Team: Parity Technologies in collaboration with Stake Technologies (Plasm)

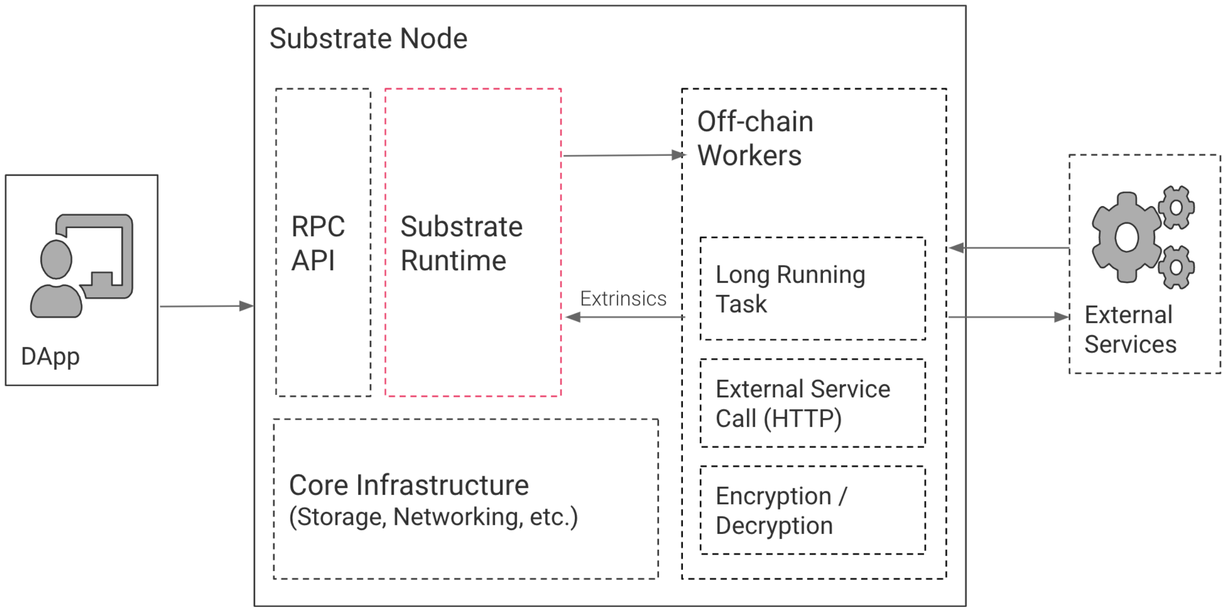
Parity Technologies Limited specializes in developing blockchain-related peer-to-peer software and decentralized technologies. The primary focus of Parity Technologies Limited's business is building the technology that will form the basis of the next evolution of the world wide web, Web 3.0. As part of this goal, Parity Technologies is developing Polkadot, a layer-0 protocol that provides a secure environment for cross-chain composability of applications and protocols across multiple shards. Parity Technologies also offers consulting services.

Plasm Network is built on Parity Substrate and aims to run as a Polkadot parachain (a sovereign blockchain running within the Polkadot ecosystem). Plasm aims to be a dApps hub on Polkadot that supports Ethereum and cutting edge layer2 solutions like ZK Rollups. Stake Technologies, Inc. is the company behind Plasm Network

Parity Technologies Github: https://github.com/paritytech
Plasm Network Github: https://github.com/PlasmNetwork

Substrate

Substrate is a modular framework for building blockchains, making it easy to create a custom blockchain optimized for any use case. Substrate blockchains are natively compatible with Polkadot.

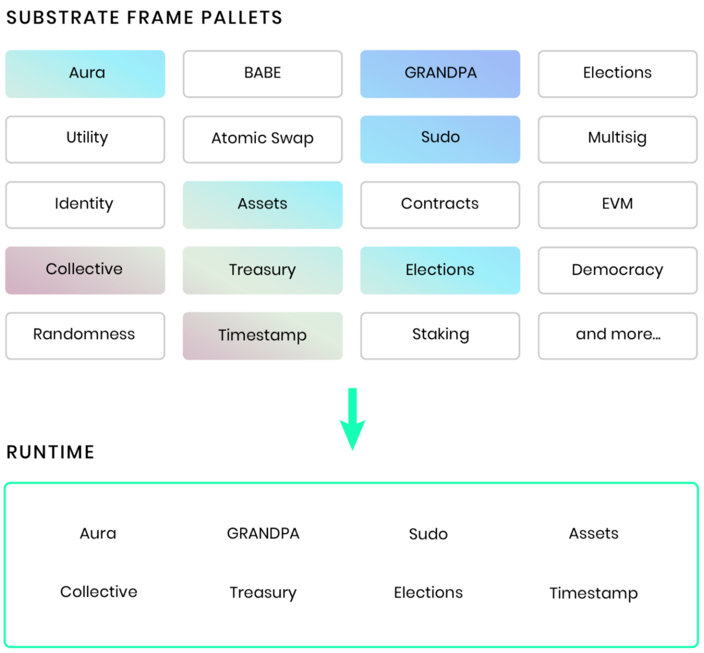


Substrate allows the runtime — the business logic — to be customized to meet specific needs. It includes what items to store and how users can interact with the stored information. The Substrate software development kit (SDK) includes a number of primitives like dispatch logic, customizable fee systems, errors, events, and hooks to perform operations at the start of each block or trigger off-chain tasks. These features allow a customized blockchain to be built easily and quickly. Substrate also provides templates to write individualized pallets (Substrate modules) to extend the system and add any custom logic needed.

The Substrate client comes with several consensus and cryptographic libraries. In the runtime, the consensus engine to be used can be defined and changes can be made without forking using a runtime upgrade.

Technical Advantages of Substrate

- **Forkless upgrades**: Substrate allows upgrades to be built into the runtime, avoiding the necessity of the hard forks found with many other protocols.
- **High customizability**: Substrate is a customizable blockchain framework. Core features (Frame) are provided by Parity Technologies and several Pallets are available, with more continually being created by developers in the ecosystem. Based on the Rust programming language, developers can build a blockchain based on their individual requirements.



- **Shared Security**: Security is the most important and scarce resource in terms of public blockchains. All parachains that are connected to the Polkadot Relay Chain by leasing a parachain slot will benefit from the economic security provided by the Relay Chain validators.
- **XCMP**: Cross-chain messaging passing (XCMP) allows data and tokens to be exchanged across the parachains in the Polkadot ecosystem. Bridges to external networks also allow communication with external networks such as Bitcoin and

Ethereum. Thus, as a parachain on Polkadot, Brave could interact with other parachains in the ecosystem and external networks in a trustless way without any intermediary.

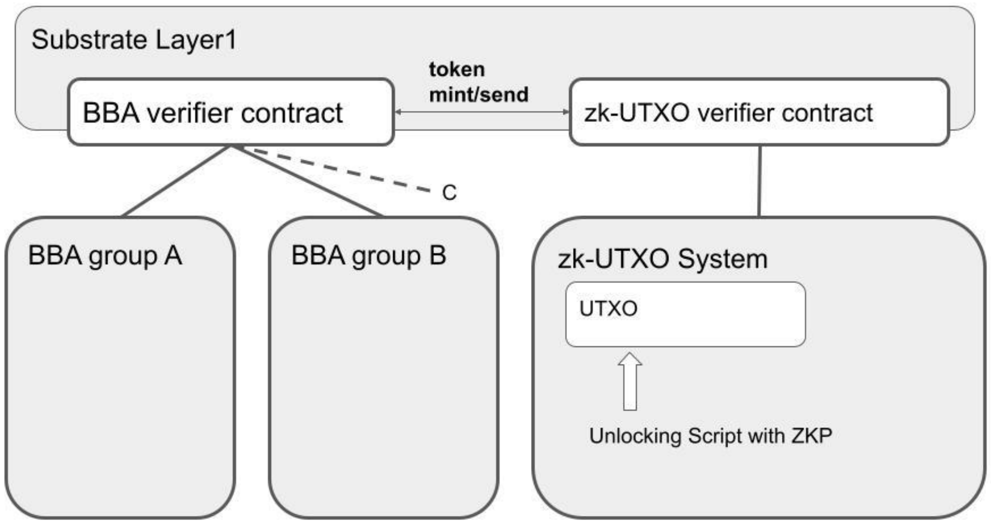
- **On-chain governance**: Governance is the key to maintaining a decentralized and sustainable public blockchain ecosystem. Polkadot has already demonstrated the efficacy of on-chain governance. Additionally, using Substrate, developers can create their own governance structure to suit the needs of their parachain.

Proposal Description

In this section, we explain three independent solutions on Substrate.

Solution 1: Substrate Customization and Scalability - usage of zkp

The core concept is parallelizing verifications of pairing processes in BBA, and attributing data availability to the offchain side (user/provider). Substrate L1 and relevant zkp systems can support this system.



We need only two contract codes in Layer1, the **BBA verifier contract** and **zk-UTXO verifier contract**.

BBA verifier contract:

This solution does not depend on the specification of BBAs, focusing on reducing the pairing calculation cost of BBA verification. Users and advertisements will be labeled to put in a set, and a limited number of validators will verify the pairing calculation. The set is started with an advertisement providers' group by user's selection or judging the user's feature. If one user joins in many groups, scalability will be compromised. This separation allows L1 verification to be parallelized, separating roles of BBA validators from roles of Substrate L1 validators. Pairing calculations in BBA's zkp verification process will be skipped with relevant proof data submitted to L1 chain and staying fraud-provable. This zk-fraud-proof has a particular feature in comparison with stateful fraud proof systems like Optimistic Rollups, this does not have a data availability problem which requires offchain(L2) validators to have the full storage and the full tx history data, since all needed information is already included in the zkp proof. For more information, see this post written by Leona Hioki, core dev at Plasm Network.

This validation process is so mathematically obvious that even a very light JavaScript code can verify it instantly. This would enable all Brave browser users to be verifiers immediately, helping prevent security incidents.

zk-UTXO Verifier Contract:
zk-UTXO is the utxo where unlocking script is a zkp verification.This can be used to bring privacy to payment. Using recursive zkp such as Aztec's zkmoney would add scalability. Currently, zkmoney uses approx 5,000 gas per tx. Though exchanging BBA's tokens through zk-UTXO can be executed in Ethereum's onchain contracts, zk-UTXO on Substrate makes the cost much cheaper. The destination will be attached in txs with BBA token verifications, hashed to the entryhash of zk-UTXO verifier contract, and aggregated to one mint execution by updating the merkle root written in the contract in the usual Rollup manner.

Customized L1 Blockchain (Substrate):
Since Substrate is a customizable blockchain framework, the Brave Substrate-based blockchain can be optimized to suit Brave's needs, such as scalability and reduction of storage costs, by optimizing Brave's specific blockchain logic/runtime.

In this section, we will consider customizing the storage of L1 blockchain. We define Proof-Action as a specific action in this chain.

Proof-Action contains two parts: Hashed-Proof and Proof.
Hashed Proof is a hashed preimage of zkp proof with sha256. This data eternally remains in the storage of validator nodes. Proof is zkp-proof data which can be verified with a verifying key. This data remains in the only validation period (like 7 days). Since most of THEMIS's logics are about the asset settlements and their relevant conditions, we assume that the asset logic does not affect the other's assets. So, the relevant data availability of proofs can be attributed to asset holders. This feature allows L1 validator nodes to free upstorage that was used for the proof of BBA and the merkle proof of zk-UTXO transactions. Hence, the relevant costs will be dramatically reduced.

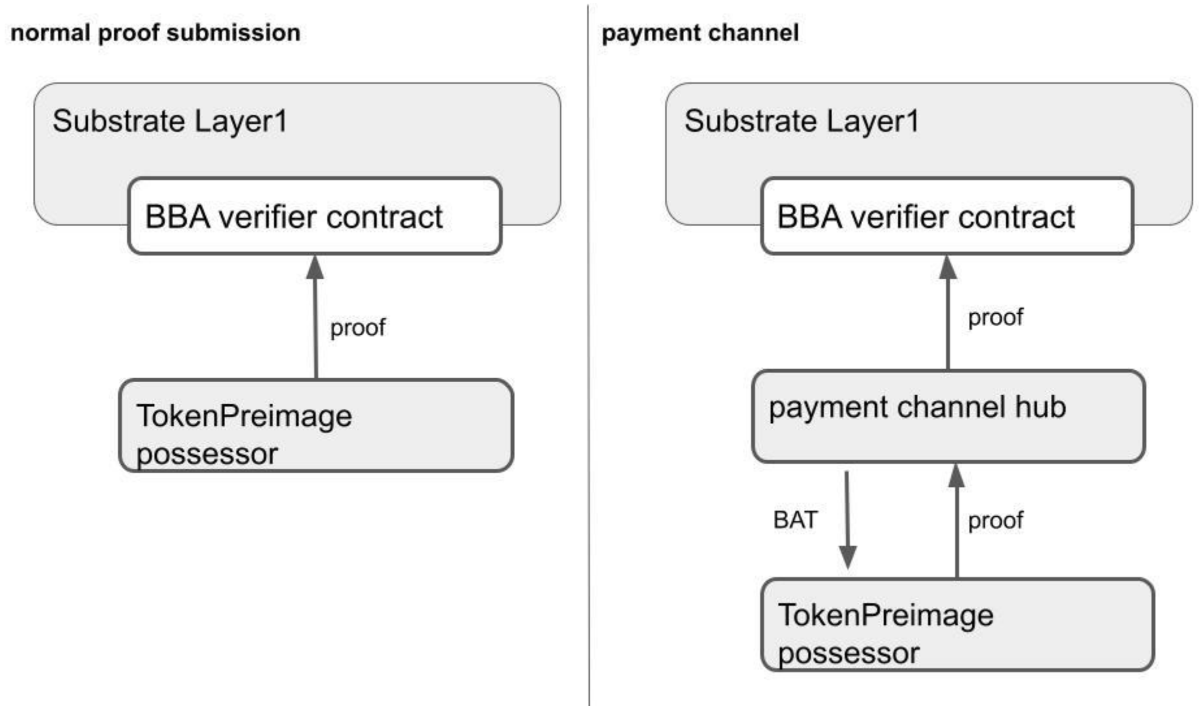
On-chain Proof data in the validation period will be used for the publication on all relevant nodes.

Solution 2: HTLC with zk-proof:
Costs can be cut further by introducing a payment channel to this system. We can keep the solution simple by implementing a one-way payment channel for users to easily receive rewards.

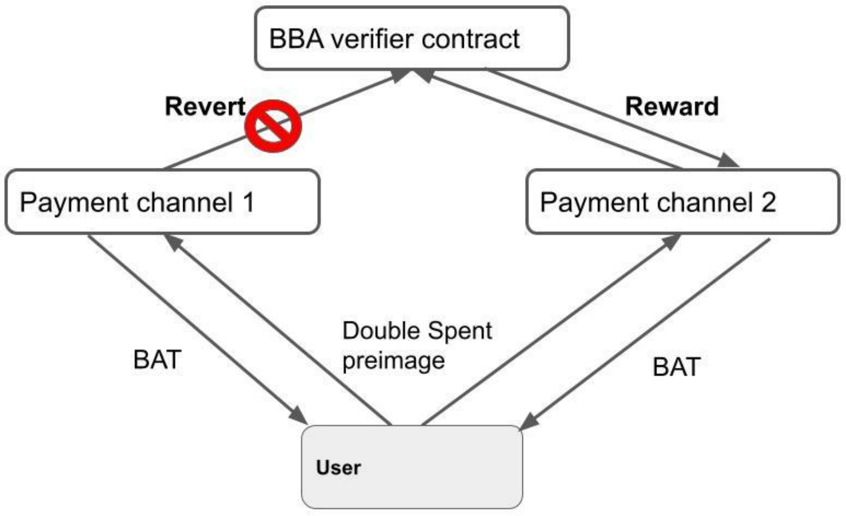
Advertisers or other fund providers can deposit to the channel with zk-UTXO, and set the HTLC payment requirement to submitting BBA proof. Here are the steps.

Ads server and a user (users) set a simple HTLC (Hashed Timelock Contract) with zk.

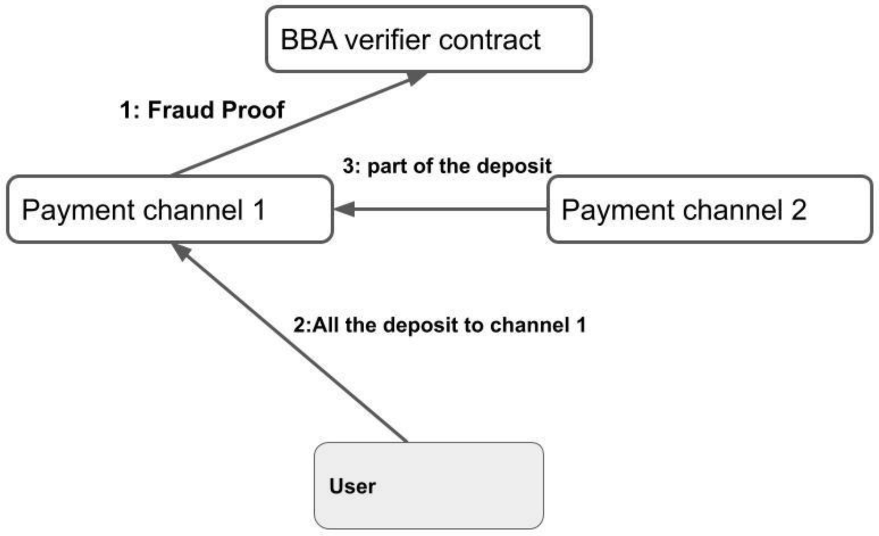
1. Ads server provides the beginning and closing date of HTLC.
2. The user can get paid by the HTLC after closing the contract.
3. The user or ads server close the channel and claim BAT (one-way)
4. Only the final settlement is reported to contract.

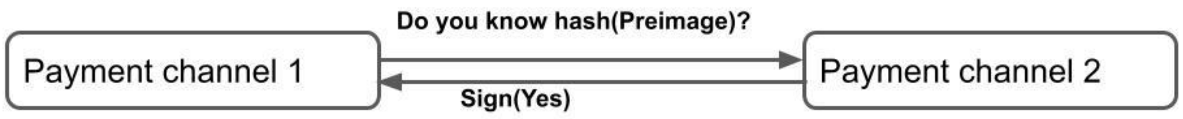


In the current BBA spec, a double reward with a token will be reverted on the verifier side. Token preimage can be spread off-chain, then this should be then reverted on chain. However, the payment channel is tolerant to such delay by its purpose, then the channel which gets closed later will be reverted in this case. To prevent this, slashing the double reward with a fraud proof results in the slasher taking all the deposit of the slashed channel and his channel for the slashed user. Providers need to protect themselves by getting the signed response from other relevant providers that express whether or not they know the token preimage.



then





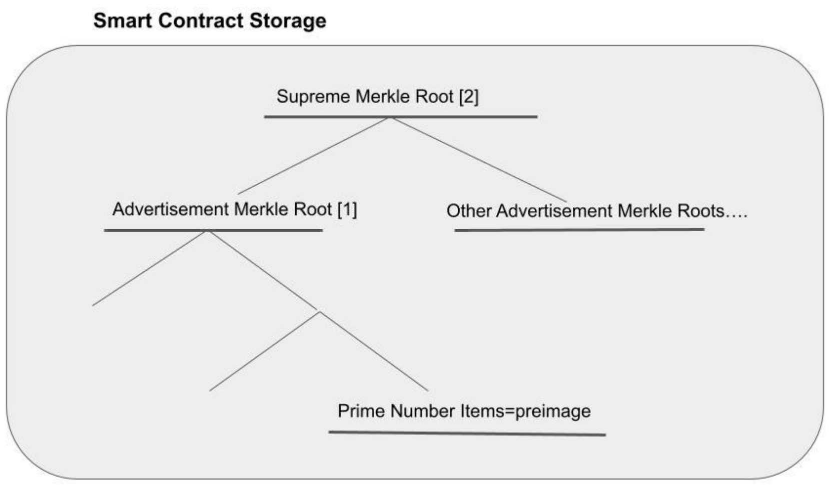
Sybil attack can be banned with the upper and lower limitations of deposit amount, since the reward amount of almost all users will be in a certain range. Robot deterrents like CAPTCHA can be effective to limit the mass creation of addresses.

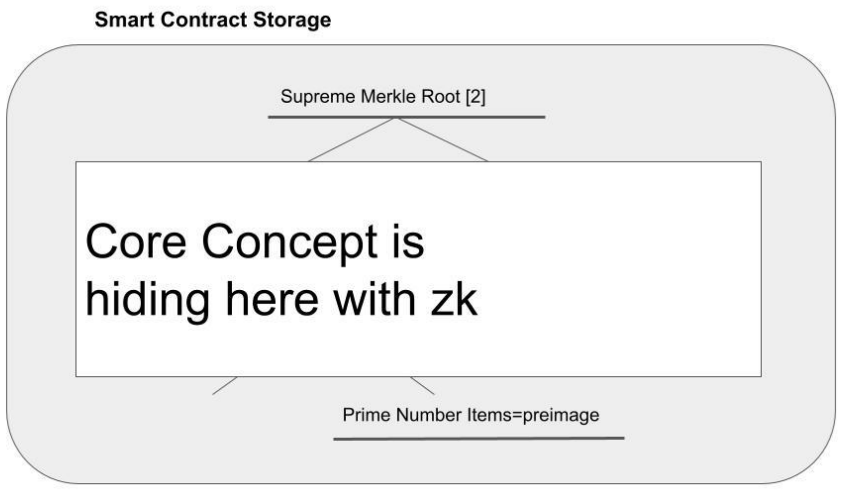
Solution 3: New version of BBA

Disclaimer: This solution may change the basic logic of the current THEMIS / BBA.

The core concept of this solution is that the loss of siblings (components of a ZK proof) can be used for privacy. In the current form, it is hard to prevent the advertisers' malicious behavior using BBA because evidence that a user has accepted a certain advertisement can be easily submitted by the advertisement provider. In other words, the logic can be manipulated by the provider. Thus, we need to conceal it from all advertisement providers, in the following way:

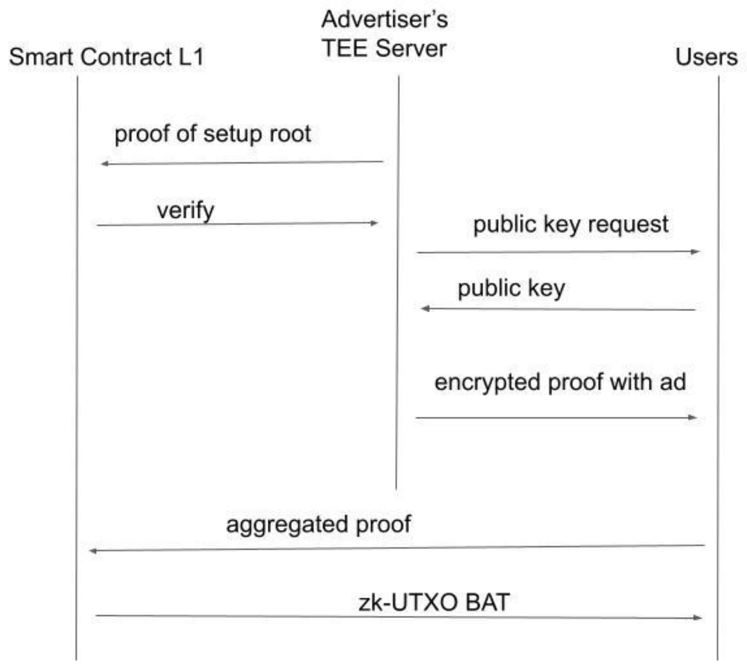
1. Vector commitment items (=token preimage) are set up with a TEE.
    a. Items are hashed numbers or prime numbers
    b. The vector commitment scheme is in the form of a merkle tree or RSA accumulator.
2. At the same time, the TEE server makes a proof of advertisement token preimage and the merkle root[1] of this token preimage with ZK-S*ARK.
3. The smart contract accepts the root and the proof, records it in the merkle tree and updates the supreme root[2] of the merkle tree.


Smart Contract Storage
Supreme Merkle Root [2]
Advertisement Merkle Root [1]
Other Advertisement Merkle Roots….
Prime Number Items=preimage


Smart Contract Storage
Supreme Merkle Root [2]
Core Concept is hiding here with zk
Prime Number Items=preimage

Second, the TEE server attaches one item and its vector commitment proof to the advertisement data and sends it to the user. The user records it. The proof is encrypted with the user's public key.

Third, a user makes the aggregated ZK-proof which proves the inclusion of items to the merkle root recorded in the smart contract and proves non-inclusion of the paid item root to prevent the double spend. This may take 10~100 seconds with execution at the client-side WASM program. If siblings are hidden in the ZK-S*ARK's proof, then no one else but the user can know the source advertisement. The non-inclusion proof can be implemented in two ways: an RSA-accumulator or SMT proof provided by the TEE server.



Fourth, the proof of a token preimage will be exchanged to UTXO ownership, whose unlocking script is ZKP. This provides privacy to the original users after the exchange. The transfers of UTXOs need to be aggregated for the sake of scalability. This can be implemented with zk^2 like Aztec.

To avoid the side-channel attack to the TEE, the TEE could potentially be replaced with functional encryption/indistinguishability obfuscation. But these are much harder to implement.

Contribution Summary and Performance Result/Estimation

Since Substrate is a modular framework, a Substrate-based blockchain can be easily customized in order to meet business and protocol requirements. For example, it is pretty straightforward to implement truly gasless transactions while also having full flexibility on the blockchain governance aspects.

Every Substrate-based blockchain can have different throughput depending on the complexity of the business logic. Depending on the number of compute and I/O operations, the weight of a block and hence throughput of the blockchain can change.

In light of the above, and considering the proposal submission timelines, we are unable to provide concrete throughput numbers. The throughput and performance numbers can change between a PoC and an optimized production solution.

Changes to the THEMIS protocol

- Though THEMIS V1 is a PoA network, our implementation can be more decentralized and supports various algorithms without hard forks.
- Utilizing layer2 solutions that solve L1's scalability and finality problems without sacrificing decentralization.

Solution Components and High-level Estimates

The following are components identified for the proposed solutions. This is not an exhaustive list. These are only the core components needed for the ZK solutions we have proposed in this document.

The additional components needed for the integration of this solution with Brave client, servers and advertisers will be estimated and implemented additionally, based on a requirement analysis after this solution is accepted.

The estimates provided below are subject to change and are only indicative.

1. **Solution 1 PoC**
    a. Verifier contracts implementation (3 months)
    b. BBA group assignment logic (1 month)
    c. zk-UTXO (2 month)
    d. Substrate L1 (3 months)

2. **Solution 2 PoC**
    a. HTLC implementation for payment channels with BBA logics (5 months)
    b. Double Reward Prevention (3 month)
    c. zk-UTXO channel deposit (1 month)

3. **Solution 3 PoC**
    a. Ad Server with TEE implementation (2 month)
    b. RSA accumulator implementation (2 month)
    c. ZKP circuits implementation (2 months)
    d. Verifier Smart Contract (1 month)

References

Substrate: https://www.substrate.io/
Substrate Documentation: https://substrate.dev/docs/en/

ZK-Rollups on Substrate: https://github.com/PlasmNetwork/ZKRollups
ZKP on Substrate: https://github.com/patractlabs/zkmega
SubstraTEE: https://github.com/scs/substraTEE