, \mathbf{b})$ if both runs were successful!
- Using $w = 0$ and $w' \xleftarrow{\$} \mathbb{Z}_q$, just one run suffices

Revelio+

$$\mathbf{C} \;=\; \{C_1 \quad \boxed{C_2} \quad \boxed{C_3} \quad C_4 \quad C_5 \quad \ldots \quad C_{n-2} \quad C_{n-1} \quad \boxed{C_n}\}$$

Revelio+

$$
\begin{array}{rcccccccccccc}
\mathbf{C} & = & \{C_1 & \boxed{C_2} & \boxed{C_3} & C_4 & C_5 & \ldots & C_{n-2} & C_{n-1} & \boxed{C_n}\} \\
\mathbf{e}_{i_1} & = & (\; . & 1 & . & . & . & \ldots & . & . & . \;) \\
\mathbf{e}_{i_2} & = & (\; . & . & 1 & . & . & \ldots & . & . & . \;) \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
\mathbf{e}_{i_s} & = & (\; . & . & . & . & . & \ldots & . & . & 1 \;)
\end{array}
$$

Revelio+

$$
\begin{array}{rcllllllllll}
\mathbf{C} & = & \{C_1 & \boxed{C_2} & \boxed{C_3} & C_4 & C_5 & \ldots & C_{n-2} & C_{n-1} & \boxed{C_n} & \} \\
\mathbf{e}_{i_1} & = & (\ . & 1 & . & . & . & \ldots & . & . & . &) \\
\mathbf{e}_{i_2} & = & (\ . & . & 1 & . & . & \ldots & . & . & . &) \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\
\mathbf{e}_{i_s} & = & (\ . & . & . & . & . & \ldots & . & . & 1 &)
\end{array}
$$

- We are to design a ZKPoK Π_+ for the statement

$$\text{PoK}\left\{(\mathbf{a}, \mathbf{r}, \mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_s}) \mid \mathbf{C}^{\mathbf{e}_{i_j}} = g^{r_j} h^{a_j} \ \wedge \ I_j = g_t^{r_j} h^{a_j} \ \forall j \in [s]\right\}$$

Revelio+

$$
\begin{array}{lcllllllllll}
\mathbf{C} & = & \{C_1 & \boxed{C_2} & \boxed{C_3} & C_4 & C_5 & \dots & C_{n-2} & C_{n-1} & \boxed{C_n} & \} \\
\mathbf{e}_{i_1} & = & (\ . & 1 & . & . & . & \dots & . & . & . &) \\
\mathbf{e}_{i_2} & = & (\ . & . & 1 & . & . & \dots & . & . & . &) \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\
\mathbf{e}_{i_s} & = & (\ . & . & . & . & . & \dots & . & . & 1 &)
\end{array}
$$

- We are to design a ZKPoK Π_+ for the statement

$$\mathrm{PoK}\left\{(\mathbf{a}, \mathbf{r}, \mathbf{e}_{i_1}, \dots, \mathbf{e}_{i_s}) \mid \mathbf{C}^{\mathbf{e}_{i_j}} = g^{r_j} h^{a_j} \ \wedge \ I_j = g_t^{r_j} h^{a_j} \ \forall j \in [s]\right\}$$

- For proving knowledge of *only* $\mathbf{e}_{i_j}$ for some $j \in [s]$,

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

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

$$\boxed{\therefore \ \mathbf{g}'_w = ((g \| g_t \| h \| \mathbf{C} \| I_j^u)^w \circ \mathbf{p}) \text{ and } \mathbf{a}' := (\xi_j \| \xi'_j \| \eta_j \| \hat{\mathbf{e}}_j \| 1)}$$

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_+)$

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_+)$

✓ Exchange cannot inflate reserves

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_+)$

✓ Exchange cannot inflate reserves

✓ Multiple exchanges cannot collude by sharing UTXOs

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_{+})$

✓ Exchange cannot inflate reserves

✓ Multiple exchanges cannot collude by sharing UTXOs

✓ Output-privacy of an exchange is conserved

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\mathrm{Rev}+} = (t, \mathbf{I}, \Pi_+)$

- ✓ Exchange cannot inflate reserves
- ✓ Multiple exchanges cannot collude by sharing UTXOs
- ✓ Output-privacy of an exchange is conserved
- ✓ Π_+ has perfect completeness and computational soundness

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_{+})$

✓ Exchange cannot inflate reserves

✓ Multiple exchanges cannot collude by sharing UTXOs

✓ Output-privacy of an exchange is conserved

✓ Π_{+} has perfect completeness and computational soundness

✓ Different exchanges can give proofs at same blockchain state

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_+)$

- ✓ Exchange cannot inflate reserves
- ✓ Multiple exchanges cannot collude by sharing UTXOs
- ✓ Output-privacy of an exchange is conserved
- ✓ Π_+ has perfect completeness and computational soundness
- ✓ Different exchanges can give proofs at same blockchain state
- ✓ Proof sizes are logarithmic in UTXO set size

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\mathrm{Rev}+} = (t, \mathbf{I}, \Pi_+)$

✓ Exchange cannot inflate reserves

✓ Multiple exchanges cannot collude by sharing UTXOs

✓ Output-privacy of an exchange is conserved

✓ Π_+ has perfect completeness and computational soundness

✓ Different exchanges can give proofs at same blockchain state

✓ Proof sizes are logarithmic in UTXO set size

✗ Revealing of own set size s

Revelio+ - The Good and The Bad

Revelio+ proof of reserves is given as $\Pi_{\text{Rev+}} = (t, \mathbf{I}, \Pi_+)$

- ✓ Exchange cannot inflate reserves
- ✓ Multiple exchanges cannot collude by sharing UTXOs
- ✓ Output-privacy of an exchange is conserved
- ✓ Π_+ has perfect completeness and computational soundness
- ✓ Different exchanges can give proofs at same blockchain state
- ✓ Proof sizes are logarithmic in UTXO set size
- ✗ Revealing of own set size s
- ✗ Proof generation and verification times

Performance

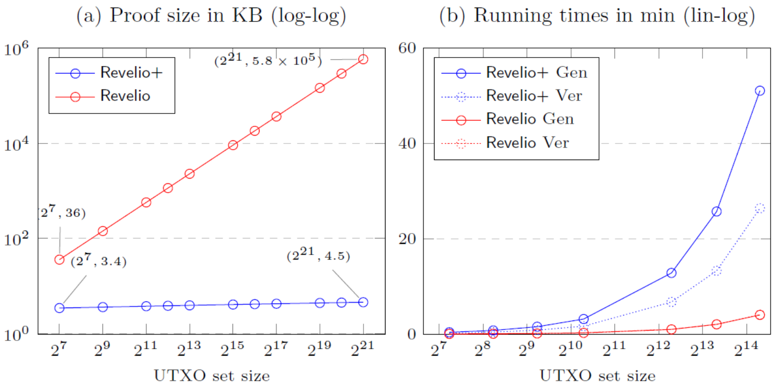


Figure: Performance comparison between Revelio and Revelio+ for $s = 50$

Performance

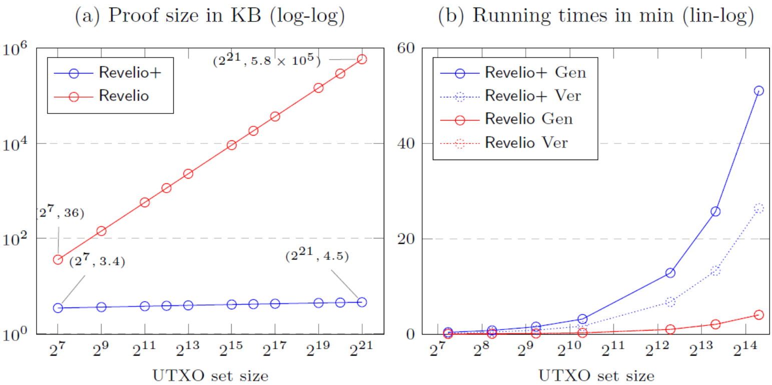


Figure: Performance comparison between Revelio and Revelio+ for $s = 50$

Performance

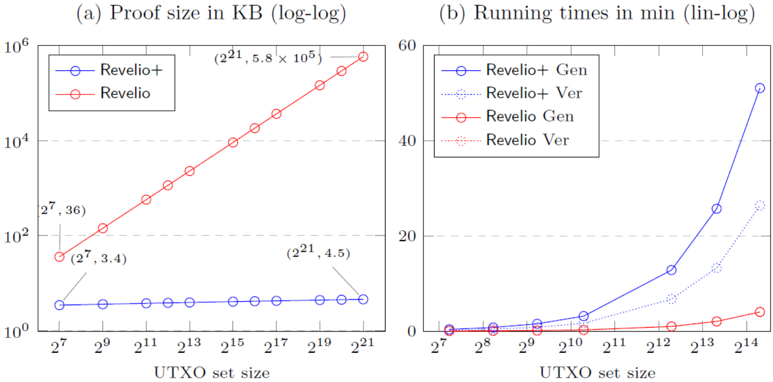


Figure: Performance comparison between Revelio and Revelio+ for $s = 50$

Performance - Generation Times

- Revelio+ protocol generation and verification times are linear in sn

		Revelio+	Revelio
n	s	$\mathcal{O}(sn)$	$\mathcal{O}(n)$
10^5	50	4.1	0.4
	10^3	82	0.4
	5×10^3	410	0.4
	10^4	810	0.4
10^6	50	41	4
	10^3	820	4
	10^4	8200	4
	10^5	16000	4

Table: Comparison of generation times[2] (in hrs)

[2]Run on a single core of a Intel i7-7700 3.6GHz CPU

Performance - Generation Times

- Revelio+ protocol generation and verification times are linear in sn

		Revelio+	Revelio
n	s	$\mathcal{O}(sn)$	$\mathcal{O}(n)$
10^5	50	4.1	0.4
	10^3	82	0.4
	5×10^3	410	0.4
	10^4	810	0.4
10^6	50	41	4
	10^3	820	4
	10^4	8200	4
	10^5	16000	4

Table: Comparison of generation times[2] (in hrs)

- Need specialized hardware! (multicore CPUs/FPGAs/ASICs?)

[2] Run on a single core of a Intel i7-7700 3.6GHz CPU