25.01.21 Compare SNARK-based Mithril and SNARK-based ATMS

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

January 21 2025

1 Mithril and ATMS

ATMS, from the *Proof-of-Stake Sidechains* paper, is a signature scheme designed for sidechains' certificate:

An important feature of our construction is merged-staking that prevents "goldfinger" attacks against a sidechain that is only carrying a small amount of stake. An important technique for pegging chains that we use in our construction is cross-chain certification which is facilitated by a novel cryptographic primitive we introduce called ad-hoc threshold multisignatures (ATMS) which may be of independent interest. We show how ATMS can be securely instantiated by regular and aggregate digital signatures as well as succinct arguments of knowledge such as STARKs and bulletproofs with varying degrees of storage efficiency

The nodes on the mainchain will get information from sidechains, and the information allow them to authenticate a small number of stakeholders on the sidechains, whom are trusted to present a majority of all stakeholders. The information changes periodly, since the stake distribution is a dynamic value.

All the transactions and authentication information will be packed per "epoch", the paper call it a *cross-chain certification.*

Mithril, from the paper *Mithril: Stake-based Threshold Multisignatures*, is a signature scheme designed for mainchain's certificate:

We also examine the problem of bootstrapping light clients in Proof of Stake (PoS) blockchains. The general challenge in this setting is that the client needs to verify the ledger upon joining the network and that block verification fundamentally depends on stake (unlike an SPV client in the bitcoin setting, that can simply count the blocks' aggregate difficulty). As a result, a client bootstrapping in the PoS setting needs to follow the stake as it moves between accounts to be in sync over time with the stakeholder distribution and validate all the blocks. The *amount of work to be performed scales linearly with the number of transactions in the ledger which can be extremely large. Using mithril, a different approach can be followed: instead of verifying transactions, the stakeholders can issue checkpoints at regular intervals using an STM signature. The client needs only to verify all checkpoints till the most recent one after which individual blocks and transactions can be verified sequentially. In this way the operation becomes linear in the number of checkpoints instead of linear in the number of transactions. The frequency of the checkpoints can be set to be at regular intervals.*

In our zero-knowledge bridge context, we refer to the second application in the Mithril paper, that is the quick bootstrapping of Mithril light client(Cardano light client). The Mithril protocol provides Blockchain checkpoints periodly, and the light client can verify the Mithril certificates continuously.

We offer a quick comparison of two schemes:

	Mithril	**ATMS**
core signature	BLS	Schnorr/BLS
aggregation	complex	simple
proving task	heavy	light
target	N/A	mainchain
participation	low percentage	high percentage
use case	bootstrapping	cross-chain

The ATMS is used for blockchain's interoperability, but as described in the paper, ATMS signature can only be verified on the mainchain. So if we want to construct a bridge using ATMS, it can only bridge from a sidechain to Cardano mainchain.

While Mithril is not designed for cross-chain, and the "light client" use case of Mithril allow it to be used for bridge. Thus if we construct a bridge using Mithril, it can bridge from Cardano to "any" other blockchain(whether in the same ecosystem or not).

2 ATMS in Schnorr and BLS

As we discussed in the former documents, when turns to BLS signature scheme, the original proving method(relations in the paper) will be very costly. So we prove the aggregation key and verify the ATMS signature.

Below is our benchmark result of ATMS with product of all keys:

num	proof_time	proof_size	verify_time
3 of 6	25.27s	11872	13.42ms
6 of 9	26.81s	12224	12.45ms
8 of 9	26.59s	12224	12.07ms
14 of 21	27.51s	12576	14.54ms
17 of 21	27.45s	12576	13.15ms
28 of 42	28.80s	13376	11.48ms
72 of 102	32.13s	15104	13.80ms
1602 of 2001	160.75s	80192	39.05ms

Table 1: Proof Time, Proof Size and Verify Time for ATMS in BLS

And we made a quick comparison of two ATMS proving method, only compare the proving time:

num	BLS ATMS	Schnorr ATMS
3 of 6	25.27s	1.26s
6 of 9	26.81s	2.33s
8 of 9	26.59s	2.33s
14 of 21	27.51s	4.33s
17 of 21	27.45s	8.03s
28 of 42	28.80s	8.15s
72 of 102	32.13s	28.76s
1602 of 2001	160.75s	$\approx$ 240s

Table 2: ATMS Proving Time Comparison

The reason why BLS ATMS's proving time do not show significant changes in aggregation number is that we did not customize circuit's degree, but it is actually a $O(n)$ algorithm.

Note that these two versions of ATMS benchmarks are running over very different settings, following is the information:

So according to the benchmark settings, the actual performance of two ATMS schemes is close.

	BLS ATMS	**Schnorr ATMS**
machine	Linux Server	PC
CPU	Intel Xeon Silver 4214 (48 cores) 3.2GHz	Intel Core i5-12500kf 3.6GHz
RAM	8 GB	128 GB
library	halo2-lib	sidechains-zk
curve	bn254	bls12-381/jubjub
hash	Poseidon	Rescue

Table 3: ATMS Benchmark Settings