# 24.09.24 Mithril MSP and Benchmark

Xun Zhang  Wuyun Siqin  Bingsheng Zhang
Zhejiang University, CHN
22221024@zju.edu.cn  3210101763@zju.edu.cn  bingsheng@zju.edu.cn

September 24 2024

## 1 Mithril MSP

There are two versions of MSP algorithms in the Mithril paper. We omitted the check of MSP-PoP( a multisignature based on Boneh Lynn Shacham(BLS) signatureswith proofs of possession), since this part will not be included in the relations.

The first one is $\mathsf{MSP.AKey}(\mathbf{mvk})$, and $\mathsf{MSP.ASig}(\boldsymbol{\sigma})$:

- $\mathsf{MSP.AKey}(\mathbf{mvk})$: Takes a vector $\mathbf{mvk}$ of (previously checked) verification keys and returns an intermediate aggregate public key $ivk = \prod mvk_i$
- $\mathsf{MSP.ASig}(\boldsymbol{\sigma})$: Takes as input a vector $\boldsymbol{\sigma}$ and returns $\mu \leftarrow \prod_1^d \sigma_i$,

The second one is is $\mathsf{MSP.BKey}(\mathbf{mvk}, \boldsymbol{e_\sigma})$, and $\mathsf{MSP.BSig}(\boldsymbol{\sigma})$:

- $\mathsf{MSP.BKey}(\mathbf{mvk}, \boldsymbol{e_\sigma})$: Takes a vector $\mathbf{mvk}$ of (previously checked) verification keys and weighting seed $\boldsymbol{e_\sigma}$, and returns an intermediate aggregate public key $ivk = \prod mvk_i^{e_i}$, where $e_i \leftarrow \mathrm{H}(i, e_\sigma)$.
- $\mathsf{MSP.BSig}(\boldsymbol{\sigma})$: Takes as input a vector of signatures $\boldsymbol{\sigma}$ and returns $(\mu, \boldsymbol{e_\sigma})$ where $\mu \leftarrow \prod \sigma_i^{e_i}$, where $e_i \leftarrow \mathrm{H}_\lambda(i, e_\sigma)$ and $e_\sigma \leftarrow \mathrm{H}_p(\sigma)$.

The reason why Mithril use $\mathsf{MSP.BKey}(\mathbf{mvk}, \boldsymbol{e_\sigma})$ and $\mathsf{MSP.BSig}(\boldsymbol{\sigma})$ is that:

*which enforce more stringent checking than that of standard multisignatures by utilizing the short random exponent batching of Bellare et al. [5]. The difference from standard multisignature aggregation (via* MSP.AKey *and* MSP.ASig*), is that the randomized check will fail with overwhelming probability if any of the individual signatures is invalid, whereas standard aggregation allows for spurious individual signatures as long as they sum up to the correct aggregate. Furthermore,* MSP.BKey *uses a weighting seed* $e_{\boldsymbol{\sigma}}$ *as input; in practice this is produced by the signature set to be verified and cannot be run ahead of time. In our use case, this can be overcome by having* MSP.BKey *be evaluated inside a proof system.*

## 2 Relations for Mithril(latest version)

Here we post the latest version of Mithril relations(which we plan to prove in SNARK). It is somewhat different from the initial version:

- $ivk = \mathsf{MSP.BKey}(\mathbf{mvk}, \boldsymbol{e_\sigma})$ and $ivk_{\mathsf{body}} = \mathsf{MSP.AKey}(\mathbf{mvk})$
- $(\mu, e_{\boldsymbol{\sigma}}) = \mathsf{MSP.BSig}(\boldsymbol{\sigma})$
- $\forall i : \mathsf{index}_i \leq m$ and $\forall i \neq j : \mathsf{index}_i \neq \mathsf{index}_j$.
- For $i \in \{1 \ldots k\}$: $(mvk_i, \mathsf{stake}_i)$ lies in Merkle tree $\mathsf{AVK}, N$ following path $\boldsymbol{p}_i$.
- For $i \in \{1 \ldots k\}$: $\mathsf{MSP.Eval}(\mathsf{topic}, \mathsf{index}_i, \sigma_i) = ev_i$.
- For $i \in \{1 \ldots k\}$: $ev_i \leq \phi(\mathsf{stake}_i)$.

Concretely, statements are $x = (\mathsf{AVK}, ivk, ivk_{\mathsf{body}}, \mu, e_{\boldsymbol{\sigma}}, \mathsf{mesg})$ and witnesses are of the form $w = (mvk_i, \mathsf{stake}_i, p_i, ev_i, \sigma_i, \mathsf{index}_i)$ for $i \in \{1 \ldots k\}$.

Note that we implemented the $\mathsf{MSP.vers}(msghash, ivk, \mu)$ in the circuit, so the actually relations will be different from the paper. But we can adjust it for free.

## 3 MSP Benchmark

### 3.1 MSP.A Benchmark

It is basically a BLS multi-signature, we have benched before, here is the result:

**Note:** The configuration with Degree , Advice , Lookup , Fixed , Lookup Bits , Limb Bits , Num Limbs is 17, 25, 3, 1, 16, 88, and 3, respectively.

| Num Aggregation | Proof Time | Proof Size | Verify Time |
|---|---|---|---|
| 2 | 17.4515s | 11520 | 72.7713ms |
| 200 | 22.3638s | 15008 | 89.0890ms |
| 2000 | 67.0237s | 45952 | 286.869ms |
| 4000 | 116.439s | 79680 | 327.100ms |
| 6000 | 165.035s | 113760 | 551.478ms |
| 8000 | 210.119s | 147840 | 664.927ms |
| 10000 | 256.993s | 181920 | 841.507ms |

Table 1: Benchmark results for varying aggregation sizes (Version 1)

## 3.2 MSP.B Benchmark

### 3.2.1 Server Benchmark

We benchmarked the MSP.BKey and MSP.BSig, as well as pairing verification(MSP.Ver).

The following benchmark is on the server:

| | |
|---|---|
| **OS** | Ubuntu 20.04.6 LTS (x86_64) |
| **Kernel** | 5.4.0-169-generic |
| **CPU** | Intel Xeon Silver 4214 (48 cores) @ 3.200GHz |
| **GPU** | 4 x NVIDIA GeForce RTX 2080 Ti |
| **Memory** | 128546 MiB |

**Note:** The configuration with Degree , Advice , Lookup Bits , Limb Bits , Num Limbs is 19, 6, 18, 90, and 3, respectively.

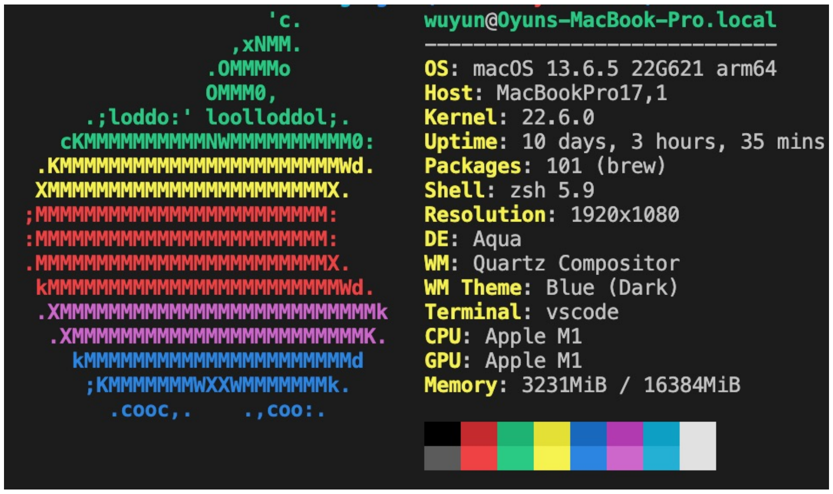
| num_aggregation | proving_time | verification_time |
|---|---|---|
| 4 | 70.5080s | 14.4204ms |
| 8 | 100.3936s | 17.5523ms |
| 16 | 145.7191s | 18.6467ms |
| 32 | 251.0654s | 21.7318ms |
| 64 | 471.5117s | 30.2121ms |

Our program can only handle signing with up to 64 public keys. When attempting to sign with 128 public keys, the server's 126 GB of RAM and 8 GB swap partition become full, leading to the program being killed. Therefore, the program cannot continue.

### 3.2.2 PC Benchmark

We benchmarked the MSP.BKey and MSP.BSig, as well as pairing verification(MSP.Ver).

The PC's configuration is shown in the following figure.



```
wuyun@Oyuns-MacBook-Pro.local
-----------------------------
OS: macOS 13.6.5 22G621 arm64
Host: MacBookPro17,1
Kernel: 22.6.0
Uptime: 10 days, 3 hours, 35 mins
Packages: 101 (brew)
Shell: zsh 5.9
Resolution: 1920x1080
DE: Aqua
WM: Quartz Compositor
WM Theme: Blue (Dark)
Terminal: vscode
CPU: Apple M1
GPU: Apple M1
Memory: 3231MiB / 16384MiB
```

The following benchmark is on the PC:

| num_advice | degree | lookup_bits | limb_bits | num_agg | proving | proof_size | verification |
|---|---|---|---|---|---|---|---|
| 25 | 17 | 16 | 88 | 4 | 73.2230s | 36192 | 14.5060ms |
| 25 | 17 | 16 | 88 | 8 | 114.6814s | 49856 | 20.6187ms |
| 6 | 19 | 18 | 90 | 4 | 75.5567s | 8864 | 5.3844ms |
| 6 | 19 | 18 | 90 | 8 | 115.3011s | 12576 | 8.0946ms |
| 2 | 21 | 20 | 88 | 4 | 118.3032s | 2848 | 4.2537ms |
| 2 | 21 | 20 | 88 | 8 | 159.2993s | 3776 | 5.5756ms |

Table 2: MSP Benchmark on PC(original)

We also benchmark the MSP in the different group, this situation is consistent with the Mithril paper:

| num_advice | degree | lookup_bits | limb_bits | num_agg | proving | proof_size | verification |
|---|---|---|---|---|---|---|---|
| 25 | 17 | 16 | 88 | 4 | 73.8717s | 36192 | 13.4883ms |
| 25 | 17 | 16 | 88 | 8 | 124.6366s | 49856 | 22.3612ms |
| 6 | 19 | 18 | 90 | 4 | 75.7733s | 8864 | 5.7507ms |
| 6 | 19 | 18 | 90 | 8 | 118.0805s | 12576 | 8.3911ms |
| 2 | 21 | 20 | 88 | 4 | 117.6121s | 2848 | 14.2941ms |
| 2 | 21 | 20 | 88 | 8 | 163.3018s | 3776 | 6.1740ms |

Table 3: MSP Benchmark on PC(swap group,where pk is on G2)

Similarly, when we attempt to run 32 signature aggregations on the PC, the 16 GB of RAM and approximately 40 GB of swap partition (which varies) become full, ultimately resulting in the program being killed.

## 4 Next Step

We plan to do more tests and benchmarks on MSP.

- Adjust the configs(degree, advice) of MSP circuit.
- Optimize the scalar_multi function in halo2-lib.
- Manege the RAM on the server to run the large scale data.