[Bitcoin-ml] BLS Signatures

Jonald Fyookball jonaldfyookball at outlook.com
Fri Apr 6 23:12:36 UTC 2018

- Previous message: [Bitcoin-ml] BLS Signatures
- Next message: [Bitcoin-ml] BLS Signatures
- **Messages sorted by:** [date] [thread] [subject] [author]

Ladies and gentlemen, these may finally be the droids
we've been looking for, as far as a fungibility solution
for Bitcoin Cash.

I cannot overstate my enthusiasm for the
One Way Aggregate Signatures (OWAS)!

To grasp why this is so significant, one has to
understand the current limitations that we're saddled
with, and their consequences.

Thus far, all joining protocols in Bitcoin (both BTC and BCH),
including CashShuffle, have a hard time gaining momentum
primarily because of a lack of network effect. Finding
join partners is difficult when participation is low
and requires coordination. Even as a protocol like
CashShuffle gains usage, it will increasingly face DOS attacks.
Those DOS attacks can be somewhat mitigated by blame protocols,
but a motivated attacker could still cause significant disruption,
which might provoke messy defensive measures (perhaps security
deposits),
or simply cause joined transactions to be unreliable or annoying to use.
These things discourage widespread use and slow down adoption.

Yet, widespread use of joined transactions is exactly
what is required to create a high level of fungibility.
When nearly everyone's transactions are jumbled,
the concept of "taint" essentially becomes meaningless.

The "non-interactive" property offered by OWAS is a game-changer.
Traditional coinjoins use the ALL|ANYONECANPAY SIGHASH type, so
that a user can sign for their input and all of the outputs.
The key problem is that everyone involved in the transaction
has to sign separately, and if any participant fails to sign,
the transaction is invalid and must be constructed again with
a different set of participants.

The other SIGHASH types aren't helpful. NONE|ANYONECANPAY isn't useful since funds could be stolen (Alice would have no way to guarantee her output would be included), and SINGLE|ANYONECANPAY can't be used since the input would be linked to the output.

Alternative implementations of traditional join schemes can be envisioned, but they all seem to run into the same basic constraints. For instance, what if instead of publishing inputs and then coordinating to attach outputs (as is done in CashShuffle), users publish outputs first and then sign various transaction sets with their inputs? This would allow for less coordination, but creates an even worse DOS attack vector, as an arbitrary number of fake outputs could be created without cost.

The DREAM SCENARIO is that a user could create a single input, single output transaction, which we can think of as a "slice" of a soon-to-be larger transaction, and another party could combine this slice with other slices, all without the user having to re-sign the transaction after creating their slice.

That seems to be impossible given the current set of SIGHASH types, because you have to sign for your own input separately along with all the outputs, and you don't know the other outputs without coordinating with the other participants.

Even if we had some "cryptomagic" that could re-sign transactions on behalf of multiple users (which we don't have), we would still have to trust the mixing agent to protect anonymity, which is not only non-ideal, but precludes a distributed solution.

Going a step farther, even if this mixing agent didn't know what it was signing (for example using blinded signatures), it would still need to be trusted with the anonymity because it could try various combinations and see what the finished transaction looks like after its published on a blockchain. Also, a blinded scheme would likely require creating a shared secret as a prerequisite step, introducing more coordination steps and friction.

However, our desired "dream scenario" DOES seem to be achievable using OWAS if we're willing to slightly modify the protocol to allow a bit more flexibility in the transaction types. I'm not sure how invasive the changes need to be, but in order to get a true fungibility solution, it will probably be worth it.

OWAS, within the context of a flexible transaction scheme, can meet our goals because the message digests being signed for can be combined in a multiplicative way, and therefore the inputs and outputs can appear in any order.

It should also be noted that we cannot overestimate the
negative impact of friction in the user experience
(which OWAS would remove). All we have to do is look
at the BTC community for proof.

In the Bitcointalk thread where I believe the OWAS paper was first
announced:
https://bitcointalk.org/index.php?topic=290971.0

The only comments were from Socrates1024 and Greg Maxwell.

Maxwell clearly understands the main benefit when he states
"The OWAS removes the need for any bidirectional interaction
or cooperation of the signing parties. A clear improvement for privacy".
Yet he also dismisses it, saying "I believe a cryptographic approach
is unnecessary".

Given the almost zero progress the BTC community
has made in actually achieving commonplace (let alone widespread)
joined transactions, we can see that this belief was mistaken,
at least if fungibility was the goal.

But their loss can be our gain. Languishing, mostly unnoticed,
these ideas come to our community awareness at a perfect time.
It is an opportunity ripe for the picking.

Finally, I think allowing users the unhindered ability to
(optionally) join their transactions provides the perfect level
of privacy and fungibility for Bitcoin Cash. If we go
"too dark" like XMR, there will be greater hostility
from governments and other entities under state and social pressure.

Strategically, if our goal is worldwide adoption, that's an important
consideration.

In a world where joined transactions are commonplace, there
is still a public ledger that can allow the flow of funds
to be observed. There is still the ability to show the origin of funds.

There is still the plausible deniability that is already present today
(but to a slightly larger degree). One could even make the argument
that
illicit transactions are better able to be investigated, because
criminals
wouldn't go out of their way as often to using extra-ordinary means.

And most importantly, fungibility is protected because taint is
washed across the entire system, while those who desire even
greater privacy always have additional options.

From: bitcoin-ml-bounces at lists.linuxfoundation.org <bitcoin-ml-bounces at lists.linuxfoundation.org> on behalf of Jim Posen via bitcoin-ml <bitcoin-ml at lists.linuxfoundation.org>
Sent: Friday, April 6, 2018 5:38 PM
To: Bitcoin and related protocol coordination
Subject: [Bitcoin-ml] BLS Signatures

I want to raise the idea of adding support for BLS signatures to the scripting language. The idea has been around for a while in the Bitcoin community, and I'm interested to see how people feel about the tradeoffs.

For those not familiar, BLS signatures [1] were invented in 2004 by Dan Boneh, Ben Lynn, and Hovav Shacham and have a number of very nice properties. I'm probably not going to do them full justice, but they 1) are smaller than ECDSA signatures (48 bytes vs ~70-72 bytes DER-encoded for equivalent security targets) 2) are provably non-malleable 3) support non-interactive signature aggregation and 4) support unique threshold signatures. Point 3 is the most important, so I'll elaborate.

Say there are N parties with different public keys providing signatures over different messages (the messages must be different though). So they each provide (Message_i, PubKey_i, and Sig_i), where the signatures are produced with their respective private keys. Anyone can then take all of the signatures and add them together to get a single, constant sized signature covering all pairs (Message_i, PubKey_i) simultaneously. What this means for Bitcoin is that all signatures in all inputs in all transactions in a block could be merged by the miner into a single 48 byte signature covering the entire block. This would allow for massive potential savings in block space.

Another very interesting property of signature aggregation is that is allows for unlinking of inputs and outputs in a transaction through a construction called one-way aggregate signatures (OWAS) [2]. With some different sighash types, one could create secure transactions that can be merged by a third party into an aggregate transaction that obfuscates the linkability between inputs and outputs. Essentially, this is non-interactive CoinJoin.

These signatures work using pairing-based cryptography [3], which is where the tradeoffs come in. Firstly, BLS signatures are defined over pairing-friendly elliptic curves, which Bitcoin's standard curve secp256k1 is not one of. So first, a different elliptic curve would have to be chosen and the best candidate I've heard of from research by the ZCash team is BLS12-381 [4], which has larger public keys. Secondly, signature verification is definitely slower than with secp256k1. On my machine the secp256k1 verification benchmarks show ~75 us per ECDSA verification with libsecp256k1 and ~7 ms per pairing operation with https://github.com/ebfull/pairing. Third, the signatures depend on

stronger cryptographic assumptions than ECDSA (weaker assumptions in cryptography are preferred). That said, the additional assumptions have held up for decades at this point.

What is the state of BLS usage in cryptocurrency today? Chia, the proof-of-space-time blockchain project by Bram Cohen, is planning to use BLS signatures in the scripting language and propose a BIP [5]. Dfinity is another project that is using BLS as a core part of their consensus algorithm, though I'm not sure about it's usage at the VM layer. Both of these teams are working on performance optimizations for BLS implementations [6].

In terms of deployment, this would require significant research into curve selection, performance, etc. Aside from that, I imagine the scope of changes outside the script interpreter is fairly limited. It'd probably be worth adding a new field to unconfirmed transaction and block structures.

For a better primer on BLS signatures, see Benedikt Bunz's talk from Scaling Bitcoin 2017 [7]. Also for some newer research on public key aggregation (further space savings) see [8].

So... what do people think?

[1] https://en.wikipedia.org/wiki/Boneh%E2%80%93Lynn%E2%80%93Shacham
[2] https://download.wpsoftware.net/bitcoin/wizardry/horasyuanmouton-owas.pdf
[3] https://en.wikipedia.org/wiki/Pairing-based_cryptography
[4] https://blog.z.cash/new-snark-curve/
[5] https://twitter.com/bramcohen/status/972208962278797312
[6] https://dfinity.org/tech
[7] https://www.youtube.com/watch?v=LDF8bOEqXt4&t=3h1m54s
[8] https://crypto.stanford.edu/~dabo/pubs/papers/BLSmultisig.html
-------------- next part --------------
An HTML attachment was scrubbed...
URL: <http://lists.linuxfoundation.org/pipermail/bitcoin-ml/attachments/20180406/036fe442/attachment-0001.html>

- Previous message: [Bitcoin-ml] BLS Signatures
- Next message: [Bitcoin-ml] BLS Signatures
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the bitcoin-ml mailing list