Towards threshold hash-based signatures for post-quantum distributed validators

Beam call #2 - 28.03.2025

Distributed validators today

- Distributed validators (DVs) currently take advantage of the fact BLS natively supports threshold signatures

- There is no difference between the generation of a partial signature share vs a non-threshold signature => can be leveraged to build middleware DV solutions between validator and consensus clients

- In case of hash-based signatures (HBSs) the above does not hold anymore!

Threshold HBSs

- Threshold HBSs can be built using SNARKs
  - given a *k*-of-*n* setting the aggregator can generate a proof attesting that it verified *k* distinct signatures over the same message and that signers are part of the quorum
  - However SNARK-based aggregation could then be problematic as threshold signatures are not raw hash-based signatures but contain a proof as well!

- Would the computation of HBSs over MPC be realistic?

MPC-friendly instantiations

- Using a prime field defined by p s.t. $\gcd(3,p-1) = 1$ is desirable as it lowers the number of multiplications in Poseidon2 (e.g. Koala Bear for 31-bit prime fields)

Table 3: Generalized XMSS instantiations with Poseidon2 over a 31-bit prime field. The reported number of permutation calls only considers hash chains during signature generation. For signature sizes, we consider two different leaf numbers, namely $L \in \{2^{18}, 2^{20}\}$. Regarding encodings, we refer to the original publication [DKKW25] for more details.

Encoding	Parameters		Sig. size (KiB)		Perm. calls
	w	chunks	$L = 2^{18}$	$L = 2^{20}$	(average case)
W	1	163	4.97	5.03	81
	2	82	2.75	2.81	123
	4	42	1.66	1.72	303
	8	22	1.11	1.34	2676
TSW ($\delta = 1$)	1	155	4.75	4.81	78
	2	78	2.65	2.7	117
	4	39	1.58	1.64	293
	8	20	1.06	1.27	2550

Benchmark using the MP-SPDZ framework

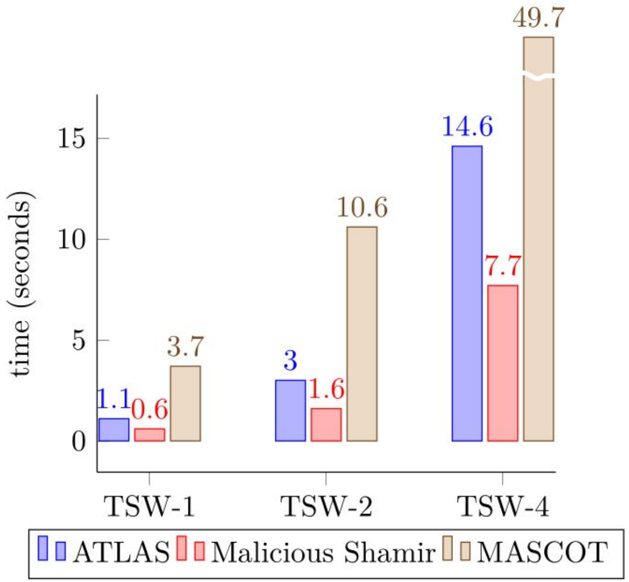

Figure 1: MP-SPDZ benchmark results for Poseidon2 hash chains calculations over MPC (online phase only) to sign a single message. Timing results are averaged over 10 runs in a network with 30ms delay.

Future work

- Identify the right security model to pick the most efficient MPC protocol (malicious two-thirds honest majority?)
- Study time-memory tradeoffs
- More benchmarks (DKGs ?)