Using Chains for what They're Good For

Andrew Poelstra

usingchainsfor@wpsoftware.net

Scaling Bitcoin, November 5, 2017

On-Chain Smart Contracting

- Bitcoin (and Ethereum, etc.) uses a scripting language to describe smart contracts and enforce their execution.
- These scripts must be downloaded, parsed, validated by all full nodes on the network. Can't be compressed or aggregated.
- Valid execution can't be assured until the transaction is confirmed: unpredictable, long delays.

On-Chain Smart Contracting

- Script verification rules must be agreed upon by all participants.
- The details of the script are visible forever, compromising privacy and fungibility.
- Miners can see contract contents before including them, and may not want to (potential legal liability, external incentives, etc.)

On-Chain Smart Contracting

- Most contracts need only one thing from the blockchain: **an immutable ordering of commitments to prevent double-spending.**

Execution vs Verification

- Blockchain validators must check whether scripts execute successfully.
- This is strictly easier than actually executing the scripts (Post's Theorem). May be assisted by a witness.
- In crypto, we talk about (zero-knowledge) arguments that some script returns true. May be assisted by a transcript.

Verifiability vs Public Verifiability

- Blockchain verifiers must check that coins are only spent with correct authorization, and only once.
- They don't necessarily need to know what "correct authorization" means, they just need to agree on it.
- Consider moving coins to a multisig output, where multiple distrusting signers check external conditions before signing to move them to their final destinations (Gibson 2017).

Scriptless Scripts

- Suppose these distrusting signers want to enforce conditions on each other: a "smart contract".
- The script paradigm demands they reveal witnesses to their conditions on the public chain, along with their signature.
- But *what if the signature was the witness?* Then the blockchain would only need to check a multisignature, or should I say. . . a scriptless script.

Scriptless Scripts

- Scriptless scripts: magicking digital signatures so that they can only be created by faithful execution of a smart contract.
- Limited in power, but not nearly as much as you might expect.
- Mimblewimble is a blockchain design that supports only scriptless scripts, and derives its privacy and scaling properties from this.

Schnorr multi-Signatures are Scriptless Scripts

- By adding Schnorr signature keys, a new key is obtained which can only be signed with with the cooperation of all parties.
- The parties must interact to sign: first they agree on the message and nonces, then they contribute to the signatures.
- (Don't try this at home: some extra precautions are needed to prevent adversarial choice of keys.)

Adaptor Signatures

- Instead use another ephemeral keypair (t, T) and treat T as the "hash" of t.
- When doing a multi-signature replace the old nonce R with $R + T$, and now the signature s must be replaced by $s + t$ to be valid.
- Now the original s is an "adaptor signature". Anyone with this can compute a valid signature from t or vice-versa. They can verify that it is an adaptor signature for T, no trust needed.

Atomic (Cross-chain) Swaps

- Parties Alice and Bob send coins on their respective chains to 2-of-2 outputs. Bob thinks of a keypair (t, T) and gives T to Alice.
- Before Alice signs to give Bob his coins, she demands adaptor signatures with T from him for *both* his signatures: the one taking his coins and the one giving her coins.
- Now when Bob signs to take his coins, Alice learns t from one adaptor signature, which she can combine with the other adaptor signature to take *her* coins.

Basic Lightning

- Suppose Alice is paying David through Bob and Carol. She produces an onion-routed path

$$\text{Alice} \to \text{Bob} \to \text{Carol} \to \text{David}$$

  and asks for public keys B, C and D from each participant.

- She sends coins to a 2-of-2 between her and Bob. She asks Bob for an adaptor signature with $B + C + D$ before signing to send him the coins.

- Similarly Bob sends coins to Carol, first demanding an adaptor signature with $C + D$ from her. Carol sends to David, demanding an adaptor signature with D.

Features of Adaptor Signatures

- Adaptor signatures work across blockchains, even if they use different EC groups, though this requires a bit more work.
- After a signature hits the chain, anyone can make up a (t, T) and compute a corresponding "adaptor signature" for it, so the scheme is deniable. It also does not link the signatures in any way.
- Adaptor signatures are re-blindable, as we saw in the Lightning example. This is also deniable and unlinkable.

Thank You

Andrew Poelstra <whattheyregoodfor@wpsoftware.net>