TL;DR

: Thanks to @JustinDrake's construction in Bitwise XOR custody scheme, we can replace the "mix" function in the Proof of Custody scheme (currently SHA256) with any PRF that produces as little as only one bit of output. The Legendre PRF does exactly that, and is efficient to compute both directly and in a direct and MPC setting, making it the ideal candidate.

Background

: Proof of Custody is a scheme for validators to "prove" that they have actually seen the block data for a crosslink they are signing on. In order to do this, they commit to a single bit upon signing an attestation. If this bit is incorrect, they can be challenged using the Proof of Custody game (previous design here: https://github.com/ethereum/eth2.0-specs/issues/568, for a much improved version that only needs one round in most cases see Justin's post Bitwise XOR custody scheme)

One open problem is to find a better candidate for the "mix" function. It is currently based on a SHA256 hash, however, SHA256 is very complex to compute in a secure Multi-Party Computation (MPC). One of the design goals of Ethereum 2.0 is, however, to make the spec MPC-friendly, so that it is easy for (a) validator pools to be set up in a secure, trustless manner and (b) allow one-party validators to spread their secret across several machines, reducing the risk of secrets getting compromised.

To replace the "mix" function, we are looking for

An MPC-friendly pseudo-random function (PRF, https://crypto.stanford.edu/pbc/notes/crypto/prf.html) F_K(X)

(F: {0,1}^n \times {0,1}^s \rightarrow {0,1}^m

), where

- K should be shared among several parties, and none of the parties should be able to infer K
- n can be any value, but larger input sizes that preserve the pseudo-randomness would be preferred (The function needs to be run on a total input of ca. 2-500 MBytes, which can be split into chunks of n bits, and run in ca. 1s)
- s (length of K) is 96*8 = 768
- m is arbitrary and can be as little as 1 bit

Legendre PRF

The Legendre symbol \left(\frac{a}{p}\right)

is defined as

- -1

if a

is not a quadratic residue \pmod p

- 1

if a

is a quadratic residue \pmod p

except if

- a \cong 0 \pmod p

then it is defined as 0

.

The Legendre symbol can be explicitly computed using the formula

\displaystyle \left(\frac{a}{p}\right) = a ^ \frac{p-1}{2} \pmod p

The Legendre PRF was suggested by Damgård [1] and is defined by taking a=K+X

, where K

is the secret and X

the PRF input. While the range of this function is {-1,0,1}

, the output 0

only happens if K+X \cong 0 \pmod p

, so effectively we can consider the Legendre PRF to produce one bit of output per given input (which can be as large as the prime p

chosen; in our case, it would be natural to choose a prime p

of similar size to a signature, which would be 768 bits, effectively covering 768 bits of input in every round).

Complexity

Direct (cleartext) computation

Computing the Legendre symbol is not a major concern in terms of computational complexity – according to [2], table 3, the Legendre PRF was able to process ca. 285 MByte/s input data using a width of 256 bits (4 cores i7-3770 3.1 GHz). This is about half the performance of SHA256 (cf https://en.bitcoin.it/wiki/Non-specialized_hardware_comparison#Intel).

MPC-friendlyness

The hard part of MPCs are multiplications, which require communication. The Legendre PRF performs exceptionally well, requiring only two multiplications to evaluate privately (with the output shared among participants) [2].

In comparison, SHA256 is very hard to compute inside an MPC, as it requires tens of thousands of multiplication (see [3] for a benchmark using 29000 AND gates). So the Legendre PRF would be a huge performance improvement.

Cryptographic assumption

The original assumption by Damgård [1] is that consecutive outputs of the function (i.e., X

, X+1

, X+2

, …) cannot lead to prediction of the next output or the secret K

in a computationally efficient way. (The “Shifted Legendre” problem).

While most cryptographic “computational hardness” assumptions (e.g. RSA problem, Discrete logs, …) cannot be proven, they have been tested by years of research in the cryptographic community. Unfortunately, not as much research is available for the Shifted Legendre assumption and it has to be treated somewhat carefully.

However, I argue that using a careful design, we actually do not have to rely on this assumption:

1. In the direct computation, it is actually irrelevant if the secret K

can be inferred and/or if further values of the Legendre PRF can be predicted from any number of outputs, because at the time that the outputs have to be revealed, the secret K

would already have been revealed.

1. When doing it inside an MPC, as long as we are also integrating the XOR of all bits inside the same MPC, the PRF bit will not be known to any MPC participant and only the XOR of all bits will become public. No inference on the secret can be performed on a single output bit. (I am assuming that doing these XORs inside an MPC will not be prohibitively expensive, I would welcome input from someone who knows more about MPCs on this; I found this resource which claims it’s essentially “free”: https://crypto.stackexchange.com/questions/44262/why-xor-and-not-is-free-in-garbled-circuit)

This means for safety, we are not actually relying on any cryptographic assumptions. However, we still want it to be a “good proof of custody” PRF. For this we actually need a somewhat different assumption:

Proof of Custody assumption

: It is not possible to share a derivative D(K)

of the secret K

in a way that will allow (efficient) computation of \left(\frac{K+X}{p}\right)

, but derivation of K

is impossible (or computationally hard).

Looking at the definition of Legendre, this feels likely true, but it would probably warrant someone with cryptographic training thinking about this. (I actually think it is not trivial to see that the equivalent assumption for SHA256 or AES is true)

Alternatives

A conservative alternative PRF would be AES, which is a well tested cryptographic standard. It can be implemented using 290 multiplications [2] and so performs a lot worse than Legendre in the MPC setting, but still 100 times better than SHA256. The direct “cleartext” computation of AES is orders of magnitudes faster than both SHA and Legendre, and it may be worth having a look at it in its own right; for example, the “validator shuffling” problem currently effectively uses SHA256 as a PRF (https://github.com/ethereum/eth2.0-specs/blob/91a0c1ba5f6c4439345b4476c8a1637140b48f28/specs/core/0_beacon-chain.md#get_permuted_index), but AES could do this job with much less computational complexity. I wonder if there are more cases where we default to hash functions although a PRF would be enough.

[1] On the randomness of Legendre and Jacobi sequences, Damgård, CRYPTO 88, https://link.springer.com/content/pdf/10.1007%2F0-387-34799-2_13.pdf

[2] https://eprint.iacr.org/2016/542.pdf

[3] http://orbit.dtu.dk/files/128048431/492.pdf