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 [**?**] transactions store permanently the pseudonymous addresses of transacting parties in the clear on the blockchain. As shown by [**?**], 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 [**?**] need active user coordination, are prone to DoS attacks and provide only heuristic privacy guarantees that can be violated by a determined adversary [**?**].

Zcash [**?**] 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 SNARKs [**?**] 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.

The aim of this proposal is to show that it is possible and practical to integrate similar privacy capabilities in Bitcoin through a soft fork [**?**]. Such an extension could bring stronger privacy to Bitcoin, which is the cryptocurrency with the highest market capitalization[4].

2 Proposal

Initially, a special `OP_RETURN` transaction, tx_{CRS}, is added to the blockchain. It contains a CRS, which is generated in a trusted manner, e.g. similar to how the CRS in Zcash was. A specific bitcoin public key $pk_{\top}$ is designated in this process, which will contain all the "shielded" funds.

Subsequently, *Alice* can generate a private bitcoin keypair (pk_A, sk_A) and create a transaction tx_A that moves $c + r + f$ coins to $pk_{\top}$ and contains a randomized commitment to pk_A.

[3] we borrow the terms "transparent" and "shielded" from Zcash.
[4] https://coinmarketcap.com

Consider now the case in which *Alice* wants to pay c coins to pk_B, a public key controlled by *Bob*. Using sk_A, pk_B and pk_{change} (the latter controlled by herself), she can produce a SNARK that proves in zero knowledge that:

- she knows the secret key sk_A corresponding to the public key pk_A, which is commited to in an as-of-yet unspent shielded transaction tx_A,
- the output of tx_A contains exactly the funds transferred $(c + r)$ plus fees f,
- she transfers c coins to pk_B and r coins to pk_{change}.

She can embed this SNARK in a tx_B that moves all but f funds in $pk_\top$ back to $pk_\top$. This results in c coins owned by pk_B and r coins by pk_{change}. Bitcoin nodes should verify the validity of the SNARK and that the bulk of the funds remains in $pk_\top$.

When *Alice* wants to move d funds back to a "transparent" address pk_t, she follows the same steps as above, with the difference that the third zero knowledge bullet is omitted and d funds are moved to pk_t instead of $pk_\top$. In this case, Bitcoin nodes should only check the validity of the SNARK.

The SNARK system we choose is TODO [**?**]. For extra flexibility, we may demand that all shielded transactions contain the hash of tx_{CRS} so that other independent CRSs can be defined as well.

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 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 [**?**] can be employed.

3 Upgrading approach

There are various update mechanisms available in Bitcoin (especially after Taproot [**?**]). We now discuss some available options. All options below can be extended to include the hash of tx_{CRS}.

3.1 New SegWit version

Transactions with SegWit version 3 (or whichever is the lowest unused version number) are all accepted by current nodes. We could demand that transactions with SNARKs carry this a version number at least equal to this. We are largely free to define the syntax and semantics of such transactions as we please, so we can copy the Zcash semantics.

3.2 New opcode

Taproot defines a mechanism for specifying semantics for new opcodes. We can define `OP_SNARK` that reads the next data field as a SNARK. This way the SNARK semantics can be combined with existing Bitcoin script.

3.3 New Taproot address type

Taproot allows for the definition of alternative semantics in the location currently occupied by public keys. We could replace this space with a SNARK and use the "annex" field for additional SNARK bits if needed.

4 Open Questions

- Which are the best SNARKs? Why do we choose TODO?
- Possible without forcing all miners to update?
- Missed pitfalls?
- Alternative upgrade paths?
- Obvious optimizations/alternative approaches?
- Wrong Zcash approaches we would like to avoid?