Proposal: Enabling SNARKs for Bitcoin

Aggelos Kiayias[1,2] and Orfeas Stefanos Thyfronitis Litos[1]

[1] University of Edinburgh
[2] IOHK
akiayias@inf.ed.ac.uk, o.thyfronitis@ed.ac.uk

1 Motivation & Aims

Currently Bitcoin [1] transactions store permanently the pseudonymous addresses of transacting parties in the clear on the blockchain. As shown by [2], correlating this information with social network graphs to deanonymize parties is practical and relatively cheap. Avoiding the reuse of addresses does not protect from such attacks against privacy. Techniques proposed in the past such as CoinJoin [3] need active user coordination, are prone to DoS attacks and provide only heuristic privacy guarantees that can be violated by a determined adversary.

Existing work [4,5,6] shows that it is possible and practical to integrate zero-knowledge proof systems into blockchains. The aim of this proposal is to enable the use of zk-SNARKs in Bitcoin. Such an extension could bring a number of useful features to Bitcoin, which is currently the cryptocurrency with the highest market capitalization[3].

- stronger privacy guarantees,
- extension of the bitcoin scripting language in a sandboxed manner,
- a stronger layer-1 system for enabling arbitrary layer-2 protocols [7,8,9,10,11] for nearly instant and low-fee smart contract capabilities in a trustless manner.

The above can be achieved with minimal changes to Bitcoin: the new feature will be available through four new Bitcoin Script[4] opcodes that can be introduced through a soft fork [12]

2 Proposal

Existing zk-SNARK systems need a *structured reference string* (SRS) to generate and verify proofs. This string is public information, but its generation must be carried out by an honest party: if the Adversary is the one that generates the SRS, then it can use information from this generation procedure to later create valid proofs for false statements, completely subverting the zk-SNARK security.

One way to alleviate this problem is to make the SRS *updateable*, i.e. to allow the SRS to change throughout the lifetime of the zk-SNARK system. Each

[3] https://coinmarketcap.com
[4] https://en.bitcoin.it/wiki/Script

update would be based on the previous in a manner that ensures that even if a single updater has been honest, then no one can generate valid proofs to false statements. Such a capability is provided by SONIC [13], we therefore choose this as our zk-SNARK system.

We propose to add four new opcodes to the Bitcoin Script: `OP_SRS_CREATE`, `OP_SRS_UPDATE`, `OP_SNARK_CREATE` and `OP_SNARK_UPDATE`. To simplify the description and avoid a number of complications, we require that if one of these opcodes appears in a script, it has to be the only opcode in that script, otherwise the transaction is invalid. Further investigation is needed to determine whether it is possible to lift this limitation.

`OP_SRS_CREATE` generates a new SRS. It can be followed by either two or three data fields. The first data field contains the new SRS, the second the proof of its correctness and the optional third field contains (a reference to) the description of an NP relation R that all SNARKs under this SRS must observe. If present, the last field defines a state machine within which the SNARK system will operate.

`OP_SRS_UPDATE` updates an existing SRS. It must be followed by exactly three data fields. The first data field contains the outpoint[5] that carries the previous SRS, the second contains the new SRS and the third the correctness proof of the update.

`OP_SNARK_CREATE` moves some normal bitcoins to a private context. It must be followed by three or four data fields: An outpoint that references the SRS, (a reference to) the description of an NP relation R that all subsequent SNARKS that use these coins must obey – needed only if the relevant `OP_SRS_CREATE` did not specify such a relation – an initial statement x and a zk-SNARK that proves that this is a valid statement (i.e., $\exists w : R(x, w) = 1$). Similarly to `OP_SRS_CREATE`, the relation defines the semantics of the state machine to be used by subsequent SNARKs.

`OP_SNARK_UPDATE` updates the state of the private state machine with a new valid statement. Following Bitcoin semantics, It must consume the outputs of the transactions that contain the previous valid statements $x_1, \dots, x_n$. It must be followed by exactly three data fields: An outpoint that references the SRS, a statement x' and a zk-SNARK that proves that the previous statements together with the new statement obey the overarching relation (i.e., $\exists w : R(x_1 \wedge \cdots \wedge x_n \wedge x', w) = 1$). The relation R can be designed so that, under particular circumstances, e.g., when the contract is resolved, x' can be a Bitcoin output that can be consumed by the current Bitcoin Script, therefore converting the private coins back to normal bitcoins.

We note that the SNARK relation can be specified either at the SRS creation or during the conversion of bitcoins to private coins. In the former case coins that are converted in separate transactions can freely interact, but the relation is tied with the specific SRS (or any of its updates) and can never change. In the second case on the other hand, a single SRS can cater to various relations, but coins converted in various transactions cannot interact.

[5] reference to a specific transaction output already in the ledger

A potential specific application of the above system (with the relation defined together with the SRS) is the reproduction of Zcash [4,5] capabilities. We can allow transactions that provide the same privacy guarantees as "shielded" Zcash transactions do. Another example application in which the relation can be defined together iwth the SRS is a private version of the lightning network [8]. An example usecase of defining the relation at conversion is private smart contracts, e.g. gambling or joint savings accounts.

As a precaution against malicious SRSs, full nodes should furthermore keep track of the number of coins paid into and withdrawn from a particular SRS. In order to be valid, the transaction containing the zk-SNARK must withdraw at most as much coins as the ones remaining in the used SRS. This is a simple safeguard that ensures the firewall property [14], i.e. that no bitcoins can be generated out of thin air.

Taproot[6] defines a mechanism for specifying semantics for new opcodes, which we can use to introduce the three new opcodes. In case some of the above requirements cannot be enforced through this upgrade mechanism (e.g. the requirement of a single zk-SNARK opcode per script), we can leverage other update paths, such as increasing the SegWit version or using the "annex" field that Taproot provides.

As this is a soft fork, full nodes with old software will accept all transactions described above as valid. They will also accept transactions with fake zk-SNARKs, so every node is advised to update. In order for these rules to be enforced, a supermajority of the mining power should have such capability activated. An update strategy similar to that used for enabling SegWit[7] can be employed.

3 Open Questions

- Other zk-SNARK systems to consider?
- Possible without forcing all miners to update?
- Missed pitfalls?
- Alternative upgrade paths?
- Obvious optimizations/alternative approaches?
- Tradeoffs of putting the relation in the SRS vs in SNARK creation? Should we provide both?
- Wrong Zcash approaches we would like to avoid?
- How to store the (rather big) SRS and relations on-chain? If impossible, store it elsewhere, trustlessly.

References

1. Nakamoto S.: Bitcoin: A Peer-to-Peer Electronic Cash System (2008)

[6] https://github.com/sipa/bips/blob/bip-schnorr/bip-taproot.mediawiki

[7] https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

2. Androulaki E., Karame G., Roeschlin M., Scherer T., Capkun S.: Evaluating User Privacy in Bitcoin. In Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers: pp. 34–51: doi:10.1007/978-3-642-39884-1_4: URL https://doi.org/10.1007/978-3-642-39884-1_4 (2013)
3. Maurer F. K., Neudecker T., Florian M.: Anonymous CoinJoin Transactions with Arbitrary Values. In 2017 IEEE Trustcom/BigDataSE/ICESS, Sydney, Australia, August 1-4, 2017: pp. 522–529: doi:10.1109/Trustcom/BigDataSE/ICESS.2017.280: URL https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.280 (2017)
4. Ben-Sasson E., Chiesa A., Garman C., Green M., Miers I., Tromer E., Virza M.: Zerocash: Decentralized Anonymous Payments from Bitcoin. In 2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014: pp. 459–474: doi:10.1109/SP.2014.36: URL https://doi.org/10.1109/SP.2014.36 (2014)
5. Hopwood D., Bowe S., Hornby T., Wilcox N.: Zcash Protocol Specification. URL https://github.com/zcash/zips/blob/master/protocol/protocol.pdf (2019)
6. Kerber T., Kiayias A., Kohlweiss M.: Kachina - Foundations of Private Smart Contracts
7. Decker C., Wattenhofer R.: A Fast and Scalable Payment Network with Bitcoin Duplex Micropayment Channels. In Stabilization, Safety, and Security of Distributed Systems - 17th International Symposium, SSS 2015, Edmonton, AB, Canada, August 18-21, 2015, Proceedings: pp. 3–18: doi:10.1007/978-3-319-21741-3_1: URL https://doi.org/10.1007/978-3-319-21741-3_1 (2015)
8. Poon J., Dryja T.: The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. https://lightning.network/lightning-network-paper.pdf (2016)
9. Dziembowski S., Eckey L., Faust S., Malinowski D.: Perun: Virtual Payment Hubs over Cryptocurrencies. In 2019 2019 IEEE Symposium on Security and Privacy (SP): pp. 344–361: IEEE Computer Society, Los Alamitos, CA, USA: ISSN 2375–1207: doi:10.1109/SP.2019.00020: URL https://doi.ieeecomputersociety.org/10.1109/SP.2019.00020 (2019)
10. Lind J., Naor O., Eyal I., Kelbert F., Pietzuch P. R., Sirer E. G.: Teechain: Reducing Storage Costs on the Blockchain With Offline Payment Channels. In Proceedings of the 11th ACM International Systems and Storage Conference, SYSTOR 2018, HAIFA, Israel, June 04-07, 2018: p. 125: doi:10.1145/3211890.3211904: URL https://doi.org/10.1145/3211890.3211904 (2018)
11. Miller A., Bentov I., Kumaresan R., Cordi C., McCorry P.: Sprites and State Channels: Payment Networks that Go Faster than Lightning. ArXiv preprint arXiv:1702.05812 (2017)
12. Zamyatin A., Stifter N., Judmayer A., Schindler P., Weippl E. R., Knottenbelt W. J.: A Wild Velvet Fork Appears! Inclusive Blockchain Protocol Changes in Practice - (Short Paper). In Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers: pp. 31–42: doi:10.1007/978-3-662-58820-8_3: URL https://doi.org/10.1007/978-3-662-58820-8_3 (2018)
13. Maller M., Bowe S., Kohlweiss M., Meiklejohn S.: Sonic: Zero-Knowledge SNARKs from Linear-Size Universal and Updatable Structured Reference Strings. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019: pp. 2111–2128: doi:10.1145/3319535.3339817: URL https://doi.org/10.1145/3319535.3339817 (2019)

14. Gazi P., Kiayias A., Zindros D.: Proof-of-Stake Sidechains. In 2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019: pp. 139–156: doi:10.1109/SP.2019.00040: URL https://doi.org/10.1109/SP.2019.00040 (2019)