SNARKTOR

A Decentralized Protocol for Scaling SNARKs Verification in Blockchains

INTRODUCTION

In recent years, the utilization of zkSNARKS has witnessed a significant upsurge in the blockchain space unlocking new possibilities for protecting users data, enhancing scalability, and allowing interoperability.

zkSNARKs play a crucial role in enabling use cases where user data privacy is fundamental while engaging in decentralized applications.

Taking a step backk, zkSNARKs are cryptographic proofs that enable one party (the prover) to convince another party (the verifier) that a certain statement is true without revealing any data involved in the statement itself. The 'zero-knowledge' property ensures that the proof does not disclose any additional information beyond the fact that the statement is true.

In the context of blockchain applications, zk-SNARKs can be adopted to enhance user data protection by allowing transactions to be verified without disclosing all the information needed for the on-chain execution. More specifically the users no longer need to share their data because they can cryptographically prove statements about their data while keeping it private and just share the proof. For instance, this enables an on-chain smart contract to verify statements on users' data by validating the Zero Knowledge proof, eliminating the necessity to access the private data of users. The smart contract integrating such capabilities enables many use cases that would have been impossible to implement on public blockchains for privacy reasons.

Looking ahead, as the need for compliance with national regulations becomes more important, digital identity will assume a pivotal role in every transaction. Zero-knowledge proofs will play a crucial role in striking a balance between privacy and regulatory adherence. This aligns with the evolving landscape of blockchain technology, where privacy-preserving mechanisms like ZKPs are pivotal for ensuring secure transactions while upholding regulatory requirements.

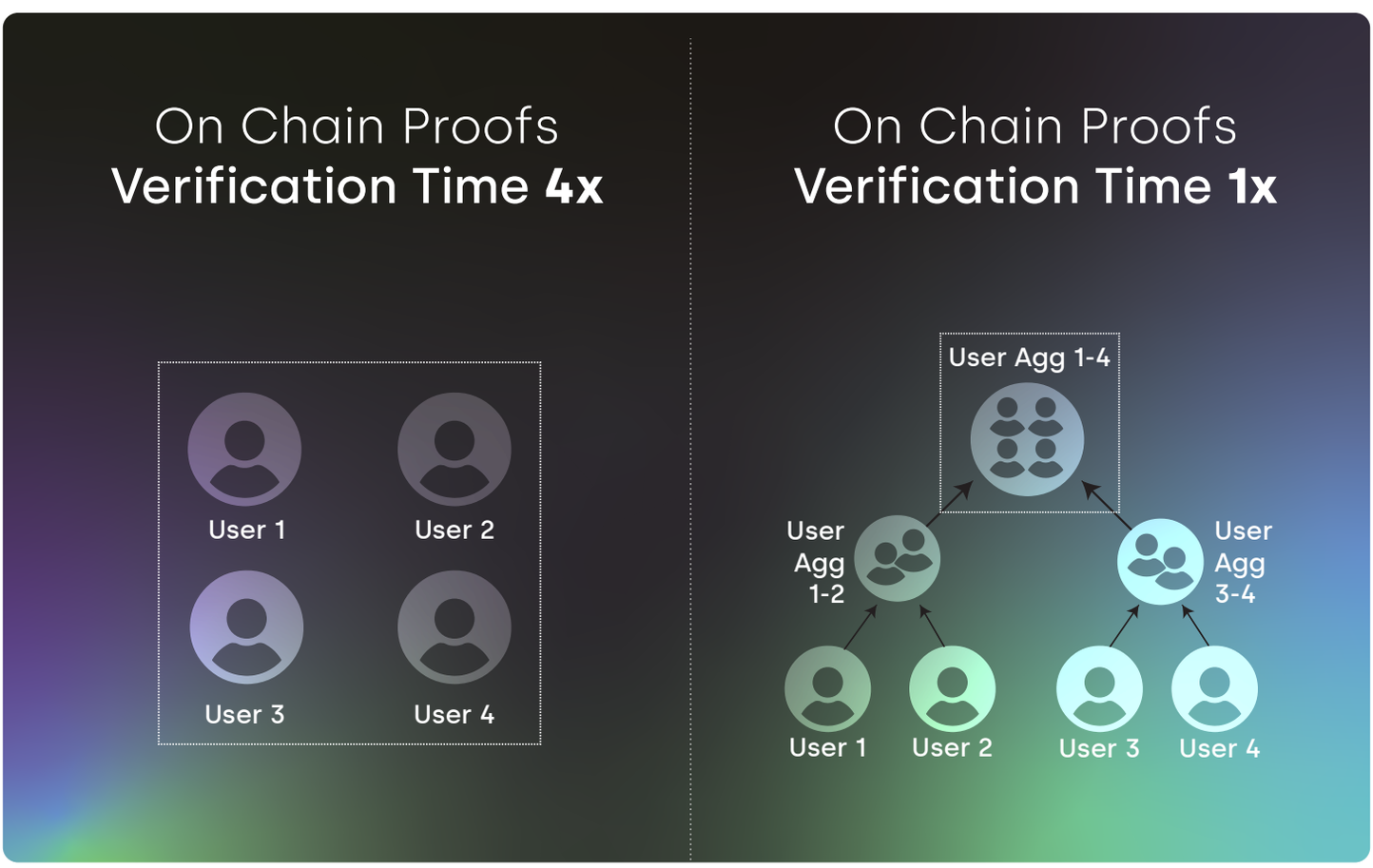
Another important application of zkSNARKs is related to scalability. As partially mentioned above, in a more general sense, the SNARK proof can be seen as a succinct proof of computation. For example, if the computation involves the execution of a VM (virtual machine), it is possible to generate a succinct proof of the VM's execution, including an Ethereum Virtual Machine (EVM). Zero-Knowledge Virtual Machines follow this approach, for instance, to enable the creation of trustless rollups leveraging zkSNARKs to prove statements to the Layer 1 chain about the rollup. This enables fully trustless, secure, and cryptographically proven communication channels.

Furthermore, the fusion of zero-knowledge principles with Artificial Intelligence (AI) holds immense potential. It allows AI models to be employed without the need for on-chain execution. Instead, only the verification of the executed result occurs on-chain, reducing computational overhead while maintaining the integrity of the AI processes.

In such a scenario, it becomes necessary to design a system that can support an increasing number of transactions using ZK proofs to be verified on-chain. For example, multiple concurrent users could submit their transactions, interacting with smart contracts featuring privacy-preserving functions, and multiple rollups or ZK bridges could post their communication proofs on the main chain. In this context, it's immediately evident that, even with proving systems with the fastest verification time, the straightforward approach of including all transactions along with their proofs in the block, and having each node verify all the proofs, quickly reaches the limits of block time and/or space budget.

MODEL

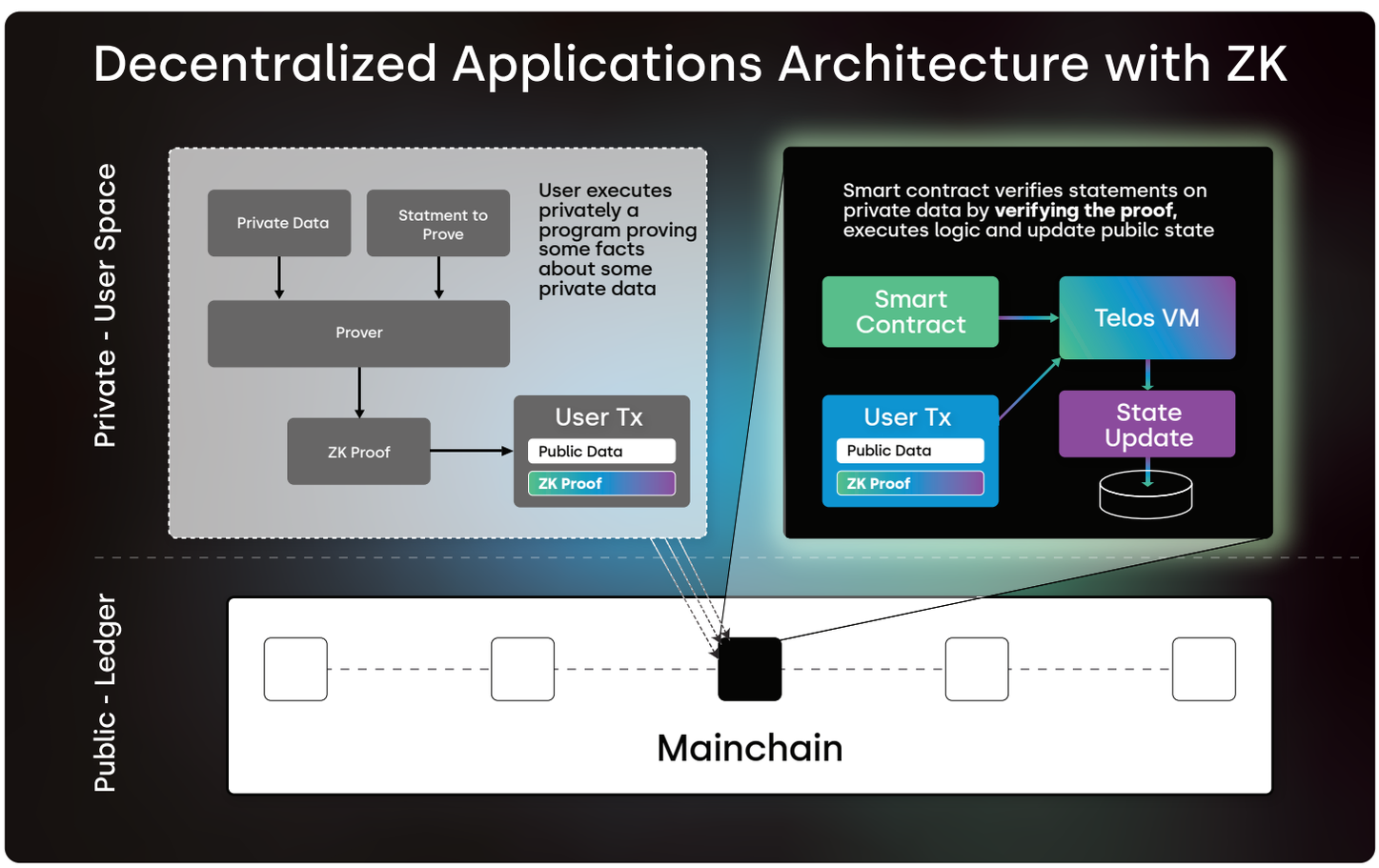
An approach that can address these challenges consists of constructing a system that leverages the advantages of efficient recursive proof composition. Recursive proof composition involves creating a new proof that encapsulates and verifies multiple existing proofs, reducing the verification cost and improving the scalability of a blockchain system supporting ZK-enabled transactions.



Building upon this concept, we present SNARKtor, a scalable and robust protocol for decentralized recursive proof aggregation. It allows aggregating many proofs for different transactions into a unique proof. The transactions can be totally unrelated (e.g. some transactions can use ZK for protecting user data privacy, some for compliance purposes, and others using ZK to validate a zk-rollup state update). The resulting proof can be verified more efficiently having a constant verification time independent from the number of aggregated proofs. This not only enhances the scalability and efficiency of a block-chain system but also makes it more feasible for use cases that require low latency, for example removing the need for expensive proof wrappings.

The SNARKtor protocol is designed to work in a decentralized environment where inde-pendent actors can join and contribute to the recursive proof aggregation process. The protocol is constructed to create competition between provers in order to incentivize fair cost of proof generation. As the proof aggregation process requires the creation and dissemination of many intermediate proofs, the protocol also avoids expensive proof verification during broadcasting, furtherly improving the efficiency of the aggregation process.

In order to describe the flow, let's take as an example an on-chain application that lever-ages ZK proofs to protect users' data privacy. In such a scenario, in order to interact with the application, a user should provide a ZK proof along with the transaction.

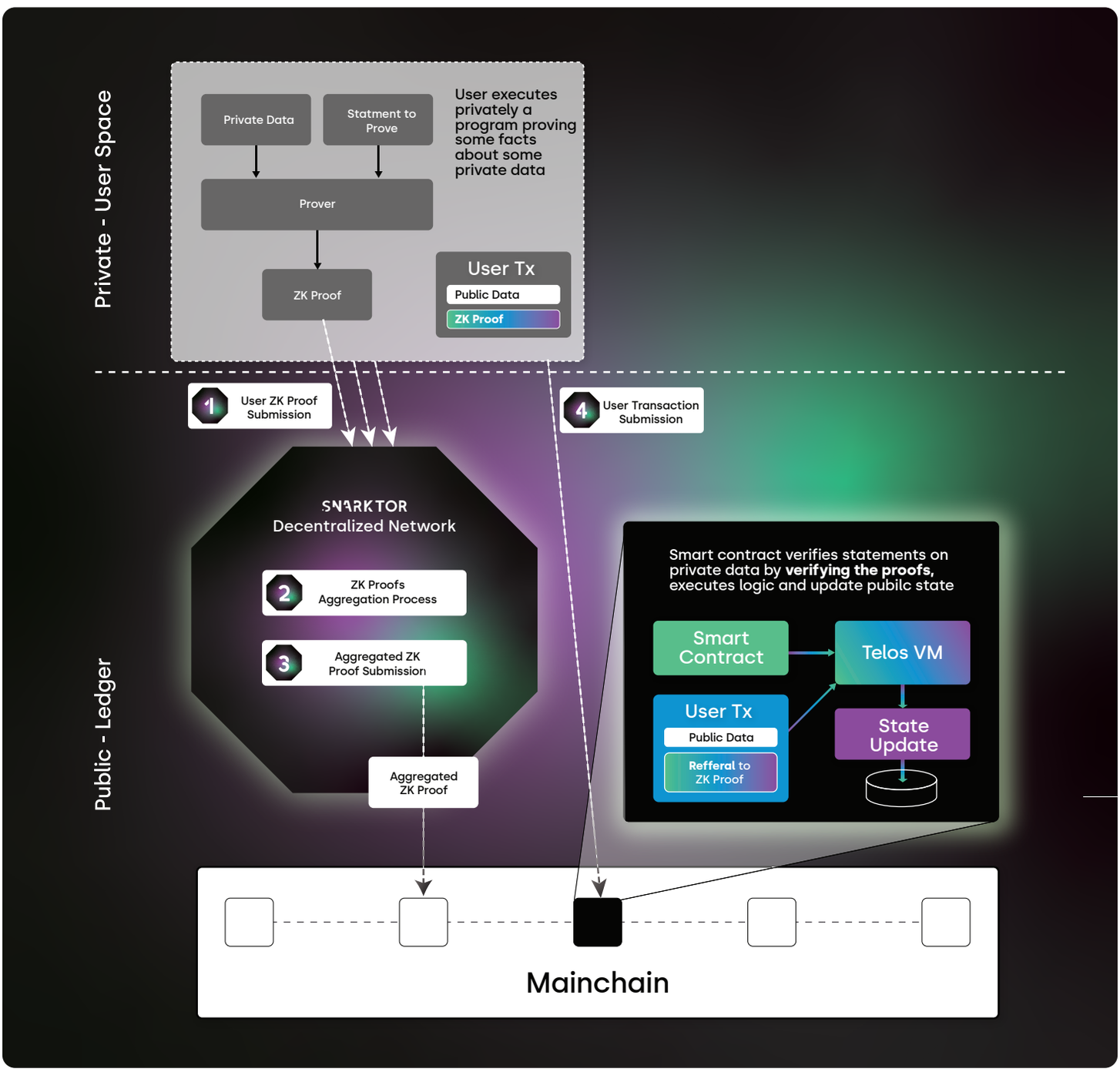


HIGH LEVEL SCHEME

In our model, the user instead of directly submitting the transaction with the proof, will first submit a request using the SNARKtor protocol to process the proof and then submit the transaction on chain referring to that proof.

At a very high level, the SNARKtor protocol continuously processes proofs by aggregating them in a decentralized fashion and periodically submits an aggregated proof on chain. This aggregated proof confirms the validity of all users' underlying proofs, allowing user transactions to indirectly prove to the on-chain smart contract the existence of a valid proof.

This process can be made totally transparent to the user by implementing a well-designed user interface.



ACTORS

Looking more closely, we can identify the following actors participating in the protocol:

Users.
Users submit requests for the ZK proofs they want to be aggregated. When the corresponding ZK proof is aggregated and submitted on-chain, the user can submit a transaction referring to the aggregated proof.

Schedulers.
Special entities that coordinate the proof aggregation process. Specifically, their task is to maintain a sequence of proofs and provide a schedule defining who, how, and when make the computational work of merging proofs.

Provers.
The actual workers who perform the task of merging proofs according to the schedule provided by schedulers.

Submitters.
They submit final aggregated proofs on-chain. From the merging protocol perspective their task is to pick up an aggregated proof and include it into a block or submit to the smart contract (depending on the implementation).

It's important to note that depending on the implementation, Schedulers and Submitters can be selected from the block producers set of the underlying chain in order to inherit its decentralization.

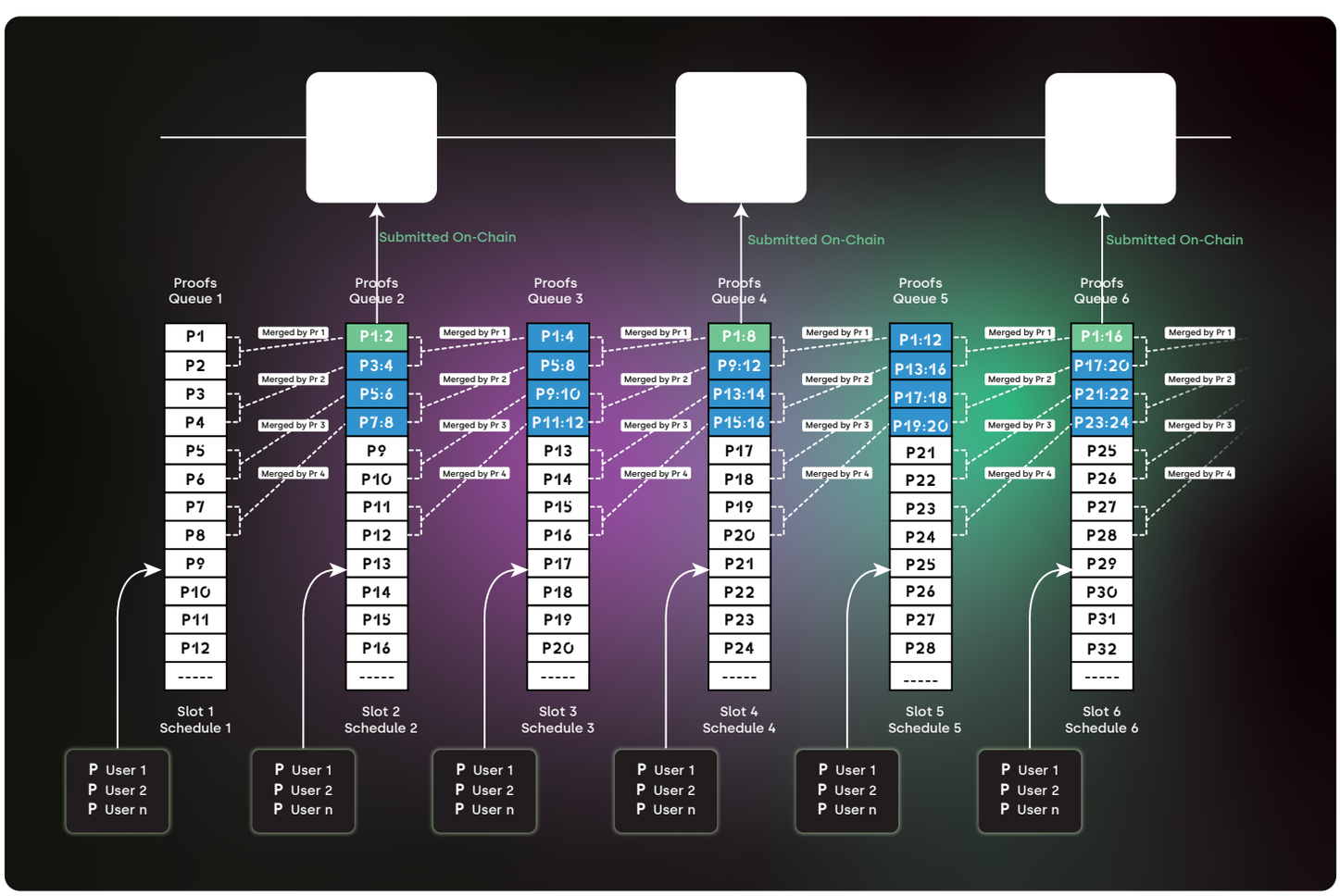
AGGREGATION FLOW

The protocol operates in an environment where time is divided into slots of constant duration.

- Users continuously submit their proofs augmenting the proofs queue.
- Every slot is assigned to a proof scheduler and at the beginning of each slot, the scheduler picks up a set of proofs from the proofs queue and issues a schedule defining which provers should merge what ZK proofs.
- Each assigned prover, merge the assigned proofs and share the resulting merged proof to the other network participants.
- The resulting merged proofs are added to the proofs queue and the relative source proofs are removed from it.
- The schedule loop continues infinitely.

In parallel to scheduling and merging, the submission process is responsible for picking up one of the merged proofs and submitting it on chain. More specifically the submission environment is divided into submission epochs. Depending on the implementation, the submission epoch can be bound for example to chain blocks or slots.

- Each submission epoch is assigned to a specific submitter.
- The assigned submitter takes one aggregated proof, finalizes it for submission and submits it on-chain. The submitter is allowed (but not obliged) to submit exactly one proof per epoch.



REWARDS

The rewards are paid out from the collected fees coming from the users' aggregation requests that are kept by the aggregation service. The on-chain component maintains a reward pool from which every actor can withdraw their rewards. Considering that a withdrawal for every generated proof or issued schedule will be very expensive to be processed on-chain, the protocol leverages SNARKs also to allow an actor to collectively withdraw many rewards with a single transaction.

For what regards incentives, distribution of fees between participants, the further scaling solution avoiding proofs verification during broadcasting and many other important aspects of the protocol we invite you to read the SNARKtor whitepaper https://eprint.iacr.org/2024/099.pdf

CONCLUSIONS

Zero-knowledge techniques play an increasingly important role in the blockchain system allowing a wide range of different applications, such as transactions protecting users data, compliance, ZK rollups, trustless interoperability between chains, and many more. Nevertheless, despite all developments in optimization of ZK proving systems, their execution on-chain is still expensive.
With SNARKtor we propose a decentralized solution to substantially reduce the verification cost and improve scalability of an existing blockchain system. As the blockchain landscape evolves to accommodate ever-expanding transaction volumes and diverse use cases, developing such protocols become increasingly important to realize the vision of decentralized, scalable, compliant and privacy-preserving blockchain networks.