Scriptless Scripts: How Bitcoin Can Support Smart Contracts Without Smart Contracts

by **Aaron van Wirdum** Nov 27, 2017 2:19 PM EST

Bitcoin's capacity is limited. Meanwhile, smart contracts can be resource intensive. So even though Bitcoin has always supported basic smart contract functionality, the two have never been a natural match.

But a recent topic of research spearheaded by **Blockstream** mathematician Andrew Poelstra could help fix this. Recently presented as key part of his **presentation** at **Scaling Bitcoin Stanford**, "Scriptless Scripts" have the potential to completely move certain smart contracts off of Bitcoin's blockchain —

Bitcoin and Smart Contracts

Smart contracts, first proposed by digital currency veteran Nick Szabo in the 1990s, are essentially self-executing contracts. Most typically, they send money from someone to someone else if specific conditions are met. For example, if someone streams a song, a payment is automated from the streamer to the artist.

While smart contracts are often associated with "second generation" blockchains like Ethereum, Bitcoin has always supported basic smart contracts, too. In a way, any Bitcoin transaction is technically a smart contract: Funds are typically moved on the condition that a valid cryptographic signature is provided. Slightly more advanced smart contracts — such as **multisig** and **timelocks** — are used to enable second-layer protocols like **the Lightning Network**.

But there are problems with blockchain-based smart contracts. For one, as they grow more complex, they require more resources to execute. This is especially problematic because all nodes on the network need to execute the contract — not just the parties involved with the contract itself.

This network-wide execution also means that the parties involved have no privacy regarding what their smart contract entails: The entire network will know exactly what it looks like. By extension, this is bad for fungibility as well. If the smart contract is unpopular for some reason, the funds involved — publicly visible on the blockchain — are tainted.

As smart contracts become more complex, they can even become a security risk. Alternative software implementations might, for example, interpret details of contracts slightly differently, making it harder to keep all nodes on the network in consensus. And potential bugs in these smart contracts are public as well, which increases the chance of hacks.

But Poelstra, among others, thinks that many of these problems can be solved by actually moving the bulk of contracts off of the blockchain. Instead of having all nodes on the network calculate the entire smart contract, only the parties involved with the contract should perform this function.

The trick is to ensure that the rest of the network does still correctly enforce the outcome of the contract: The payment must only be made if the required

conditions are met.

Schnorr

Poelstra originally began researching “Scriptless Scripts” (a phrase he also coined himself) in the context of the **Mimblewimble** protocol. This stripped down version of Bitcoin offers more privacy and better scalability but does not support script: the bits of code embedded in Bitcoin transactions that allow for most basic smart contract features.

So, Poelstra figured out how to get the utility offered by scripts without actually requiring them on the blockchain: Scriptless Scripts.

The key to Scriptless Scripts is that (fairly) regular cryptographic signatures can indirectly reveal something that’s not part of the transaction that includes the signature. In other words, when someone signs to validate an ordinary Bitcoin transaction, it holds that a smart contract that is not hosted on the blockchain still executes faithfully.

This is made possible with **Schnorr signatures**. These types of signatures are not yet implemented on the Bitcoin protocol, but it is possible that they could be deployed within a year or so from now.

Schnorr signatures allow for signature aggregation; several signatures can be mathematically combined into a single signature. And, importantly for this use case, this math is “linear.” This basically means it’s possible to perform relatively straightforward but very expressive math on these signatures.

Oversimplified, it works something like this:

Private keys and signatures are, of course, really just numbers, where the latter is derived from the former. Since this is a simplified example, let’s say one private key looks like 10, and half of the Schnorr signature derived from that private key looks like 10000. And the other private key looks like 15, with the second half of the Schnorr signature looking like 15000. In this simplified example, the Schnorr signature would then look like 25000 (or 10000 + 15000).

And since both halves of the signature are just numbers, it’s possible to perform math between them. For instance, in this simplified example, the difference between these halves is 5000 (or 15000 – 10000).

While the reality is more complex, Schnorr’s linearity allows for several of these kinds of math “tricks.”

The Smart Contract

Now let’s say that a streamer wants to listen to a song by an artist. The artist has the right to this song, and it will play for the streamer if (and only if) the artist's signature is provided to a server where the song is hosted. Since we’re simplifying, let’s say that this “song signature” looks like 7000. The streamer is willing to pay the artist one bitcoin for this song signature, to listen to the song. (He wants to listen to the song really badly.)

In this simplified example, the streamer and the artist can automate this trade by doing two things. First, they create a fairly normal Bitcoin transaction that sends one bitcoin from the streamer to the artist, if the streamer and the artist both provide their half of a Schnorr signature to create a full Schnorr signature. (In reality, this step requires some extra safety precautions to ensure no one loses money, but it is relatively simple.)

The next step is where it gets a bit more complex.

The artist knows what her half of the Schnorr signature looks like; since we’re simplifying, let’s say it looks like 8000. And she knows what her song signature looks like: 7000. As such, she can calculate the difference between these two: 1000. This is called the adaptor signature. The artist then hands this adaptor signature — 1000 — to the streamer.

Here’s where the cryptographic magic happens.

By modifying the ordinary signature verification method, the streamer can actually verify that the adaptor signature he just received (1000) is indeed the difference between the artist's half Schnorr signature and her song signature — even though the streamer does not have access to either signature yet. (And thanks to

cryptographic tricks called “zero-knowledge proofs,” something like this can actually be done in a surprisingly broad range of scenarios, not just in signatures as this example portrays.)

Now, having verified that the adaptor signature (1000) checks out, the streamer can, in turn, give his half of the Schnorr signature to the artist because once the artist uses the streamer’s half to create a full signature and broadcasts this over the Bitcoin network, she automatically reveals her half of the Schnorr signature (8000) to the streamer as well.

Using the artist’s half of the Schnorr signature, the streamer can now subtract the adaptive signature: 1000. By subtracting the adaptive signature from the artist’s half Schnorr signature (8000 – 1000) the streamer indeed learns the artist’s “song signature”: 7000. And now he can listen to the song.

In other words, by broadcasting the transaction that pays her one bitcoin, the artist automatically sells the streamer the signature: a smart contract.

From the perspective of the blockchain — that is, the rest of the world — the transaction is quite regular. Nothing about the smart contract, other than the “settlement transaction,” is ever recorded on the blockchain. No one will ever know that an underlying contract was executed — never mind what song the streamer listened to — and the contract-related data never needs to be calculated or stored by anyone other than the parties involved.

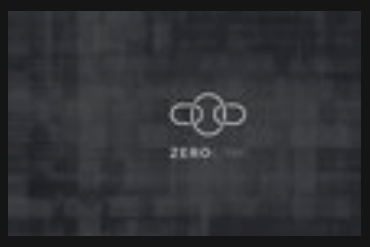
*To see Poelstra’s Scaling Bitcoin presentation that includes Scriptless Scripts, “Using the Chain for What Chains Are Good For”, click **here**. An alternative, in-depth explanation of Scriptless Scripts was published by JoinMarket developer Adam “Waxwing” Gibson and can be found **here**.*

#SMART CONTRACTS #MIMBLEWIMBLE #SCALING #PRIVACY #BITCOIN #SCRIPTS

RELATED ARTICLES:

WikiHow Users Can Now Secure Their Online Identities with Civic

by Tanzeel Akhtar



HiddenWallet and Samourai Wallet Join Forces to Make Bitcoin

by Aaron van Wirdum

Latest Cryptocurrency Exchange Hack Highlights Need for Better Security

by Darryn Pollock

Subscribe

Store

Call for Writers

We are always looking for talented writers to join our team. If you have an article you'd like to have published to our audience please reach out to

editor@bitcoinmagazine.com