bridge survey and benchmark

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

March 26, 2024

1 Bridge uses zk-proofs

There are severe cross-chain bridge based on zk-proof techniques:

1. Succinct by Succinct Labs Succinct Labs has developed a system that allows for a trust-minimised connection between Gnosis and Ethereum 2.0, a proof-of-stake consensus blockchain. The system uses SNARKS to efficiently verify the validity of consensus proofs on the Gnosis chain.

The Ethereum 2.0 network has a committee of 512 validators randomly chosen every 27 hours and is responsible for signing every block header during that period. If at least 2/3 of the validators sign a given block header, the state of the Ethereum network is considered valid.

The verification process of Ethereum mainly includes the verification of the following contents: the Merkle proof of the block header; the Merkle proof of the validators in the synchronization committee; the BLS signature of the correct rotation of the synchronization committee, etc. The core idea here is to use zk-SNARKs (Groth16) to generate constant-size validity proofs that can be efficiently verified on-chain on Gnosis.

2. zkIBC by Electron Labs Specifically, zkIBC hopes to simulate the Inter Blockchain Communication Protocol (IBC), the trustless communication protocol used by the Cosmos sovereign chain, and expand its use to Ethereum. zkIBC uses ZK-SNARKs for light client status verification, quickly proves transactions on Ethereum, and keeps up with the block time of the Tendermint consensus chain.

However, using a light client from the Cosmos SDK on Ethereum presents some challenges. The Tendermint light client used in the Cosmos SDK operates on the Ed25519 curve, which is not supported natively on the Ethereum blockchain. This makes it expensive and inefficient to verify Ed25519 signatures on Ethereum's BN254 curve. Electron Labs plans to solve this problem by creating a system based on a zkSNARK, which can generate a proof of signature validity off-chain and only verify the proof on the Ethereum chain.

The current testnet requires waiting approximately 20–30 minutes for finality, which includes Goerli network finality (15–20 minutes), ZK-Proof generation

(5–8 minutes), Near chain minting (10–20 seconds) .

3. zkBridge by BerkleyRDI The construction we want to follow. The main innovations of zkBridge are:

· deVirgo: Uses a distributed approach to generate ZK-SNARK proofs without trust assumptions. The deVirgo method greatly improves the time to generate ZK-SNARK proofs off-chain by splitting the calculation work and allocating it to more devices.

· Recursive proof: In order to reduce on-chain costs, zkBridge uses recursive proof. Through two recursions, the size of the ZK-SNARK proof is compressed to about 131 bytes. The first step is to generate deVirgo proofs, and the second step is to use the Groth16 proof generator for compression. The Groth16 verifier generates integrity proofs of executing deVirgo circuits.

· Batch processing: zkBridge implements a block header update contract, which takes the block height as input and returns the corresponding block header. However, zkBridge does not call the update contract when each new block is generated. The prover can first collect N block headers to generate a single proof. The N value can be set. The larger N, the longer the user waiting time but the lower the system operating cost.

Currently, zkBridge has implemented an instance of Cosmos Client on Ethereum using Solidity. According to tests, it can generate a ZK-SNARK proof of the Cosmos Zone block header in **2** minutes.

2 Comparison of bridge

We compare the existing zero-knowledge bridges from diverse dimensions, including architecture, proof systems and their communication complexity. See it in Table. 1.

	Succinct labs	zkIBC	zkBridge
Architecture	Application specific	Application specific	Application agnostic
Zk-proofs	Groth16	Groth16	deVirgo, Groth16
Communication Complexity	O(1)	O(1) per batch of signatures	$O(N \log_2 R)$ N: number of machines R: gates per layer

Table 1: Comparison of bridges

And we also do a survey about the proof generation cost in these bridge schemes. In order to clarify this issue, we conducted investigations from verification task, constraint size, proof generation time and other aspects. And the cost of verifying the proof is also presented. See it in Table. 2.

	Succinct labs	zkIBC	zkBridge
Verification Task	Sync Committee: 512 high stake signatures Verification of Ethereum BLS signatures on Gnosis	Verification of EdDSA Curve25519 signatures on Ethereum Validators: 32 high stake signatures	Verification of EdDSA Curve25519 signatures on Ethereum Validators: 32 high stake signatures Computation distributed across relay networks
Constraint Size	Sync Committee rotation: 68M constraints Verify signed header: 21M constraints	2.5M constraints for every Ed25519 signature	2.5M constraints for every Ed25519 signature
Proof Generation Time	~180 secs	~300 secs per batch proof of 32 signatures. With parallelism: $<$ 7 secs	~18.21 secs per proof of 32 signatures.
Gas Cost	226k gas per signature	300k gas per batch	227k gas(constant because of Groth16)

Table 2: Comparison of cost

3 Comparison of Groth16 and ATMS

We first compare the on-chain cost of Groth16 verifying with Ad-hoc Threshold Multi-Signatures(atms) over Cardano.

First, there are some bls12-381 curve operation costs, which has been estimated Kenneth that show how expensive each function is. Follwing is the table of cost, where x is the size of the input in bits divided by 64. See it in Table. 3.

Function name	cost(cpu units)
bls12_381_G1_compress	3341914
bls12_381_G1_uncompress	16511372
bls12_381_G1_add	1046420
bls12_381_G1_equal	545063
bls12_381_G1_hashToCurve	66311195 + 23097*x
bls12_381_G1_mul	94607019 + 87060*x
bls12_381_G1_neg	292890
bls12_381_G2_compress	3948421
bls12_381_G2_uncompress	33114723
bls12_381_G2_add	2359410
bls12_381_G2_equal	1102635
bls12_381_G2_hashToCurve	204557793 + 23271*x
bls12_381_G2_mul	190191402 + 85902*x
bls12_381_G1_neg	307813
bls12_381_GT_finalVerify	388656972
bls12_381_GT_millerLoop	402099373
bls12_381_GT_mul	2533975
blake2b_256	358499 + 10186*x (521475, with x = 16)
addInteger	85664 + 712*max(x,y) (88512, with x = y = 4)
multiplyInteger	1000 + 55553*(x+y) (641924, with x = y = 4, and we include the price of modular reduction, as we need one per mult)
divideInteger	if x>y then 809015 + 577*x*y else 196500
modInteger	196500
expInteger	We estimate 32 mults and adds (23373952)

Table 3: Function Cost

Then we can make a back-of-the-envelope computations to see how feasible it is to verify SNARKsn or a atms on main-net.

Here is the comparison table of the cost, where M is the total number of signers(or the committee size), and N is the number of non-signers. See it in Table. 4.

Funtion	Groth16 Verification	ATMS Verification
bls12_381_G1_uncompress	4	N
bls12_381_G2_uncompress	4	0
bls12_381_G1_mul	1	0
bls12_381_G1_add	1	$N+1$
bls12_381_GT_millerLoop	4	2
bls12_381_GT_mul	2	0
bls12_381_GT_finalVerify	1	1
blake2b_256	0	$\log_2 M * N$
Total	2,299,066,153	1,914,978,116 ($M = 100, N = 34$)

Table 4: Verification Cost

When the number of signers incrase, the cost of ATMS behave like Table. 5.

signers number	non-signers number	cert sizr	total cost
100	34	3284	1,914,978,116
200	67	6280	2,649,784,802
300	100	8988	3,419,008,838
400	134	12132	4,175,545,116
500	167	14712	4,909,830,327
600	200	18796	5,748,410,538
700	234	21460	6,522,676,966
800	267	24104	7,274,170,852

Table 5: ATMS Verification Cost

4 Proving Time

we also benchmark the proving time of EdDSA signature over curve 25519, using Halo2 as proof system.

We use the code from Ayush Shukla, and the result are follwing:

degree	num_advice	num_lookup	num_fixed	limb_bits	proof_time	proof_size	verify_time
19	1	1	1	88	11.9s	1920	67.20ms
18	2	1	1	88	8.7s	3200	63.43ms
17	4	1	1	88	5.7s	5632	44.95ms
16	8	2	1	90	5.5s	10976	47.02ms
15	17	3	1	90	6.0s	22688	40.15ms
14	34	6	1	91	7.3s	46240	57.24ms
13	68	12	1	88	9.6s	94048	76.64ms
12	139	24	2	88	14.4s	192736	126.37ms
11	291	53	4	88	35.8s	416800	189.40ms

Table 6: EdDSA signature benchmarks

And compare with the zero-knowledge bridges we mentioned before, it will be competitive just using halo2 trivially. The proof generation phase may cost about 200 secs, which is 10x slower than zkBridge.