Transferring Bitcoin Among Altchains

Paul Sztorc (truthcoin@gmail.com)

What follows is a technical problem I encountered while designing Truthcoin, which I was unable to solve. I have no training/experience in crypto, so I just ignored the problem at the time but now I reopen it in the hopes that someone will find an answer.

Problem Description

Satoshi taught us that a blockchain can solve many protocol problems: software can easily check arbitrary block validation rules, and a simple selection rule (cumulative difficulty) can force a unique protocol history onto network participants (despite their varying level of participation).

Yet blockchains only carry information one way. We can use our phones to alter the blockchain (with a Bitcoin transaction), but the blockchain all by itself can't alter our phones. To do this we must run applications on our phones that are designed to use blockchain data, for example a 'Bitcoin Call App' that makes a pre-specified phone call when a certain transaction is made.

Similarly, a Second Blockchain (B2) can easily be programmed to read the Bitcoin blockchain (B1), but alone cannot write to the Bitcoin blockchain. So depositing Bitcoin into S2 is easy (as S2 can scan the blockchain for the deposit-transactions that S2 users claim are there), yet withdrawals are not (as S1 is not scanning S2 at all, and cannot be forced to). To complete the transfer, we need (I assume) a tool analogous to 'Bitcoin Call App' which watches S2 for withdrawal requests and signs/broadcasts them for S1.

Thus the problem: **Precisely how should a distributed piece of software sign Bitcoin transactions?** I assume that partially signed[1] withdrawal transactions are embedded within an S2 block, but this may not be necessary. I also assumed that placing the private key into a piece of open source software would necessarily be insecure (which might not be true).

Working Solution : The Lagged Oracle[2]

The Lagged Oracle (LO) watches the longest valid chain of S2 (possibly checking for forks, version changes, or execution problems), and, upon the discovery of each new block, signs the withdrawal

[1] Obviously, users must authenticate a request for withdrawal. These requests would in some way require at least 2 signatures, although Signer might create 'copy' 1 of 1 Bitcoin transactions from scratch and sign them (once).
[2] Of the 7 potential solutions I proposed in the Truthcoin whitepaper (solely for the sake of completeness), this is my uneducated guess at which is optimal.

transactions present in the block N=20 blocks beneath.[3] Thus orphaned blocks are irrelevant (as resigning transactions that have already been signed has no effect).

Main Questions

- Can we derive (and use) private keys such that they are never known? Can I prove that I don't know/can't use a private key?
- Can we hide private keys in an application we widely distribute? To what extent could this oracle be open-source/trustworthy?
- Can we use randomness (chaotic inputs, or iterative randomness with block-hashes/nonce) to derive keys? Can such a piece of software copy itself or be copied?

Advanced Questions

- Possibly a software update to S2 or fork will cause the oracle to become confused. What steps can we take to anticipate and plan for this? Should we allow miners to 'pause' the oracle (for example, if 75 of the last 100 blocks contain a pause vote)?
- Can I build a S2 <-> S1 "exchange" into the lower blockchain (S2)? As what has been proposed above is essentially an escrow account, this should be possible.

Details

Withdrawal transactions in S2 should probably have a Bitcoin fee. In fact I would expect them to look exactly like normal Bitcoin transactions.

Other Implications

If successful, this method would greatly assist in the creation of so-called DACs (Distributed Autonomous Corporations), as it would allow individuals to send and receive Bitcoin to new blockchains. Bitcoin deposited with the oracle would become 'credits' on a second blockchain, with its own flexible rules about how to rearrange those credits, which could then be withdrawn as Bitcoin. This could then create a whole family of distributed services which only run on blockchain currency.

[3] S2 could officially 'lock in' the deposit after, say 10 blocks, so that users cannot exploit orphaned blocks to double-withdraw.