BitcoinVM

Saravanan Vijayakumaran

August 8, 2022

Abstract

A BitcoinVM is a circuit which maintains the set of all Bitcoin UTXOs in a sparse Merkle tree (SMT). Each new Bitcoin block will be used to update the SMT using a validity proof. The SMT can be used to generate privacy-preserving proofs of ownership of a certain amount of bitcoins. Such proofs will enable privacy-preserving proofs of reserves of cryptocurrency exchanges.

1 Motivation

Cryptocurrency exchanges may engage in fractional-reserve operations, where they sell more bitcoins to their customers than they own. Provisions [1] was the first privacy-preserving proof of solvency protocol to address this issue.

A shortcoming of the Provisions protocol was that it could only include bitcoins stored in *Pay To Public Key (P2PK)* and *Pay To Public Key Hash (P2PKH)* addresses, with the additional constraint that the preimage of the P2PKH addresses must be known. Even before SegWit activation, Bitcoin supported two additional address types: *m-of-n Multi-signature (multisig)* and *Pay to Script Hash (P2SH)* adddresses [2, Section 5.5].

Multisig addresses offer better security and flexibility by requiring signatures from any m out of n public keys to spend a UTXO. Multisig addresses require all n keys to be explicitly specified in a UTXO, thereby increasing the transaction cost of sending bitcoins to a multisig address.[1] P2SH addresses solve this issue by requiring that only a 20 byte hash (SHA256 + RIPEMD160) of a multisig address be specified to receive funds into it.

To the best of our knowledge, there is no publicly verifiable privacy-preserving proof of reserves (PoR) protocol for Bitcoin that supports all the address types. Current PoR protocols used in practice involve a trusted party — an auditor or the exchange themselves [3]. For example, the Kraken exchange uses a third-party auditor to help run their PoR protocol [4]. In Feb 2020, Kraken CEO Jesse Powell mentioned lack of multisig address support in Provisions as one of the reasons to defer attempting its use [5].

Goal: Enable privacy-preserving proof of Bitcoin reserves that is publicly verifiable on-chain.

[1] Bitcoin transaction fees is proportional to the length of the transaction.

What about collusion? An exchange with insufficient Bitcoin reserves could generate a valid PoR by colluding with another party who has sufficient reserves. The colluding party would not need to transfer the funds to the exchange. They would only be required to share the output of the PoR protocol with the exchange.

One approach to prevent collusion is to have the PoR protocol output a nullifier for the UTXOs used by the prover. This would help detect collusion only if the party colluding with the exchange also generates their own PoR. The colluding party may itself be an exchange, who generates PoRs to assure customers. But if the colluding party does generate their own PoR, then the nullifier approach fails to detect collusion.

Even without collusion-resistance, it seems worthwhile to have a privacy-preserving Bitcoin PoR protocol that supports all UTXO types. Generating a proxy PoR for an insolvent exchange could be considered unethical, which would reduce the number of willing parties. For large reserve amounts, parties capable of generating the proxy PoR may not want to risk litigation or reputational damage.

2 BitcoinVM

To prove ownership of Bitcoin UTXOs in a privacy-preserving manner, they have to be stored in data structure that is amenable to zero-knowledge proofs. Our initial plan is as follows:

1. Store all the Bitcoin UTXOs upto a certain block height in a sparse Merkle tree (SMT). The key to the leaves could be a hash of the transaction ID (TxID) and output index that identifies a UTXO.
2. Each Bitcoin block has transactions that spend old UTXOs and create new ones. The SMT would need to be updated after each block. The public inputs to the SMT update circuit will be the following:
   - The previous SMT root
   - The new SMT root
   - The block hash of the new Bitcoin block
3. The SMT update circuit will not verify the correctness of the new Bitcoin block. We assume that the verifier will check that the block hash input corresponds to a valid Bitcoin block that has enough confirmations.
4. Given a valid SMT with UTXOs as its leaves, ownership of a certain amount of coins can be proven using privacy-preserving Merkle proofs
5. The circuit that proves ownership of UTXOs needs to be able to verify scripts written in Bitcoin Script. These scripts have a sequence of 1-byte opcodes and are not restricted to a few templates (see examples on the Miniscript page [6]). To support different kinds of UTXOs, the circuit needs to interpret the UTXO's scriptPubkey one byte at a time.

6. Some opcodes in Bitcoin Script like OP_CHECKSIG require a large number of rows (approximately 140,000) in halo2, due to the ECDSA signature verification. Some other opcodes like OP_1 require only one row. Implementing constraints of all opcodes at each byte of the scriptPubkey would be inefficient.

7. We propose to parse a UTXO's scriptPubkey and **accumulate the public inputs for all cryptographic opcodes** that need valid witnesses [7]. Examples of such inputs are the following:

   - Public keys for the OP_CHECKSIG opcode
   - Public keys for the OP_CHECKMULTISIG opcode
   - Hash outputs corresponding to the OP_HASH160 opcode

   To illustrate the accumulation, suppose a UTXO has the following scriptPubkey:

   ```
   <key1> OP_CHECKSIG OP_SWAP <key2> OP_CHECKSIG OP_BOOLOR
   ```

   A valid witness for this UTXO would be an ECDSA signature corresponding to one of the two keys. Only one of the keys needs to be accumulated in the accumulator corresponding to the OP_CHECKSIG opcode.

8. The BitcoinVM circuit will verify that the prover can provide witnesses for all accumulated cryptographic opcodes **upto a maximum number**.

   - Suppose we fix the maximum number of OP_CHECKSIG opcodes to be 5. This would mean that we cannot use BitcoinVM to generate an ownership proof for UTXOs that require witnesses corresponding to 6 or more OP_CHECKSIG opcodes.

9. A PoR will require combining proofs of ownership of several UTXOs. This step might require proof aggregation and/or recursion.

3 Current Status

So far, we have implemented the following circuits. The code is available at `https://github.com/avras/bitcoinvm`.

- RIPEMD160 hash function
- Data push operators (`https://en.bitcoin.it/wiki/Script#Constants`)
- P2PK UTXO witness verification

A lot of work remains. Here is a partial list of circuits to be implemented.

- P2PKH UTXO witness verification

- P2SH UTXO witness verification
- Multsig UTXO witness verification
- HASH160 witness verification
- Sparse Merkle Tree
- SegWit UTXOs witness verification

References

[1] Gaby G. Dagher, Benedikt Bünz, Joseph Bonneau, Jeremy Clark, and Dan 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)*, pages 720–731, New York, NY, USA, 2015. https://crypto.stanford.edu/~dabo/pubs/abstracts/provisions.html.

[2] Saravanan Vijayakumaran. An Introduction to Bitcoin, 2017. https://www.ee.iitb.ac.in/~sarva/bitcoin/bitcoin-notes-v0.1.pdf.

[3] Nic Carter. Proof of Reserves, 2022. https://niccarter.info/proof-of-reserves/.

[4] Kraken. Proof of Reserves, 2022. https://www.kraken.com/proof-of-reserves.

[5] Jesse Powell. Tweet in response to being informed about Provisions, 2020. https://twitter.com/jespow/status/1229643409448554496.

[6] Pieter Wuille. Miniscript documentation. https://bitcoin.sipa.be/miniscript/.

[7] Bitcoin Cryptographic Opcodes. https://en.bitcoin.it/wiki/Script#Crypto.