A ZK-SNARK based Proof of Assets Protocol for
Bitcoin Exchanges

Swaroopa Reddy B
Department of Electrical Engineering, Indian Institute of Technology Hyderabad, Hyderabad, India
ee17resch11004@iith.ac.in

Abstract—This paper proposes a protocol for Proof of Assets of a bitcoin exchange using the Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) without revealing either the bitcoin addresses of the exchange or balances associated with those addresses. The proof of assets is a mechanism to prove the total value of bitcoins the exchange has the authority to spend using its private keys. We construct a privacy-preserving ZK-SNARK proof system to prove the knowledge of the private keys corresponding to the bitcoin assets of an exchange. The ZK-SNARK toolchain helps to convert an NP-Statement for proving the knowledge of the private keys (known to the exchange) into a circuit satisfiability problem. In this protocol, the exchange creates a Pedersen commitment to the value of bitcoins associated with each address without revealing the balance. The simulation results show that the proof generation time, size, and verification time are efficient in practice.

Index Terms—Bitcoin Exchange, Zero-Knowledge Proofs, ZK-SNARK, Proof of Assets, Rank1 Constraint System, Quadratic Arithmetic Programs, Pedersen Commitment.

I. INTRODUCTION

Blockchain technology gained popularity due to its immutability, trustlessness, and decentralized architecture. Every public blockchain network is associated with a corresponding virtual currency named cryptocurrency (or shortly crypto). Satoshi Nakamoto introduced the first crypto called bitcoin with the deployment of the bitcoin blockchain [1] in 2009. The blockchain networks issue cryptocurrency through a mechanism known as the mining process, e.g., the Proof-of-Work [1] mechanism in bitcoin. The field of cryptocurrencies is ever-expanding, and as of today, there are more than 4000 cryptocurrencies in existence. Bitcoin has achieved one trillion-dollar market capitalization [2] as there is huge demand from institutional and retail investors. Other popular cryptocurrencies are Ethereum [3], Zerocash [4], etc.

In a blockchain network, every owner of the crypto holds a private key to spend the crypto through a chain of digital signatures [1]. If private keys are stolen or misplaced or the device where the private key stored crashes, the owner loses crypto ownership. So, the users prefer to keep their crypto holdings with exchanges like coinbase [5], binance [6], etc. The crypto exchanges facilitate crypto trading for fiat currencies or other cryptocurrencies and gain profits through commissions/brokerage charges, listing charges, etc. The exchanges act as an intermediary between buyer and seller by using the mechanism of order-book, similar to the traditional stock exchanges.

The crypto exchanges accept deposits from users through bank transfers or other standard means of deposit. The exchanges hold the private keys on behalf of the users and provides authentication facility through username and password to authenticate the customer’s identity and also provide the recovery facility in case of customer forgets or lost authentication details. So, the customers are free from storing private keys for their cryptocurrencies. But there is a risk of missing customer assets maintained by exchanges as in the case of Mt.Gox [7] (February 2014) and FTX (November 2022) [8] due to internal or external frauds. In the traditional banking system, the central bank imposes restrictions on the commercial banks to maintain a fraction of their total liabilities called fractional reserve ratio [9] as reserves, expecting that only a fraction of depositors seek to withdraw funds at the same time. But, the crypto community is expecting a fully solvent exchange instead of proving a fractional solvency of exchange’s reserves.

In this paper, we propose a proof of assets protocol for a bitcoin exchange based on the ZK-SNARK proof system [10], [11]. ZK-SNARK is an advancement in the zero-knowledge proofs. Zcash protocol proposed in [12] uses ZK-SNARK for constructing the decentralized anonymous payments. ZK-SNARK is a succinct, non-interactive zero-knowledge proof which facilitates the public verifiability of the proof of a witness. ZK-SNARKs enable the prover to convince the verifier on any non-deterministic decision circuits (NP-statements) with auxiliary information (witness) and public inputs without revealing the witness. A trusted third party takes the circuit as input and generates a common reference string (CRS) consisting of proving and verification keys needed to prove and verify the statement of the ZK-SNARK scheme.

In this framework, we define the non-deterministic circuit as a statement for verifying the knowledge of all the private keys owned by the exchange to prove the value of bitcoin assets held by the exchange. The exchange acts as a prover of its total assets and the customers of the exchange play the role of a verifier in the ZK-SNARK proof system. The proof of exchange assets is equivalent to proving ownership of the private keys associated with the bitcoin addresses (In this work P2PK (Pay to Public Key) addresses [13]) owned by an exchange to match the liabilities of the exchange to the customers.

The exchange as a prover takes the private keys as an
auxiliary input, public keys and the corresponding balances as public inputs. The exchange take the proving key and the inputs (auxiliary and public) as input parameters and constructs the proof for the witness. The exchange outputs a Pedersen commitment [14] to the balance associated with the key pair (part of the auxiliary input). The customers are the public verifiers for verifying the knowledge of the private key to acknowledge the reserves/assets of the exchange from the proof generated by exchanges. The proof size is succinct and it does not leaks the private keys (witness) used in the proof construction. The proposed protocol also preserves the privacy of the exchange as the proof system neither reveals the bitcoin addresses information nor the value of the bitcoins held by the exchange.

The results demonstrate that the construction of the proof requires a few hours on an ordinary computer which could be reduced further on a server with high-end processors, which allows the prover to generate proof of assets very frequently. The ZK-SNARK system generates the proof of size, approximately 128 bytes per private key, and the customer can verify the proof of exchange assets in the order of minutes.

The rest of the paper is organized as follows - In section II, we discuss the related work. Section III describes the preliminaries. In section IV, we discuss the proposed Proof of Assets protocol. In section V, we present the results and discussion. Section VI concludes the paper and gives future directions of the research.

II. RELATED WORK

In [15], the authors discuss the proof of reserves for bitcoin exchange. The maxwell’s proof of reserves discloses the number of bitcoins an exchange holds and the bitcoin addresses for which it knows the private keys. This framework uses the Merkle tree approach to prove the exchange’s liabilities by including each customer’s funds as a leaf of the Merkle tree. The proof of assets is a straightforward approach of providing signatures with all private keys owned by the exchange.

In Provisions [16], the authors propose a privacy-preserving proof of solvency for bitcoin exchanges using crypto primitives like zero-knowledge proofs [17], and Pedersen commitments [14]. Provisions discusses three protocols – Proof of assets, proof of liabilities, and proof of solvency. It also discusses the proof of non-collusion between exchanges. The proof of assets Σ protocol proves the knowledge of exchange’s assets by providing Pedersen commitments to the amounts of bitcoins the exchange holds for a set of known public keys. It also proves the knowledge of a binary value using Pedersen commitments if it knows the private keys corresponding to the known public keys.

In [18], an exchange constructs a transaction as proof of reserves with all the bitcoin UTXOs spendable by the exchange by adding an extra invalid input such that the exchange is unable to spend its own UTXOs. So, this approach discloses all the UTXOs of the exchange along with the public keys owned by an exchange.

III. PRELIMINARIES

In this section, we describe the background on cryptographic primitives used in the protocol - Elliptic curves, Pedersen commitments and ZK-SNARK. In this work, we stick to the Elliptic curve cryptography [19] used in bitcoin to prove the ownership of private keys corresponding to the bitcoin addresses.

A. Point Addition and Point Double

Let E (including a point at infinity O) be a group of order q corresponding to points on the elliptic curve $y^2 = x^3 + ax + b$ over a finite field $\mathbb{F}_p$. The addition operation + on E [19] is defined as follows -

If $P(x_1, y_1)$ and $Q(x_2, y_2)$, then $P + Q = (x_3, y_3)$, where $x_3 = m^2 - (x_1 + x_2)$, $y_3 = m(x_1 - x_3) - y_1$ \hspace{1cm} (1)

and,

$$m = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1}, & \text{if } P \neq Q \\ \frac{3x_1^2 + a}{2y_1}, & \text{otherwise} \end{cases}$$

B. Bitcoin public and private keys

Let $G \in E$ be a generator or base point. The public key $K$ (bitcoin address) corresponding to the private key $k \in \mathbb{Z}_q = \{1, 2, \ldots, q - 1\}$ is calculated as a scalar multiplication [19] of $k$ with $G$.

$$K = kG = G + G + \cdots + G \ (k \ \text{times}) \hspace{1cm} (2)$$

Calculating $k$ from $K$ is called the discrete logarithm problem [20] which is assumed to be hard.

C. Pedersen Commitments

Pedersen commitment [14] $c \in E$ is used to perfectly hide a message $m \in \mathbb{Z}_q$. Let $H \in E$ be another generator independent of $G$ and ensure that the discrete logarithm of $H$ for $G$ is unknown. The Pedersen commitment $c$ to a message $m$ is

$$c = mG + bH \hspace{1cm} (3)$$

Where $b \in \mathbb{Z}_q$ is a randomly chosen blinding factor. The Pedersen commitments are additively homomorphic.

D. ZK-SNARK

ZK-SNARK [10], [11] is a variant of Zero-knowledge proof of knowledge [17] with succinct proof. ZK-SNARK is used to prove and verify any instance $l$ belongs to an NP language $L$ such that the prover convinces verifier that it has knowledge of auxiliary input $w$ and public input $z$ for $l \in L$ without revealing $w$ to the verifier. This can be further formalized as a relation

$$\mathcal{R} := \{(z, w) \in \{0, 1\}^* \times \{0, 1\}^* \ s.t \ C(z, w) = 0\} \hspace{1cm} (4)$$

ZK-SNARK uses a trick to reduce any NP-statement $L$ to circuit satisfiability problem (NP complete problem). The NP-statement is converted to non-deterministic decision circuit $C$ such that the input to the statement is transformed as the input
to the circuit $C$. Let $L$ be an NP statement and $C$ represents the non deterministic decision circuit. The ZK-SNARK toolchain takes the circuit $C$ of the instance $l$ as input and generates ZK-SNARK proof system as described in Fig. 1.

In this paper, we stick to the ZK-SNARK framework proposed by Groth in [11] as this protocol consists of a shorter proof with 3 group elements (2 elements from group $\mathbb{G}_1$ and 1 element from group $\mathbb{G}_2$) compared to 9 group elements in Pinocchio protocol [10]. Also the proof construction and verification times are less in Groth’s protocol compared to Pinocchio’s protocol. ZK-SNARK consists of three algorithms defined as follows -

- **Setup**($1^\lambda, R$) $\leftarrow \sigma$: On input security parameter $\lambda$ and relation $R$, outputs a common reference string $\sigma$.
- **Prove**($\sigma, x, w$) $\leftarrow \pi$: On input $\sigma$ and a statement-witness pair $(x, w) \in R$, outputs a proof $\pi$.
- **Verify**($\sigma, x, \pi$) $\leftarrow 0, 1$: On input $\sigma$, statement $x$ and proof $\pi$, outputs a bit to indicate if the proof is valid.

IV. THE PROPOSED PROOF OF ASSETS PROTOCOL

The protocol consists of three major entities - Trusted third party, Prover - crypto exchange $E$ and, Verifier - customer $C$ of the exchange $E$, who holds crypto assets with $E$. In Proof of Assets protocol, the exchange proves the total bitcoins over which it has the ownership authority to spend. In the proposed protocol, the exchange $E$ proves its total assets in zero-knowledge and also it preserves the privacy without revealing its public key addresses and associated balances. The exchange $E$ generates a ZK-SNARK proof for an NP-statement which says $E$ knows the private keys for a subset of bitcoin addresses (public keys) and also computes a Pedersen commitment to its bitcoin assets.

Let $\textbf{PK}$ be the set of total bitcoin public keys on the blockchain.

$$\textbf{PK} = \{y_1, y_2, \ldots, y_k\} \subseteq \mathcal{E}$$

Let $x_1, x_2, \ldots, x_k \in \mathbb{Z}_p$ are the set of private keys corresponding to the public keys from the set $\textbf{PK}$, such that $y_i = x_iG$ for $i = 1, 2, \ldots, k$.

Let $\textbf{S}_{\text{own}}$ be the subset of the public keys for which the exchange knows the private keys and $\textbf{S}_{\text{own}} \subseteq \textbf{PK}$. If $E$ provides proofs only for the private keys associated with addresses in $\textbf{S}_{\text{own}}$, it reveals the bitcoin addresses and total assets owned by $E$. So, $E$ takes an anonymity set $\textbf{S}_{\text{anon}} = \{y_1, y_2, \ldots, y_n\}$ ($n < k$) such that $\textbf{S}_{\text{anon}} \in \textbf{PK}$ and $\textbf{S}_{\text{anon}} \supset \textbf{S}_{\text{own}}$ to prove the assets owned by $E$.

Let $s_i \in \{0, 1\}$ denotes which public keys the exchange knows the private key. If $s_i = 1$, then the exchange knows the private key $x_i$ corresponding to the bitcoin address $y_i \in \textbf{S}_{\text{own}}$. Let $v_i$ denotes the amount of bitcoins associated with address $y_i \in \textbf{S}_{\text{anon}}$ and $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$, the total assets of the exchange called $\mathcal{E}_{\text{Assets}}$ is defined as

$$\mathcal{E}_{\text{Assets}} = \sum_{i=1}^{n} s_i v_i = \sum_{y_i \in \textbf{S}_{\text{own}}} v_i$$ (6)

A. The NP-statement PoA for Proof of Assets

We construct a Proof of Assets protocol for an exchange to prove the ownership of the bitcoins it hold. We use ZK-SNARK and Pedersen commitment to prove exchange’s assets in zero-knowledge. The ownership of the bitcoin is defined by $s_i$, which can be evaluated by checking the equation $y_i = x_iG$. To preserve the integrity and privacy of the exchange, $x_i$ and $s_i$ are used as a part of the witness in the construction of the ZK-SNARK proof system. The Pedersen commitment hides $v_i$ through the secret $r_i$.

Let $F = \text{PoA}(z_i, w_i)$ be the NP-statement or the non-deterministic decision function with the public inputs $z_i = \{y_i, v_i\}$ and the witness $w_i = \{x_i, s_i, r_i\}$, then the NP-statement $\text{PoA}$ for proving the exchange’s ownership on $x_i$ is defined as

“Either I know the private key $x_i$ corresponding to a public key $y_i$ in which case $s_i = 1$ and $c_i$ is the commitment to the value $v_i$ or I don’t know the private key $x_i$ corresponding to the public key $y_i$ in which case $s_i = 0$ and $c_i$ is the commitment to the value 0.”

The statement $\text{PoA}$ is captured by the relation $R_{\text{PoA}} = \{(z_i, w_i) \in \mathbb{F}_p^{l+1} \times \mathbb{F}_p^{m-1} \text{ s.t. } C_{\text{PoA}}(z_i, w_i) = c_i\}$. Where, $C_{\text{PoA}}$ is an arithmetic circuit representation of the statement $\text{PoA}$ and the tuple $(z_i, w_i)$.

B. Arithmetic Circuit $C_{\text{PoA}}$ for verifying NP-statement $\text{PoA}$

As shown in Fig. 1, ZK-SNARK is a proof system for arithmetic circuit satisfiability problem to prove the witness $w_i$ for an instance $z_i$. So, we express the checks in the NP-Statement $\text{PoA}$ to arithmetic circuit $C_{\text{PoA}}$ as depicted in Fig. 2. There are three subcircuits in $C_{\text{PoA}}$ - Scalar multiplication circuit $C_{\text{MUL}}$, Comparison circuit $C_{\text{CMP}}$, and Pedersen commitment circuit $C_{\text{PED}}$. Each individual circuit consists of arithmetic gates for ‘+, ‘∗’, ‘−’, or ‘/’.

1) Scalar Multiplication circuit verification:: The circuit $C_{\text{MUL}}$ is the scalar multiplication of $x_i$ with base point $G$. It is used to test whether the exchange $E$ knows the private key $x_i$ matches to the corresponding public key $y_i$. The scalar multiplication is defined as

$$y_i = x_iG = G + G + \cdots + G \ (x_i \text{ times})$$ (7)

Where, $x_i$ and $x, y$-coordinates of the $G$ are 256-bit numbers over the field $\mathbb{F}_p$ used for bitcoin addresses. Instead of constructing $x_i$ number of point addition circuits of $G$, we construct $C_{\text{MUL}}$ from 256 point addition circuits. Initially, scalar $x_i$ is unpacked into a vector $xvec_i$ of 256-bits with...
cuit evaluates the Pedersen commitment of $C$.

Fig. 2. Circuit $C_{PoA}$ for the NP-statement $PoA$.

257 constraints (Each constraint represents a single arithmetic gate).

Let $P_{add}$ be a point addition circuit. We construct $P_{add}$ gadget with 3 arithmetic gates as per point addition for elliptic curve points defined in (1). We process 2-bits (out of 256 bits) of $x_i$ at a time to reduce the number of $P_{add}$ gadgets to half of the total number of bits (i.e to 128) by adding an extra constraint (total of 4 constraints per $P_{add}$) for the product of the two bits of $x_i$.

The summary of the total number of constraints (or arithmetic gates) required to construct the proof for $C_{MUL}$ circuit is listed in Table I.

2) Comparison circuit verification:: The comparison function $CMP$ compares the output of the scalar multiplication $y_i$ with the input $y_i$. The circuit ensures the comparison of $x$, $y$ coordinates of $y'_i$ and $y_i$ to yield a binary output $s_i$. This is achieved using a single constraint to obtain the product of $sx$ and $sy$ as $s$, where $sx$ and $sy$ are the outputs for comparing $x$ and $y$ coordinates of $y'_i$, $y_i$ respectively. We also add a constraint to $C_{MUL}$ circuit to ensures that $s_i \in \{0,1\}$ using the operation $s_i \ast s_i = s_i$. So, we express the $C_{MUL}$ circuit using two constraints.

3) Pedersen Commitment circuit Verification:: $C_{PED}$ circuit evaluates the Pedersen commitment of $b_i = s_i \ast v_i$ with a randomly chosen blinding factor $r_i$ which yields an output $c_i = b_iG + r_iH$. This circuit requires two $C_{MUL}$ circuits, an arithmetic gate for computing $b_i$. Since the total bitcoin supply is limited to 21 million coins, the value of $v_i$ is bounded by $2^z$, where $z \in [0, 51]$. So, the total number of constraints for the first $C_{MUL}$ circuit, i.e., $b_iG$ depends on the maximum value of $z$. Since the blinding factor $r_i \in \mathbb{Z}_q$, the second multiplication is similar to the $C_{MUL}$ circuit discussed in Table I. The summary of the total number of constraints required to construct a proof for $C_{PED}$ circuit is listed in Table II.

Finally, by combining all the circuits, a total of 1702 constraints are required to prove circuit $C_{PoA}$ for NP-statement $PoA$. When $E$ wants to provide a proof $\pi_i$ for $y_i \in S_{anon}$ and $y_i \notin S_{own}$, it can directly compute the proof with only considering the Pedersen commitment circuit $C_{PED}$ by directly taking $s_i = 0$ to reduce the number of constraints to 928.

C. The Proof of Assets protocol

The proof of assets protocol shown in Table III for bitcoin exchanges is constructed based on the ZK-SNARK framework illustrated in Section III.

1) CRS Setup:: A trusted third party construct a ZK-SNARK’s common-reference string $\sigma$ for the NP-statement $PoA$ as per the ZK-SNARK toolchain described in Fig. 1. First, it construct a QAP $Q$ from the circuit $C_{PoA}$ for the statement $PoA$, then it generates CRS $\sigma$.

2) Generation of the proof:: The crypto exchange $E$ (Prover) constructs a proof $\pi_i$ to prove the knowledge of a private key $x_i$ corresponding to the public key $y_i$. Where, $z_i = (y_i, v_i)$ represents the public input from the blockchain and $w_i = \{x_i, s_i, b_i, r_i, t_i\}$ represents the $E$’s auxiliary input . Where, $t_i$ is the assignments for all the internal wires of the circuit $C_{PoA}$ excluding $s_i$ and $b_i$.

The prover also provide output $c_i$ to the verifier, which is a Pedersen commitment to $b_i = s_i \ast v_i$, for all $i = 1, \ldots, n$. If the exchange knows the private key $x_i$, then $s_i = 1$ and $c_i$ is the commitment to balance $v_i$, other wise it is a commitment to value zero.

3) Verification of the proof and total assets:: The customer $C$ (Verifier) takes the public inputs $z_i = (y_i, v_i)$ from blockchain and the proof $\pi_i$ from $E$ verifies the proof $\pi_i$, for $i = 1, \ldots, n$ along with the proof for $C_{Assets}$.

D. Security and Privacy analysis

In this section, we discuss the security properties of the proposed Proof of Assets protocol - completeness, soundness and statistical zero-knowledge. We also discuss the privacy of the exchange.
The experiment for a single instance that for all AddressPrivacy as it captures if $E$ follows:

The proof for the security properties of the Proof of Assets protocol is straightforward from the construction of Lemma IV.1.

The above construction of the proof of assets protocol satisfies the completeness, soundness and statistical zero-knowledge properties.

Proof. The proof for the security properties of the Proof of Assets protocol is straightforward from the construction of the ZK-SNARK proof system. We omit the proof due to the space constraints.

The privacy of $E$ depends on the value of $s_i \in \{0, 1\}$ as it captures if $E$ knows a private key or not. We define an experiment and we call $AddressPrivacy^D_{y_i, c_i}$. It denotes that for all $y_i \in S_{anon}$ a distinguisher $D$ tries to distinguish $y_i \in S_{own}$ or not, balance $v_i \in z_i$ and output commitment $c_i$. The experiment for a single instance $z_i \in L_{PoA}$ is defined as follows:

1) The public parameters are generated as $\mathbb{F}_p, q, G, H \leftarrow Gen(1^\lambda)$, where $\lambda$ is the security parameter.
2) $E$ picks $b_i \in \{0, 1\}$.
3) $E$ picks $r_i \in \mathbb{Z}_q$ and computes $c_i = b_i v_i G + r_i H$ and it determines the corresponding ZK-SNARK proof $\pi_i$ for

$\pi_i \leftarrow \text{SNARK.Prove}(\sigma, z_i, w_i)$ (8)

$\pi_i \leftarrow \text{SNARK.Verify}(\sigma, z_i, c_i, \pi_i)$ (9)

$\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ (10)

$C_{Assets} = \sum_{i=1}^{n} c_i$ (11)

$C_{Assets} = \sum_{i=1}^{n} c_i$ (12)

$C_{Assets} = \sum_{i=1}^{n} c_i$ (13)

TABLE III

PROOF OF ASSETS PROTOCOL

| $i$ | $\pi_i \leftarrow \text{SNARK.Prove}(\sigma, z_i, w_i)$ | $r_i \leftarrow \mathbb{Z}_q$ | $c_i = s_i v_i G + r_i H$ | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, z_i, c_i, \pi_i)$ | $C_{Assets} = \sum_{i=1}^{n} c_i$ |
|-----|--------------------------------------------------|-----------------------------|-----------------------------|--------------------------------------------------|-----------------------------|
| 1   | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ | $\pi_i \leftarrow \text{SNARK.Verify}(\sigma, \pi_i, C_{Assets})$ |

Consider a proof of assets protocol run between an exchange $E$ and a customer $C$. Let $\text{out}_{PoA} \in \{\text{Accept}, \text{Reject}\}$ be the output of $C$ based on the verification checks (a), (b) and (c) in verification phase of the protocol.

Lemma IV.1. The above construction of the proof of assets protocol satisfies the completeness, soundness and statistical zero-knowledge properties.

Proof. Suppose an adversarial exchange $A_E$ wants to solve a DL problem. $A_E$ picks $b_i \in \{0, 1\}$ and determine $c_i = b_i v_i G + r_i G$ (step 2 in $AddressPrivacy^D_{y_i, c_i}$). It gives $(z_i = (y_i, v_i), c_i, \pi_i)$ as input to $D$. Since $D$ is a PPT algorithm, $A_E$ also a PPT algorithm.

The generators $G$ and $H$ are chosen uniformly and independently from $\mathbb{F}_p$. We have $G = kH$, for some unknown $k$ ($k$ is not known to $A_E$).

$y_i = x_i G, c_i = (kv_i + r_i) H$, if $b_i = 1$ (15)

If $b'_i = D(z_i, c_i, \pi_i)$, then $A_E$ outputs $b'_i$. Suppose $D$’s success probability $Pr[b'_i = b_i] > \frac{1}{2} + \text{negl}(\lambda)$, then $A_E$’s success probability is also larger than $\frac{1}{2} + \text{negl}(\lambda)$. The PPT adversary cannot solve the DL problem, this is a contradiction. Thus, $D$ has an advantage less than $\frac{1}{2} + \text{negl}(\lambda)$ to reveal the privacy of the exchange.

V. IMPLEMENTATION AND PERFORMANCE EVALUATION

We have implemented the Proof of Assets protocol in C++ using the libsnark library [21] developed by scipr-lab.

We divide the implementation of PoA.gadget for PoA into three subcircuits or gadgets. The scalar multiplication gadget (scalar_mul) verifies the knowledge of $E$ on private key $x_i$. The comparison gadget (cmp_gadget) checks the equality of the computed public key $(q_i^p)$ from the given input address $(y_i)$. Finally, the Pedersen commitment gadget

$\sum_{i=1}^{n} c_i$
(Pedersen_gadget) verifies the commitment \( c_i \) to the balance \( v_i \). The PoA_gadget generates a total of 1702 constraints for each instance \((z_i, w_i) \in R_{PoA}\).

We performed tests on a personal computer with Intel(R) Core(TM) i9–9900K CPU @ 3.60GHz processor with 16GB RAM using a single core. The details of the proof construction time by an exchange \( \mathcal{E} \), proof verification time by the customer \( \mathcal{C} \), and the proof size are described in Table IV. The proof construction time depends on the number of the ZK-SNARK circuit’s constraints. The proof size is the combination of size of 3 group elements of each proof \( \pi_i \) and size of the Pedersen commitment to \( v_i \).

Table V illustrates the performance of the Proof of Assets protocol with the size of the set \( S_{\text{non}}(n) \). We test the protocol for \( n = 100, 1000 \) and, 10000. We choose \( |S_{\text{own}}| \) as 25%, 50% and 75% of \( n \). We assume \( \mathcal{E} \) provides proofs for \( n - |S_{\text{own}}| \) number of addresses by considering \( s_i = 0 \) for all \( y_i \notin S_{\text{own}} \) to reduce the proof construction time. The proof construction time includes the time required for the construction of the proofs for \( n \) number of \( C_{PoA} \) and the time required to generate the proof for the Pedersen commitment \( C_{Assets} \) (10). Similarly, for the proof verification and proof size.

The construction time, verification time, and proof size increases linearly with \( n \). The results show that the Proof of Assets protocol is efficient in practice as the regular PC constructs the proof in less than an hour and the proof size is less than \( \approx 15 \) MB for \( n = 10000 \) with short verification time. The performance of the protocol will be improved on servers with high-end processors.

### VI. Conclusions and Future Research

In this paper, we described the ZK-SNARK based proof of assets protocol for bitcoin exchanges by preserving the privacy of the exchanges without revealing the public keys or the balances associated with the public keys. This is achieved by proving the knowledge of the private keys associated with the public keys (Bitcoin P2PK addresses) using the ZK-SNARK mechanism with Pedersen commitment as the output of the circuit. We also analyse the security and privacy properties of the proposed protocol. Through the simulation results, we showed the efficiency of the protocol for proof construction, verification and proof size. In the future, we foresee the construction of the proof of assets protocol for bitcoin P2PKH (Pay to Public Key Hash) addresses by proving the knowledge of the hash preimage through the ZK-SNARK framework. We may also combine these proof of assets protocols with proof of liabilities by proving the membership of customer funds using the set-membership proofs and ZK-SNARK mechanism.

### REFERENCES

[1] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[2] CoinMarketCap. [Online]. Available: https://coinmarketcap.com

[3] Ethereum. [Online]. Available: https://ethereum.org/en/

[4] Zerocash. [Online]. Available: http://zerocash-project.org/

[5] Coinbase. [Online]. Available: https://www.coinbase.com/

[6] Binance. [Online]. Available: https://www.binance.com/en

[7] M. Gox. [Online]. Available: https://en.wikipedia.org/wiki/Mt_Gox

[8] FTX. [Online]. Available: https://en.wikipedia.org/wiki/FTX

[9] FRR. [Online]. Available: https://www.investopedia.com/terms/t/reserveratio.asp

[10] B. Parno, J. Howell, C. Gentry, and M. Raykova, “Pinocchio: Nearly practical verifiable computation,” in 2013 IEEE Symposium on Security and Privacy, 2013, pp. 238–252.

[11] J. Groth, “On the size of pairing-based non-interactive arguments,” in Advances in Cryptology – EUROCRYPT 2016, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326.

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

[13] bit2me, “P2pk.” [Online]. Available: https://academy.bit2me.com/en/what-is-a-p2pk/

[14] T. P. Pedersen, “Non-interactive and information-theoretic secure verifiable secret sharing,” in Proceedings of the 11th Annual International Conference on Advances in Cryptology, ser. CRYPTO ’91. Berlin, Heidelberg: Springer-Verlag, 1991, p. 129–140.

[15] “Proving-bitcoin-reserves.” [Online]. Available: https://web.archive.org/web/20170114112433/https://iwilcox.me.uk/2014/proving-bitcoin-reserves.

[16] 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, ser. CCS ’15. New York, NY, USA: Association for Computing Machinery, 2015, p. 720–731. [Online]. Available: https://doi.org/10.1145/2810103.2813674

[17] U. Feige and A. Shamir, “Witness indistinguishable and witness hiding protocols,” in Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing, ser. STOC ’90. New York, NY, USA: Association for Computing Machinery, 1990, p. 416–426. [Online]. Available: https://doi.org/10.1145/100216.100272

[18] Blocksream, “Standardizing bitcoin proof of reserves.” [Online]. Available: https://bitcoin.stackexchange.com/questions/56246/standardizing-bitcoin-proof-of-reserves.

[19] D. Harkerson, A. J. Menezes, and S. Vanstone, Guide to Elliptic Curve Cryptography. Berlin, Heidelberg: Springer-Verlag, 2003, ch. 3, pp. 96–106.

[20] S. D. Galbraith and P. Gaudry, “Recent progress on the elliptic curve discrete logarithm problem,” Cryptology ePrint Archive, Report 2015/1022, 2015. [Online]. Available: https://eprint.iacr.org/2015/1022

[21] scipp lab, “Libsnark.” [Online]. Available: https://github.com/scipp/libsnark

| \( n \) | \( |S_{\text{own}}| \) (% of \( n \)) | Construction time (seconds) | Verification time (seconds) | Proof size (MB) |
|---|---|---|---|---|
| 100 | 25 | 16.96 | 0.456 | 0.01914 |
| 100 | 50 | 19.55 | 0.455 | 0.01914 |
| 100 | 75 | 22.76 | 0.455 | 0.01914 |
| 1000 | 25 | 166.026 | 4.528 | 0.1914 |
| 1000 | 50 | 191.942 | 4.514 | 0.1914 |
| 1000 | 75 | 217.934 | 4.512 | 0.1914 |
| 10000 | 25 | 1657.54 | 45.36 | 1.914 |
| 10000 | 50 | 1913.79 | 45.07 | 1.914 |
| 10000 | 75 | 2196.55 | 45.09 | 1.914 |