GOAT Network: Natively Extending Bitcoin using Entangled Rollup

GOAT Network Research Group

July 2024 - Draft Version 0.5

Abstract

The essence of a Rollup is its ability to inherit the security of the underlying L1 chain; we call this *native security*. Two aspects determine the security of a Bitcoin Rollup. First, the user's asset should be controlled by Bitcoin script and can be withdrawn from Rollup chain under 1-of-n honestly assumption. Second, the Rollup chain's transaction inclusion and finalization can be verified by Bitcoin and for Bitcoin. The inclusion is the process of packing offchain transactions into a batch fairly, and the finalization verifiably settles down the batch on Bitcoin.

To achieve the above security goals, we introduce an Optimistic Challenge Protocol (GOAT-OCP) based on BitVM2, and apply this mechanism on GOAT's Rollup, including the asset peg-in and peg-out, and the verification of the offchain transaction batch. Furthermore, to realize GOAT's higher liveness and fairness, we are the first L2 that introduces Bitcoin Script based Locking Mechanism, which is a concrete instance of GOAT OCP, to decentralize the GOAT's sequencer network. In combination with ZKM Entangled Rollup's trust-minimized and universal settlement infrastructure, this results in a decentralized and secure Bitcoin Layer 2 (L2) solution.

1 Introduction

1.1 Motivation and high-level intuition

For a long period of time, Bitcoin's market cap has been more than a trillion dollars. And that's a lot of money: even a millionth of a trillion is still a million. But investors who own a large number of bitcoins have a problem, since other than hoping that its value goes up, there is currently no way to generate income from having BTC. The capital just sits without generating any yield.

The GOAT Network wants to change this.

It is well-known that huge efforts are underway to scale Ethereum, using side chains and Layer 2 rollups, leading to billions of dollars of Value locked in two L2 solutions. If we aspire to something similar for Bitcoin, the first thing we must solve is to have an L1-L2 bridge.

But a serious restriction of bitcoin is that its scripting language is very limited. For instance, in the case of a Layer 2, when bitcoins (that were earlier deposited) are withdrawn (or redeemed), usually a computationally intensive verification procedure needs to be performed. Unlike other main chains, Bitcoin does not have the means to implement a zkRollup, in which the rollup data sent to L1 comes

with a zkProof which get verified by a smart contract on L1. Because of bitcoin's restrictions on its scripting language, implementing such a zkProof verifier on L1 is not an option.

We believe that in this setting there is an important role to play for bitcoin whales; namely, as guarantors in this withdrawal procedure. Provided they deposit a locked asset, they are allowed to act as Operators for conducting the withdrawal procedure. For providing this service they are entitled to a fee; but on the other hand, they risk losing their locked asset if they behave dishonestly.

To solve this problem, this paper presents an Optimistic Challenge Protocol (GOAT-OCP) as an important building block of the GOAT Network. Our solution is based on BitVM2, combined with ZKM Entangled Rollup. The high-level idea of OCP is as follows.

When an Operator proposes a withdrawal transaction from L2 to L1 he needs to prove the correctness of this transaction. The Operator, now called Prover, transforms this transaction into a validity proof. He commits to this proof using the specific tree structure offered by Bitcoin Script (called Taproot – details in later sections), with a leaf script for every logic gate in the circuit. This tree structure is constructed to have a very specific property: if the Prover did not create this tree correctly and a Challenger finds out, the latter can trigger a transaction transferring the Prover's initial locked asset to its (the Challenger's) account. This slashing of the Prover's locked asset and transfer is immediate, in the sense that no collaboration of the Prover is needed, nor can he stop it.

To phrase it differently: the protocol is such that it forces the Prover to create his own trap (punishment) in case he is dishonest. Refusing to cooperate is not an option either, because a timer contract is running which will eventually transfer the locked asset to the Challenger. So a Prover has a strong incentive to act correctly, and we therefore assume that most of the time he does. This philosophy is similar to Optimistic Rollups, which also use a Challenger.

This Optimistic philosophy means that almost all computation and interaction can be performed offline. Only in the case of a challenge will the on-chain protocol be executed, with bitcoin L1 serving as the arbitrator.

Note that L1 is not used to perform a computation, it is used to verify that the result of some computation is correct. So there is no contradiction with Script not being Turing complete, as long as it is powerful enough to implement a zkProof verification algorithm.

1.2 Comparison to other Bitcoin scaling solutions

The following Table 1 shows a comparison to other solution.

GOAT Network introduces the GOAT-OCP to enhance the security of its consensus layer and finality layer, allowing the honestly assumption reduced from n-of-n to 1-of-n. This significally improved the security of existing Bitcoin L2s. GOAT Network is the first Bitcoin L2 that introduces the Decentralized Sequencer and secures the Consensus Layer by Bitcoin Locking/Slashing mechanism. Powered by those capabilities, GOAT Network aims to be the most decentralized Bitcoin scaling solution.

Table 1: Main Bitcoin Scaling Solutions

	Bsquare	Lightning Network	RGB/RGB++	Stacks	Chainway	GOAT
Scaling Type	Rollup	State Chanel	Client-side Validation / Sidechain node Validation	Sidechain	Rollup	Rollup
Smart Contracts	Yes	No	Yes	Yes	Yes	Yes
EVM Compatible	Yes	No	No	No	Yes	Yes
TPS	2k-4k	Millions	Millions	19	Millions	Millions
Finality	zk/challenge response on Bitcoin	HTLC	Lightning Network /Sidechain	PoX consensus	sovereign rollup node	challenge response on Bitcoin
Decentralized Sequencer	No	-	No	No	No	Yes
L1 asset	n-of-n multisig	2-of-2 multisig	2-of-2 multisig	2-of-2 multisig	n-of-n multisig	n-of-n multisig

2 Preliminary notions

In the context of this paper, the word bridge can refer to an L1-L1 asset transfer method, but also asset transfer between L1 and L2, and vice versa. An L1-to-L2 transfer is called a *deposit*, an L2-to-L1 transfer is a *withdrawal*.

What constitutes a proof of transaction? This depends on the type of transaction. For an L1-to-L1 transaction, A SC on L1 has direct access to its L1 transaction. So the txId serves as a proof-of-transaction. A similar statement is true for a transaction. For an L1-to-L2 transaction, a SC on L2 has indirect access to L1 and its transaction, and the rollup SC can communicate the information. So again the txId serves as a proof-of-transaction, but for an L2-to-L1 transaction, A SC on L1 does not have direct access to its L2 transaction, since the data is compressed and the stateRoot is a hash. So it needs a zkProof if we want native security.

Note that such zkProofs are like a certified check. And by sending them to other chains, it is possible to do cross-chain transfer.

2.1 Entangled rollup: several L1s sharing one L2

Given two blockchains A1 and B1, and a rollup, an interesting design is to run a copy of the Rollup SC in parallel on two different underlying blockchains. In other words, one L2 with a rollup to two different L1s. This idea can be extended to several underlying L1s for one rollup/L2.

The GOAT Network is a BTC Rollup integrated with such an Entangled Rollup Network, where the latter works as a unified liquidity provider and serves various Layer 1 Blockchains with one, universal settlement layer. The Entangled Rollup uses ZKM's infrastructure and reuses the validity proof of the Rollup transaction batch as the L2-to-L1 bridge transaction receipt. In this way, the destination chain can easily verify the receipt and complete the bridge transaction.

By using this Entangled Rollup, we can support the direct depositing of base blockchain assets without the need to introduce separate trusted entities that function as a bridge. In this approach, the deposited assets on the other end can be seen as native assets (secured on the same consensus of the original blockchain), enabling native liquidity across all connected blockchains.

2.2 Winternitz signatures

We use Winternitz signatures as our one-time signature scheme because they offer a significant reduction in the size of both the signature and the public key compared to the Lamport One-Time Signature. This improvement is accompanied by a higher computational complexity for generating and verifying signatures.

3 Natively Extending Bitcoin

3.1 Native Security

GOAT Network introduces an Optimistic Challenge Protocol (GOAT-OCP) mechanism as its scaling type, which uses the native Bitcoin Script to implement both its Rollup validity proof's verification and Decentralized Sequencer's locking/slashing scheme.

GOAT-OCP is built on the idea of BitVM2, which is a permessionless bridge protocol, using a n-of-n Multisig pre-signed Assert transaction to commit the intermediate values in the input and computing trace as Taproot in the output. It allows anyone can serve as a verifier without being part of the initial group of n operators.

An optimization of GOAT-OCP from the BitVM2 is, we come up with an abstraction of the computing trace to allow any computation to be converted to the computing trace via ZKM's Continuation techniques.

GOAT-OCP can provide native Bitcoin-level security to any off-chain computation, like securing the L2's transaction inclusion in the sequencer network. The overall GOAT-OCP architecture is depicted via an example in Figure 1.

3.2 Implementing General Computation via Bitcoin Script

BitVM is a NAND-circuits-based solution that converts logic and binary operations into a NAND circuit, and commits the computing traces via logic gate commitments and binary gate commitments, and allows Optimistic Challenge via Bisection protocol. But the Bisection is slow on Bitcoin, and BitVM2 is introduced via MAST and Taproot to commit the computation traces.

Theoretically, the SNARK proof verifier can be expressed by NAND circuits regardless of the Script size. For instance, the STARK verifier doesn't need a trusted setup, but it is heavily dependent on hashing and on-field operations, like plonky2 and eStark. Another pain point of STARK is its large proof size. The SNARK verifier, take Groth16 for instance, on the other hand, which is based on elliptic curve pairings, needs a trusted setup but achieves a very small proof, which is about 200 Bytes ($2G_1 + 1G_2$ or 4 elements).

BitVM's analysis shows that a full block can commit to about 4 MB / block * 26 bytes/bit = 16 kB/block of signed data, where 4MB corresponds to the maximum script size, and 26 bytes to a Winternitz signature for one bit.

It follows from Zulu's work that the script size of Groth16/fflonk reaches up to 7GB, and it needs to be split into fewer than 2,000 sub-scripts. This scenario is likely impractical to kick-off a realistic challenge-response game on Bitcoin. But latest development shows that the number of the sub-scripts can be reduced into about 1000, and this allows one Taproot to commit a Groth16 verifier.

Another slower attempt is trying to have the FRI-based STARK executed by Bitcoin Script, due to the lack of OP_CAT, the concatenation operation, in Bitcoin Script.

So we adopt the following approach. GOAT Network uses ZKM to generate the validity proof, and ZKM produces a SNARK proof for on-chain verification. Hence, GOAT Network is implementing the SNARK verifier scripts on top of BitVM verifier gadgets as its settlement layer with more proof compression optimizations.

As we mentioned, GOAT-OCP implements an abstraction MAST generator via ZKM Continuation techniques. The generator also introduces some high-level operators, like extension field operation.

Another basic technique is precompiling. From the previous work of Geometry Research and Zeta Function Technologies, this method significantly improves the pairing verification by replacing the final exponentiation step of pairing verification with a more efficient exponentiation and introducing Pre-computing Lines and Predefined Q, which allows the precomputation for G_2 points for predetermined pairing. These ideas make it more practical to compress the SNARK Proof and make it possible to compute on Bitcoin Script.

Furthermore, for the public input, we can commit the 8 elements (32*8 bytes) of public input using a Winternitz signature.

3.3 Protocol description

We introduce an Optimistic Challenge Protocol (GOAT-OCP), which uses Script to implement both its Rollup validity proof's verification as well as its Decentralized Sequencer's Locking/Slashing scheme. OCP is a variant of BitVM2 in which anyone can serve as a Challenger. After initialization, anyone can challenge an invalid assertion, without being part of the initial group of n Operators. GOAT OCP provides native (Bitcoin-level) security to any off-chain computation, such as securing the L2's transaction inclusion in the sequencer network. Its overall architecture is depicted in Figure 1.

In the initial phase, the Operator must deposit a bond as permission to operate. GOAT-OCP then proceeds as follows:

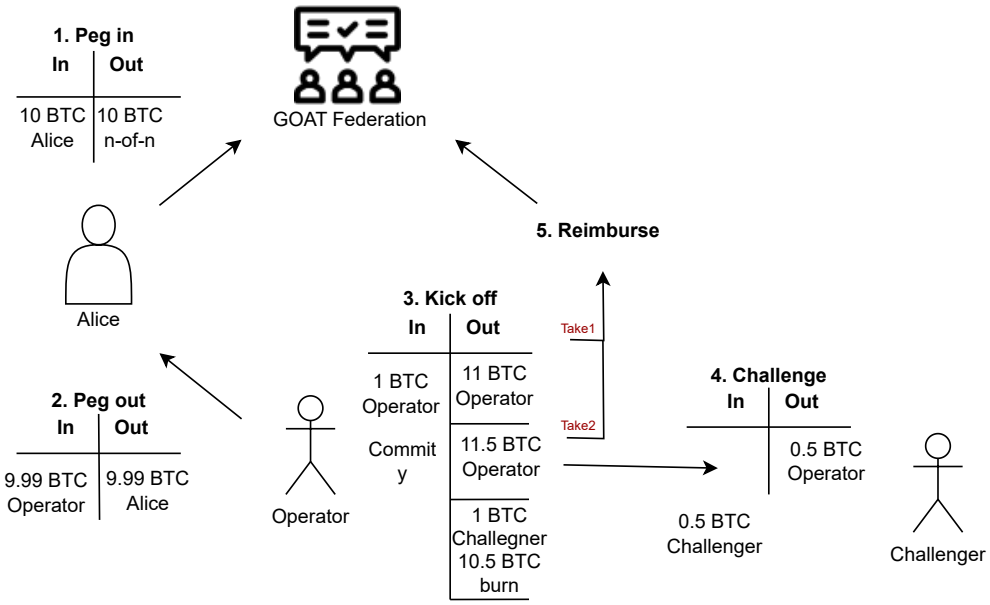

Figure 1: GOAT-OCP Mechanism

1. **Peg-In:** Alice deposits 10 BTC into a pool managed by the GOAT Federation (more on GOAT Federation later). When the asset is pegged in (known as the asset being deposited to L2), the user's asset is locked into a pre-signed n-of-n MultiSig script, signed by members of the GOAT Federation.

2. **Peg-Out:** Alice initiates a withdrawal of 10 BTC, and the Operator immediately transfers (advances) 9.99 BTC to Alice.

3. **Kick-Off:** Using the proof-of-advance, the Operator requests the transfer of 11 BTC from the pool to his account. In this step, the signer pre-signs a transaction including a challenge script and an asset script.

4. **Challenge:** During the challenge phase, any user can invoke the challenge script by paying an input UTXO of 0.5 BTC as a deposit and ask the Operator to open their commitment by providing the intermediate computation trace to the asset script. There are two possibilities:

   (a) If the Operator behaved dishonestly and the challenge was successful, the reimbursement process fails, the Challenger receives 1 BTC and 10.5 BTC is burned.

   (b) If the Operator acted honestly and the challenge failed, the challenger loses its deposit to the Operator. Two weeks later, the Operator received 11.5 BTC.

5. **Reimburse:** If no dishonest behavior was detected, reimbursement occurs after the challenge period expires (typically 2 weeks). The Multisig's redemption script should be activated after the challenge period has expired. Then the signer sends the target user's asset in advance and initiates the challenge phase.

The Operator's execution traces and trace transitions are both committed via MAST and Taproot. Each leaf of the tree represents an intermediate value and a step function, and the intermediate value is calculated by the step function with previous function output as current input, meanwhile a one-time signature of the output is generated. In this way, a Bitcoin script can perform this calculation directly, and once a prover commit two different value for some trace in the commit phase, any verifier can submit a fraud proof.

3.4 Unified Native Liquidity

The GOAT Network is a BTC Rollup built on the Entangled Rollup Network. The Entangled Rollup Network is a unified liquidity provider and serves various Layer 1 Blockchains with a universal settlement layer.

The Entangled Rollup uses ZKM's infrastructure and reuses the validity proof of the Rollup transaction batch as the bridge transaction receipt. In this way, the destination chain can easily verify the receipt and complete the bridge transaction. By using this Entangled Rollup, we can support the direct pegging of base blockchain assets without additional consensus gadgets (bridges). In this approach, the pegged assets on the other end can be seen as native assets (secured on the same consensus of the original blockchain), enabling native liquidity across all connected blockchains. This unified native liquidity can imbue GOAT Network with extremely fast warm-up times and strong community consensus. The unified native liquidity brings stability and higher yield to participants and improves the GOAT Network's robustness even further.

4 Modularity and Decentralization Design

GOAT Network is a Bitcoin L2 built on the Entangled Rollup principle of modularity and decentralization from its inception. This rationale allows us to support the development of a universal infrastructure, reducing friction for users in the Blockchain ecosystem. As shown in Figure 2, GOAT Network consists of the following modules:

GOAT Federation: A Decentralized Autonomous Organization (DAO) who consists of operators. Each Operator should be able to act as an Optimistic Challenge Protocol (OCP) Prover. An OCP Prover can manage the asset Peg-out for the end-user and kick off the asset transaction for any off-chain computation.

GOAT Nodes: Nodes are responsible for L2 transaction execution. They consist of the BridgeService and Execution Layer. The BridgeService fetches the Bitcoin block headers and updates the Simplified Payment Verification (SPV) Merkle Tree within GOAT Network. The Execution Layer, acting as the Execution Layer, select the best transaction batch, execute the batch, and confirm the transaction on GOAT Network.

Decentralized Sequencer: The decentralized sequencer includes GOAT Network transactions into batches with a decentralization mechanism. The protocol details are provided in subsection 4.2.

Proof Network: A proof factory that accepts the user's program and program input, and generates the corresponding validity proof. The protocol details are provided in subsection 4.3.

Relayer: The relayer is responsible for cross-chain communication between L1 and L2, but differs from the Rollup bridge in our case. The Rollup Bridge transmits the message from L1 to the counterpart L2, while the relayer transmits the message from L1 to its entangled L2. For instance, say there are two Rollups in an Entangled Rollup, Bitcoin L2 and TON L2. The Rollup bridge handles messaging between Bitcoin and Bitcoin L2 or TON and TON L2, while the relayer manages message passing between Bitcoin and TON L2, or TON and Bitcoin L2. The GOAT Network adopts the Inter-Blockchain Communication (IBC) protocol to implement a decentralized and secure relayer. More details are provided in the Entangled Rollup light paper.

Bridge Contract and Shadow Contract: Both the Bridge Contract and the Shadow Contract can verify the Zero Knowledge Proof from the Rollup chain. The Bridge Contract uses a Merkle Tree to keep track of the end user's bridging asset. The Shadow Contract focuses on verifying the proof from a cross-chain. More details can be found in the Entangled Rollup light paper.

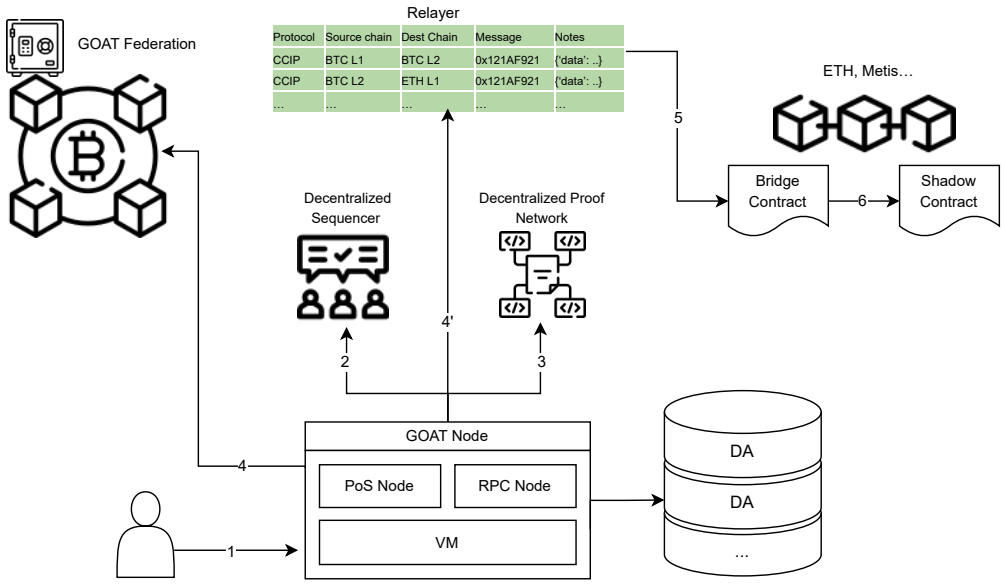

Figure 2: Entangled Rollup for BTC and ETH

Consider Figure 2 as an example to outline a high-level Entangled Rollup transaction between BTC L2 and Ethereum L2. The user submits a cross-chain transaction in step 1, moving its asset from source L2 to destination L2. The transaction first is included into a batch by the decentralized sequencer network in step 2, and the batch being proved by the proof network in step 3. Then the validity proof (including both off-chain execution and storage) is submitted to source L1 in step 4 and relayed to destination L1 in step 4'. Then the bridge contract on source L1 verifies the validity proof and unlocks the token to shadow contract in Step 5 and 6. After that, the destination L1 shadow contract can also verify the validity proof, and transfers the equivalent amount of tokens to the bridge contract. This will trigger a deposit on destination L2. Finally, the user can get the bridged tokens on destination L2.

This mechanism is built on the fact that the validity proof produced by the same infrastructure (ZKM) can be verified on both source L1 and destination L1. Compared to a traditional bridge protocol, cross-L2s bridges inherit the security of their L1s, not the bridge network itself. Since this paper focuses on

GOAT Network, more details about Entangled Rollups can be found in the previously mentioned white paper.

4.1 Trust-minimized Bridge Protocol

A trust-minimized bridge protocol between Bitcoin and GOAT Network should be able to bridge an asset without any additional security assumption. For the asset peg-in, we run the Simple Payment Verification on a GOAT Node; for the assert peg-out, it's more complicated. We split the latter procedure into three phases, 1) Inclusion phase: the user starts a peg-out transaction, and the transaction gets confirmed on GOAT, 2) Commit phase: the transaction is sealed on L2 with ZKP as the proof of correct execution, the sequencer then submits the proof to Bitcoin via Batch-submitter; 3) Challenging phase: The monitor gets notified that a new commitment has been submitted to Bitcoin, and it dispatches this message to Challenger.

From a modular perspective, we can see how GOAT Network works in Figure 3.

Asset Deposit. When we deposit the asset into L2, we use SPV proof to represent the fact that L1 has confirmed the transaction, and then verify the SPV proof on ZKM by smart contract. The L2 smart contract maintains the SPV Merkle Tree.

1. The user starts a transaction on Bitcoin with some amount of its UTXO as input, and pay to a n-of-n P2MS address. Meanwhile, the operator generates two pre-signed transactions Take1 and Take2. The Take1 can be satisfied and confirmed after two weeks if there is no challenge happened. Otherwise, if a challenge is submitted but failed to disprove, the Take2 is happened.
2. The ZKM Relayer sends the latest Bitcoin block header to the ZKM Node, and the ZKM Node maintains the SPV Merkle Tree root in GOAT Network's smart contract.
3. The user submits a claim transaction with SPV proof to the ZKM node. If the SPV proof is valid, the smart contract will mint the wrapped token on GOAT Network for the user.

Observe that the final mint operation is handled by EVM on L2's PoS Layer once the SPV is verified successfully.

Asset Withdrawal. The asset peg-out is the withdrawal process that allows the user to redeem the asset from GOAT Network back to Bitcoin. This is not only the inversion process of deposit with an additional ZKP proof; it also includes the validity proof verification.

1. When the user sends a withdrawal transaction to an RPC Server, the RPC Server adds the transaction into its TxPool. This transaction triggers the burn operations of the wrapped token, and generates a Merkle proof as a proof-of-burn on GOAT Network;
2. The RPC Server forwards the transaction to BridgeService, in which all transactions both from RPC Server and on-chain monitor are aggregated on Execution Layer. The Execution Layer chooses the best transaction batch, executes the batch, and updates the state database; Meanwhile, we also generated the pre-state and post-state of the current batch, which will be used to generate the ZKP later;

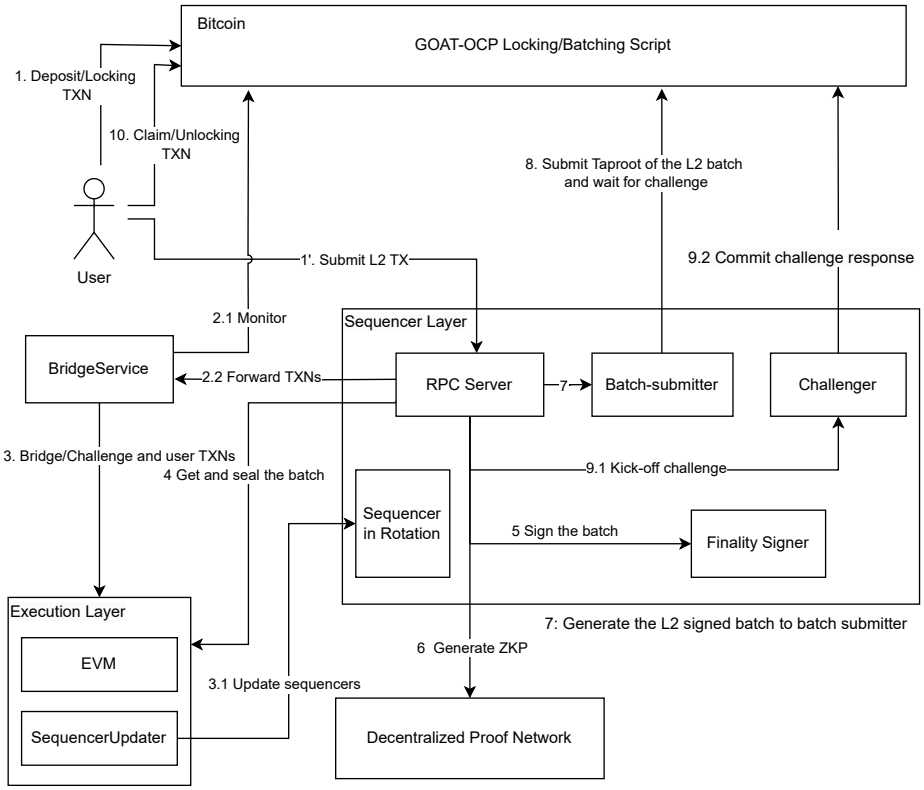

Figure 3: GOAT Network Architecture

3. The sequencer fetches the above batch as a proposal, and starts the consensus phase inside the decentralized sequencer network;

4. Once the batch is settled on GOAT Network, the Nodes publish the data into DA and submit the proof to Proof Network to generate the batch's validity proof;

5. The Nodes submit the ZKP and DA credential to Bitcoin. GOAT-OCP is used to verify the correctness of the ZKP, and the user can claim its assets on Bitcoin from one of the Sequencer Node, which is acting as the operator of OCP. It then kicks off the process of reimbursement. The Take1 and Take2 happened finally, the Sequencer Node claims its asset successfully, otherwise, the verifier wins the reward and the operator loses its BTC. More details are presented at 3.3.

Data Availability. DA services are crucial in ensuring the accessibility, integrity, and security of critical data within our system. By leveraging external DA solutions, we enhance the reliability of our operations by ensuring that data remains accessible and recoverable even in the event of system failures or disruptions.

These services typically involve storing data redundantly across multiple locations or nodes, implementing robust backup and recovery mechanisms, and employing encryption and access control measures to safeguard data against unauthorized access or tampering. Furthermore, DA services contribute to

compliance with regulatory requirements and industry standards regarding data protection and continuity, thereby instilling confidence in stakeholders and users regarding the reliability and resilience of our systems.

4.2 Decentralized Sequencer

4.2.1 Problem Space

When performing transactions on a Bitcoin L2, a sequencer is responsible for verifying, and ordering these transactions into a package that can be shipped to Bitcoin. In return for its efforts, the sequencer is paid a small portion of the fees collected from users. However, leading rollup operators use a sole-sequencer model to package transactions and update Bitcoin, which can introduce significant centralization and single-point-of-failure risks. A single sequencer has control over transaction ordering, potentially excluding user transactions and extracting Maximum Extractable Value (MEV) from transaction pools. Liveness issues may occur if the sole sequencer experiences downtime, affecting the entire rollup. While users can bypass sequencers and submit transactions directly to the L1, this negates the purpose of using an L2 rollup for cost-saving and efficient transactions.

A Sequencer Node may misbehave during the consensus process in a way that can severely harm the security of GOAT Network with Long-Range Attack, including:

- Proposing and signing two different batches for the same slot (Proposer violation)
- Attesting to a batch that "surrounds" another one (effectively changing history)
- "Double voting" by attesting to two candidates for the same block (Attestation violation)

To solve these issues, we design a Proof of Stake Byzantine Fault Tolerant (PoS BFT) with GOAT-OCP.

4.2.2 Solution Overview

As shown in Figure 3, GOAT Network consists of the Execution Layer and Consensus Layer. The Sequencer Layer consists of the Finality Signer and Sequencer Node (Sequencer in Rotation). The Finality Signer engages in signing the batches and acts as the operator in GOAT-OCP, and the Sequencer Node plays the role of batch produder.

With the enhancement of GOAT-OCP, when an operator performs any misbehavior above, any verifier can submit evidence (fraud proof) to prove this misbehavior. As a consequence, the operator would lose its locking asset, and the verifier wins the reward.

Any user can join the GOAT Network as a Sequencer Node via Sequencer Network Initialization, which is depicted by steps 1-3 in Figure 3 about how to add a new user as a Sequencer Node.

The main responsibity of the Sequencer Node is to execute the Rollup Batches Determination. This procedure is depicted by the left steps in Figure 3, which describes how the Sequencer Node creates a new batch, the Finality Signer signs the batch, and the Finality Signer/Sequence Node acts as the operator/verifier in OCP Locking scheme to play the challenge/response game during GOAT-OCP to finalize the batch on Bitcoin.

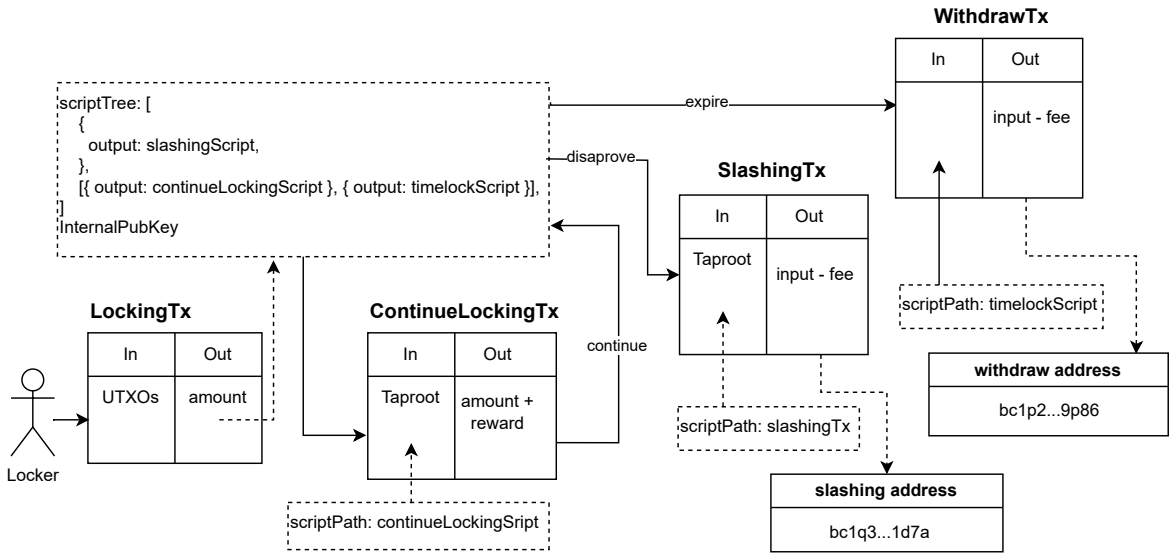

Figure 4: OCP Locking Data Flow

4.2.3 Sequencer Network Initialization

Locking. A sequencer candidate starts from making a Locking transaction that locks its asset into a Taproot, with three outputs: SlashScript, TimelockScript, and ContinueLockingScript, as shown in Figure 4.

TimelockScript. The timelockScript is a simple timelock restricting the specified future time of unlocking the locker's asset. The script checks the timelock and the locker's signature. Only when both checks are satisfied can the asset be claimed successfully by the locker.

ContinueLockingScript. If the locker has realized its responsibility of producing blocks in its epoch, and wins the challenge/response game and reaches the time point continueLockingTimelock, the locker should be able to continue locking the BTC before the timelock is released. The lockers and GOAT Federation initialize the ContinueLocking transaction by PSBT transaction.

SlashingScript. If any Finality Signer attempts to sign different batches in same height, it's secret key will be exposed via one-time signature scheme. Any verifier can extract the signer's secret key and submit a fraud proof, and the slashingScript path will be exected to slash the signer's BTC. Meanwhile the node would be kicked out of the Sequencer Network due to it's locked asset is decreased to be less than the minimal power requirement as a valid Sequencer Node.

When the BridgeService monitors a new Locking transaction confirmed on Bitcoin, BridgeService forwards this message to the SequencerUpdater. The SequencerUpdater broadcasts these messages to the Sequencer Layer, and updates the Sequencer in rotation if necessary.

4.2.4 Rollup Batches Determination

1. Users submit transactions to the RPC Server of any sequencer. The RPC Server forwards transactions to BridgeService and Execution Layer.
2. The Execution Layer produces a batch with its best strategies and notifies the RPC Server to seal the batch.
3. The RPC Server fetches the batch from Execution Layer, and requests a proposal voting signature and a one-time signature for the batch from the Finality Signer. Ideally, the Finality Signer can access the entire L2 state, and verify the state transition before signing the batch.
4. The batch submitter of the Sequencer Node submits the state root of the transaction batch with the one-time signature to Bitcoin if a withdrawal transaction submitted on GOAT Network. As a optimization, if there are two withdrawal transactions submitted within an epoch, the proof recursion will be applied.

Powered by the decentralized sequencer, we can achieve trust-minimized censorship resistance, and improve liveness.

- **Trust-minimized.** Centralized sequencers may manipulate transaction order for profit. In contrast, decentralized sequencers cannot manipulate transaction order; distributed nodes can only agree on the transaction order through the consensus mechanism.
- **Censorship Resistance.** Decentralized sequencers sort transactions through a consensus mechanism, with nodes distributed in different locations. No one institution can censor the nodes or hinder the stable operation of the consensus mechanism.
- **Improve liveness.** Centralized sequencers are prone to single points of failure, whereas decentralized sequencers do not have single points of failure. Additionally, any malicious behavior will be defended against by the consensus mechanism.

4.2.5 Differences from Babylon

Both GOAT's Decentralized Sequencer model and Babylon use the one-time signature scheme as the finality gadget tool to improve the consensus layer's liveness and censorship resistance.

Nevertheless, there are two main differences between the two networks' approach.

First, the GOAT decentralized sequencer, as the decentralized consensus layer of GOAT Network, is fully aware of the transaction details, and can prevent malicious MEV attacks and achieve realistic fairness for end users on GOAT Network.

Second, we have different locking mechanisms. Babylon allows lockers to quit locking liquidity after the unbinding timelock expires. GOAT Network encourages miners to continue locking, and can commit substantial returns via universal native liquidity. Thus, we can significantly increase the stability of GOAT Network and maintain high liveness and censorship resistance.

Nevertheless, GOAT Network could in the future partner with Babylon, natively allowing Babylon to re-lock its BTC to GOAT, act as a sequencer node, earn fees while doing so, and further improve its decentralization and liveness.

4.3 Decentralized Proof Network

GOAT Network is designed for "Type 1" zkEVM[1] and features a prover set that generates validity proofs for the EVM. The proof network is a distributed network that generates proofs for segments by ZKM, and improves GOAT Network liveness and censorship-resistance.

With the Continuations[2] supported by ZKM, we introduce a Stake-based prover selection model. Like the Bitcoin-hardened PoS we introduced in the decentralized sequencer, the provers must lock assets to participate in the network. At each proving slot, a prover is selected at random, weighed by their value of locked tokens, and computes the output. The prover is rewarded for producing a proof when chosen.

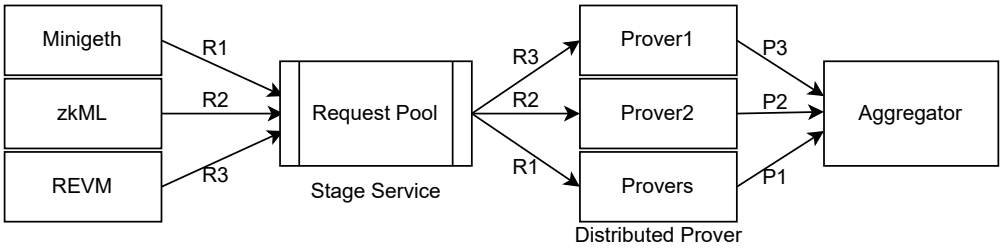

Figure 5: Decentralized Proof Network

As shown in Figure 5, a proving task is split into multiple subtasks. Different programs, like Minigeth, zkML or REVM, are compiled into MIPS ELF, and submitted to the Request Pool of Stage Service. The Stage Service splits the ELF into segments and notifies the Provers to fetch the task.

The Prover consists of the Root Prover, Aggregation Prover, and SNARK Prover. The Root Prover generates the proof for MIPS VM on zkMIPS; the Aggregation Prover aggregates two or more consecutive Root proofs; and the SNARK Proof generates the SNARK proof for the STARK verifier of Aggregation Prover. Finally, all the SNARK Proofs will be aggregated by the GOAT Node and posted to the settlement layer.

5 Analysis

We introduce a practical challenge protocol GOAT-OCP and leverage it to build the decentralized infrastructure. This allows GOAT Network to inherit Bitcoin-level security, and become a decentralized Bitcoin L2.

[1]The definition can be found at https://vitalik.eth.limo/general/2022/08/04/zkevm.html

[2]In the zkVM execution, the Continuation is a mechanism for splitting a large program into several small segments that can be computed and proven independently

Generally speaking, security can be roughly defined as the cost to manipulate history. The security of Proof-of-Stake Blockchains can be measured as the penalty that can be applied if a history reorganization is done.

5.1 Security Assumption

The security of the overall GOAT Network consists of its trust-minimized, permissionless, decentralized architecture.

Trust-minimized. Trust-minimized is defined further by state liveness and state safety. State is live if it can be consumed by its owner in finite time (in other words, it cannot be burned/locked forever), and safe if it cannot be consumed by its non-owners ever (in other words, it cannot be stolen). It is critical to evaluate the trust-minimization of a system across both these dimensions and consider worst-case scenarios.

Permissionless. A system is permissionless if it requires no in-system action for a new user to begin participating. It is permissioned otherwise.

Decentralized. While commonly (and incorrectly) described as node scale, decentralization is measured as the cost to run a fully validating node. Actually running a full node at all times is not required, but we must consider the worst-case scenario of having to run a full node. While permissionlessness is a necessary property for decentralization, it does not otherwise provide a metric for measuring decentralization.

Under the above goal of GOAT Network, the assumption is outlined as follows.

- There exists at least one honest sequencer locker, which requires a one-time setup with a 1-of-n honesty assumption, and anyone can challenge an invalid assertion without having to be part of the initial group of n during the runtime.
- The honest challenger should be the final winner for the successful slashing, and any honest user can withdraw its wrapped asset.
- The slashing for living violations can work in terms of economic benefits. This means the locker shall lose its locking asset when it misbehaves.

5.2 Security Analysis

For GOAT Network users, the abstraction of peg-in/out asset liquidity on the network is depicted in Figure 6. The user's asset is locked in an n-of-n Multisig address and unlocked by OCP Locking, the state transition of GOAT Network is verified by validity proof, and the proof is finalized by the OCP Batching. Both the OCP instances are under the 1-of-n honestly assumption.

Trust-minimized. The one-time signature guarantees the liveness of the sequencer network by providing a secure and efficient mechanism for validating transactions and updates by offering a robust solution by allowing each block or message to be signed uniquely, thus preventing replay attacks and ensuring that each operation is fresh and legitimate.

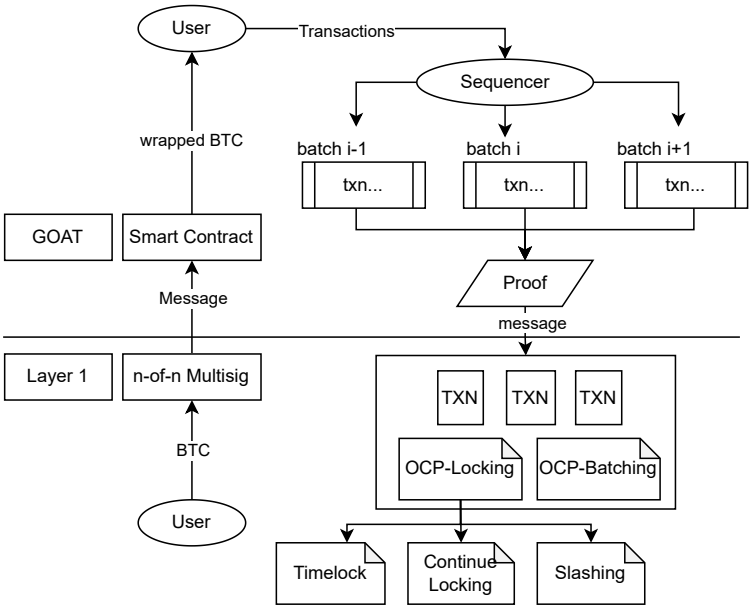

Figure 6: Asset liquidity on GOAT Network

Permissionless. All Bitcoin holders can participate by locking their BTC to become a sequencer node, or by delegating their BTC to an existing sequencer node. This inclusivity means that any BTC holder, regardless of their resources or technical expertise, can contribute to and benefit from the network's operations. By enabling BTC holders to participate directly or through delegation, GOAT Network fosters a broad and diverse set of participants, enhancing the network's resilience and decentralization. The decentralized nature of the sequencer network ensures that no single entity has control over the entire network, thereby maintaining its permissionless characteristic.

Decentralized. Decentralization is a cornerstone of the GOAT Network, underscored by its permissionless sequencer network and robust data availability solutions. These features collectively ensure that any participant can operate as a fully-fledged GOAT node, contributing to and benefiting from the network's distributed nature. The permissionless assurance means that there are no gatekeepers; anyone holding BTC can lock and become a sequencer node, or delegate their lock, fostering a diverse and resilient ecosystem.

6 Conclusion

The integration of GOAT-OCP, Decentralized Sequencer, and ZKM Entangled Rollup, collectively addressing Bitcoin's scalability challenges. GOAT-OCP improves the Bitcoin scripting capabilities, Decentralized Sequencer guarantees fair transaction ordering, Entangled Rollup ensures complex L2 smart contract execution and connect multiple L2 to achieve substantial yield. Together, these enable Bitcoin to scale natively, supporting high transaction throughput and more complex use cases without compromising security or decentralization.

Reference

1. ZKM, Entangled Rollups: Multi-chain Interoperability Without Bridges, 2024
2. Novakovic A, Eagen L. On Proving Pairings[J]. Cryptology ePrint Archive, 2024.
3. Dziembowski, Stefan, Sebastian Faust, and Kristina Hostáková. General state channel networks. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018.
4. Singh, Amritraj, et al. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications 149 (2020): 102471.
5. Gramoli, Vincent, et al. Rollup: Non-disruptive rolling upgrade with fast consensus-based dynamic reconfigurations. IEEE Transactions on Parallel and Distributed Systems 27.9 (2015): 2711-2724.
6. Buchmann, Johannes, et al. On the security of the Winternitz one-time signature scheme. International Journal of Applied Cryptography 3.1 (2013): 84-96.
7. Babylon, Bitcoin Staking: Unlocking 21M Bitcoins to Secure the Proof-of-Stake Economy, 2023
8. Nick, Jonas, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. Annual International Cryptology Conference. Cham: Springer International Publishing, 2021.
9. Gennaro, Rosario, and Steven Goldfeder. Fast multiparty threshold ECDSA with fast trustless setup. Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018.
10. Lindell, Yehuda. Secure multiparty computation. Communications of the ACM 64.1 (2020): 86-96.
11. Masip-Ardevol H, Guzmán-Albiol M, Baylina-Melé J, et al. eSTARK: Extending STARKs with Arguments[J]. Cryptology ePrint Archive, 2023.
12. Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments[J]. 2016.
13. Groth J. On the size of pairing-based non-interactive arguments[C]//Advances in Cryptology EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques
14. Gabizon, Ariel, and Zachary J. Williamson. fflonK: a Fast-Fourier inspired verifier efficient version of PlonK. Cryptology ePrint Archive (2021).
15. Ben-Sasson, Eli, et al. Fast reed-solomon interactive oracle proofs of proximity. 45th international colloquium on automata, languages, and programming (icalp 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.