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.

Zcash [4,5] is another blockchain with semantics similar to Bitcoin. The main difference is that the former attempts to solve the issue of privacy leakage by employing the use of zk-SNARKs [6,7] for users that wish to use them. At a high level, each such transaction carries a zero-knowledge proof of the fact that it transfers coins between some parties. This proof ensures that no new coins are created out of thin air, but does not disclose neither the value of coins nor the addresses of the implicated parties, creating thus one big anonymity set for all parties that have ever transacted using zcash "shielded"[3] addresses.

Drawing inspiration from Zcash but following a different approach, the aim of this proposal is to enable the use of zk-SNARKs in Bitcoin through a soft fork [8]. Such an extension could bring stronger privacy capabilities to Bitcoin, which is currently the cryptocurrency with the highest market capitalization[4]. This can be achieved with minimal changes to Bitcoin: the new feature will be available through two new Bitcoin Script[5] opcodes.

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.

[3] we borrow the terms "transparent" and "shielded" from Zcash.
[4] https://coinmarketcap.com
[5] https://en.bitcoin.it/wiki/Script

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 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 [9], we therefore choose this as our zk-SNARK system.

Two new opcodes are added to the Bitcoin Script: `OP_SRS` and `OP_SNARK`. 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` generates a new or updates an existing SRS, depending on its arguments. If followed by exactly two data fields, the opcode is a generation of a new SRS. The first data field contains the new SRS and the second the proof of its correctness.

If the opcode is followed by exactly three data fields, the opcode is an update of an existing SRS. The first data field contains the outpoint[6] that carries the previous SRS, the second contains the new SRS and the third the correctness proof of the update.

`OP_SNARK` must be followed by exactly two data fields: An outpoint that references the SRS and a zk-SNARK that is valid based on the specified SRS and all previous proofs done under this SRS (and its previous versions). Within the zk-SNARK system, a player *Alice* can make the following operations in zero knowledge:

– Pay in normal bitcoins to a hidden public key. The SNARK should prove that the hidden public key now owns these private bitcoins. This can only happen by a transaction that transfers bitcoins to the specified SRS.
– Transfer private bitcoins to another hidden public key. The SNARK must prove that *Alice* knows the secret key corresponding to a hidden public key that owns enough private bitcoins. Such transfers can only happen by transactions that move 0 coins from the SRS to the SRS[7].
– Convert private bitcoins to normal ones and move them from a hidden to a conventional public key. The SNARK must prove that *Alice* knows the secret key corresponding to a hidden public key that owns enough private bitcoins. Such transfers can only happen by transactions that move the specified number of coins from the SRS to the conventional public key.

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 [10], i.e. that no bitcoins can be generated out of thin air.

[6] reference to a specific transaction output
[7] miner fees and dust limits should be taken into account here.

Taproot[8] defines a mechanism for specifying semantics for new opcodes, which we can use to introduce `OP_SRS` and `OP_SNARK`. 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[9] 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?
- Wrong Zcash approaches we would like to avoid?
- How to store the (rather big) SRS on-chain? If impossible, store it elsewhere, trustlessly.

References

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

[8] https://github.com/sipa/bips/blob/bip-schnorr/bip-taproot.mediawiki
[9] https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki

6. Bitansky N., Canetti R., Chiesa A., Tromer E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013: pp. 111–120: doi:10.1145/2488608.2488623: URL https://doi.org/10.1145/2488608.2488623 (2013)
7. Groth J.: On the Size of Pairing-Based Non-interactive Arguments. In Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II: pp. 305–326: doi:10.1007/978-3-662-49896-5_11: URL https://doi.org/10.1007/978-3-662-49896-5_11 (2016)
8. 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)
9. 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)
10. 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)