Confidential Ethereum Smart Contracts

Second State and Oasis Labs

Abstract—As Decentralized Finance (DeFi) applications take off and pose to revolutionize the world's financial infrastructure, the current public blockchains fall short in financial privacy. That is a significant barrier for traditional financial institutions to enter DeFi. The Oasis Ethereum ParaTime proposes a new privacy-first blockchain network to execute Ethereum smart contracts and trade Ethereum digital assets. It provides full transparency into smart contract code and logic. The smart contract is executed on replicated virtual machines on a decentralized set of validator nodes, and the result is guaranteed to be correct through the consensus of those nodes. Yet, the contract data and state are confidential even to the validators and node operators. The transactions and results are only revealed to users who submit the transactions.

1 Introduction

ETHEREUM [1] is a suite of protocols that specify how to replicate state (i.e., stored data) on a decentralized and trustless set of computer nodes. The nodes can run user-submitted programs, known as smart contracts, to manipulate the state. Ethereum requires that all nodes must manipulate the state in exactly the same way, and hence maintains replicated state across all nodes. The protocol guarantees this through an underlying blockchain. The blocks record the state, and each new block (i.e., changes to the state) must be accepted by all nodes through consensus.

A key characteristic of the Ethereum public blockchain is its transparency. Anyone can join the network and become a node. This transparency enables anyone in the public to download and verify full transaction history. But it also allows motivated parties to analyze the state data and transactions to derive identities and their relationships. In the context of decentralized finance (DeFi) applications, malicious actors can infer, and preemptively act upon, transactions made to financial smart contracts.

A recent case of front-running attack [2] demonstrates the need to protect transaction confidentiality. When DeFi transactions are all in the open, it is far more profitable to engage in bot-based front-running instead of developing trading strategies and algorithms. Today's blockchain DeFi is a very hostile environment for small investors or technically un-sophisticated investors.

Another example of DeFi attack is the vampire mining scheme of SushiSwap [3]. It threatened not only to copy the IP and software, but also to "steal" the community from Uniswap exchange [4] since all Uniswap LP addresses are publicly viewable in liquidity pool smart contracts. For DeFi exchanges, lenders, and service providers, there is also a need to protect the internal data of their smart contracts to protect user information.

In this paper, we present the design and implementation of an Ethereum-compatible blockchain system that protect privacy in its smart contracts even from node operators themselves. We aim to preserve the transparency of smart contract code and logic, and the integrity of smart contract execution. At the same time, we aim to hide transaction contents, and to selectively encrypt state variables in contracts.

Our specific requirements include the following.

Decentralized public blockchain: We must create a public and permissionless network. Anyone can become a node and process all transactions. Node operators are anonymous and cannot have access to raw transaction data or designated confidential data even on their own node.

Backward compatibility: All existing smart contracts must work as they are today on Ethereum.

Fine-grained confidentiality: For newly created Solidity contracts, developers can mark part of the contract state as confidential. Node operators execute the smart contracts, but are never able to see the confidential state in the contract.

Encrypted transactions: The client sends encrypted Ethereum transactions to nodes and receives encrypted responses. The node broadcasts transactions to other nodes on the network in encrypted messages.

Our proposed solution is built upon well-established research in privacy computing.

2 Privacy-first infrastructure

The Oasis protocol is built on solid theoretical foundation and provides a technical infrastructure to support confidential transactions in a public blockchain network.

The Oasis protocol [5] specifies a Tendermint-based Proof-of-Stake blockchain [6] mainnet that serves as the single source of truth for transactions on the network. Each Oasis validator node can choose to run additional runtime software to perform arbitrary computation tasks with off-chain (i.e., confidential) data, such as machine learning with private data, and record the hash of the consensus results on the Oasis blockchain.

The runtime software on nodes are parallel to each other. It is also called a ParaTime [5]. Each node can choose a number of ParaTimes it wants to support. All nodes running the same ParaTime must produce the same results for transactions to be recorded on the Oasis mainnet. The Oasis protocol utilizes consensus committee nodes to detect discrepancy in ParaTime computation results and to ensure ParaTime integrity. That design is more efficient than Polkadot's parachain [7] and Ethereum 2.0's shard chain approaches.

The Oasis protocol requires confidential ParaTime to run inside a Trusted Execution Environment (TEE) [8]. Each TEE receives secrets from the rest of the network's already-bootstrapped TEEs (i.e. the key manager) once it's been attested. The ParaTime then performs computation inside the TEE utilizing the confidential information.

For example, the TEE receives its private key from the key manager from which it derives its state encryption key. Nodes use the public key to encrypt transaction data directly into the ParaTime, thus achieving end-to-end confidentiality. Any external party can encrypt its data with the public key, and send the encrypted data to the TEE software for processing. The data is decrypted and processed inside the TEE. Outside parties can not see the data. They can only see computation results the TEE software chooses to output.

The TEE is often hardware based. It ensures that even the node operator does NOT have access to private data, including immediate computation results. All major CPU vendors have rolled out their TEEs (e.g., ARM TrustZone, Intel SGX, and AMD SEV). Those CPU-based TEEs are often called enclaves.

The Oasis protocol supports confidential information exchange between TEE runtime nodes. It provides a decentralized mechanism for TEE nodes to attest and register themselves, and to broadcast encrypted transactions in a blockchain network. The Oasis runtime, which runs on all Oasis node, provides key management capabilities to facilitate secure key exchanges between TEE nodes.

In the next sections, we will discuss how the Oasis Ethereum ParaTime provides strong confidentiality and privacy protection for Ethereum smart contracts.

3 Ethereum on Oasis

The confidential Oasis Ethereum ParaTime [9] implements the Ethereum protocols on the Oasis blockchain network. Confidentiality of Ethereum transactions and state is attained by encrypting both and processing them only within the confines of the trusted execution environments (TEEs) in which the ParaTime runs. Backward compatibility is maintained by selective encryption of state variables and preservation of non-sensitive transaction headers.

The ParaTime provides an alternative public Ethereum blockchain, but with Oasis's strong privacy and confidentiality features. The Oasis Ethereum ParaTime software consists of the following components.

- An Ethereum Virtual Machine (EVM) [10] based on OpenEthereum [11]. It is fully compatible with the current Ethereum blockchain. It executes all existing Ethereum smart contracts, and hence is capable of managing smart contract-based digital assets, such as ERC-20 tokens, ERC-721 tokens, DAOs, Uniswap exchanges etc.
- An Ethereum flavored WebAssembly (Ewasm) virtual machine [12] based on the Second State VM (SSVM) [13]. Ewasm is the next generation execution engine of the Ethereum 2.0 protocol. It provides many performance and feature enhancements over EVM 1.0. In the Oasis Ethereum ParaTime, we experiment with new privacy features in Ewasm.
- An Ethereum host environment for state data, such as Ethereum account-based transaction and state semantics.

Even as the Oasis Ethereum ParaTime is fully compatible with today's Ethereum applications, it is much faster than the Ethereum mainnet due to its Proof-of-Stake and lightweight consensus. Ethereum 2.0 is moving to Proof-of-Stake for the same reasons.

Ethereum compatibility also means that Ethereum digital assets, such as ETH and any ERC-20 tokens, can move between Ethereum and Oasis Ethereum in a decentralized and permissionless manner. Anyone can setup smart contracts to do Atomic Swap of tokens without any intermediaries.

But most significantly, the Oasis Ethereum ParaTime allows Ethereum transactions and smart contract states to stay confidential even from the node owners and operators. It is a blockchain runtime designed for financial privacy and DeFi.

4 Confidential transactions

The Oasis Ethereum ParaTime protects Ethereum transactions both in transit and during execution.

An Ethereum user generally submits a transaction to the network via a Web3 gateway, which exposes a JSON RPC protocol. The RPC gateway should only be accessed through HTTPS / TLS to ensure confidentiality of the transaction from the client to the node. Since the ParaTime software, including the RPC (Remote Procedure Call) gateway, runs inside the TEE, end-to-end confidentiality is guaranteed.

In this setup, the client must trust the RPC gateway since the HTTPS / TLS web server software on the node would be able to decrypt the transaction into plain text. However, since anyone can create an Oasis Ethereum ParaTime node on the network, one can always find or create a trusted RPC gateway.

Another approach to remove the trusted RPC is to use client-side encryption to communicate directly with the paratime's TEE (e.g., the oasis.js [14]). However, this solution is less desired since it breaks Web3 compatibility, which would require porting almost all Ethereum-based applications to the new library.

Once the RPC node receives the transaction, it broadcasts the transaction through the Oasis protocol, which ensures that the transaction is properly encrypted for each node's TEE.

Each Oasis Ethereum ParaTime node receives the transaction, decrypts it inside the TEE, and executes it in the TEE. This ensures that no one, not even the node operator, can see what's inside the transaction.

At the same time, the Oasis protocol guarantees that the transaction is decryted and executed correctly according to the smart contract. All Oasis Ethereum ParaTime nodes must agree on the hash of the execution result in order for the transaction to be finalized, confirmed, and included in the blockchain.

5 Confidential contract state

The execution of the transaction produces a state change that must be recorded. For example, it could change account balances or variable values inside smart contracts. The Ethereum mainnet records the state in an unencrypted database on each node. For the Oasis Ethereum ParaTime, we must keep the state confidential as well.

A straightforward approach is to encrypt all data inside the TEE before they are written to the database. This way, only the node's TEE can read from its state store and execute new transactions against it.

However, the downside of this approach is poor performance. When a contract function is pure view, a Web3 RPC gateway should be able to execute it locally by simply reading from its local Ethereum node's current view of state. The pure view functions can be executed without any gas as it does not alter the state of the blockchain and hence requires no consensus. But if all state data is encrypted by the TEE, even the read-only pure view operations must be done via a transaction across the Oasis protocol since, by design, the Web3 gateway operator does not have access to the ParaTime's state encryption key.

6 Fine-grained confidentiality

A better design is to have finely-grained confidentiality in smart contracts. Smart contract developers just need to add the confidential modifiers in their Solidity contract source code. All fields marked with confidential will stay encrypted and not revealed to any human. In the following example, the data field can be set by external transactions, but its value can never be viewed.

```
contract A {
  confidential int data;
  function f(int in) public {
    data = in;
  }
}
```

Together with transaction confidentiality, the smart contract's data field is secured as follows.

- The transaction to call function f is encrypted at all times even without any annotation, as we discussed in the previous section;
- Each node decrypts and processes this transaction inside the TEE, and saves an encrypted data field to the node's local state store, which the node operator cannot decrypt;
- There is no contract function a user can call to get the current value of data.

The Oasis Ethereum ParaTime would support the confidential language keyword across its Ewasm compiler toolchain and runtime.

- The Second State VM (SSVM) supports two additional bytecode instructions for memory operations. The sload and sstore instructions are similar to the standard WebAssembly load and store instructions [15]. The SSVM executes sload and sstore instructions with private encryption keys in the TEE to make sure that the confidential marked smart contract data is properly encrypted and decrypted before saving to or loading from the data store.
- The Second State Ewasm compiler (SOLL) [16] supports the confidential keyword. It emits the sload and sstore bytecode instructions for memory operations on the confidential variables.

With the Oasis Ethereum ParaTime, we can ensure that confidential data inside smart contracts are protected while public data can be accessed freely from the node at low cost and high performance.

7 Conclusions

In this paper, we introduced the Oasis Ethereum paratime for confidential Ethereum smart contracts. Through token bridges and Atomic swap contracts, this new Ethereum network can provide a secure platform for DeFi and other real world Ethereum Dapps that require privacy for its users.

Acknowledgment

This work was supported in part by a grant from Oasis Foundation.

References

[1] Buterin, V., "Ethereum Whitepaper" https://ethereum.org/en/whitepaper/ 2020

[2] Robinson, D. and Konstantopoulos, G., "Ethereum is a Dark Forest," https://medium.com/@danrobinson/ethereum-is-a-dark-forest-ecc5f0505dff 2020

[3] Sushiswap, "The SushiSwap Project," https://medium.com/sushiswap-org/the-sushiswap-project-8716c429cee1 2020

[4] Adams, H., Zinsmeister, N. and Robinson, D., "Uniswap v2 Core," https://uniswap.org/whitepaper.pdf 2020

[5] The Oasis protocol project, "The Oasis Blockchain Platform," https://docsend.com/view/aq86q2pckrut2yvq 2020

[6] Kwon, J., "Tendermint: Consensus without Mining," https://tendermint.com/static/docs/tendermint.pdf 2014

[7] Wood, G., "Polkadot: Vision for a Heterogeneous Multi-chain Framework," https://polkadot.network/PolkaDotPaper.pdf 2016

[8] Lee, D. et al., "Keystone: An Open Framework for Architecting Trusted Execution Environments," https://dl.acm.org/doi/pdf/10.1145/3342195.3387532 2020

[9] Second State, "The Oasis Ethereum ParaTime," http://www.oasiseth.org/ 2020

[10] Wood, G., "Ethereum: A Secure Decentralised Generalised Transaction Ledger," https://ethereum.github.io/yellowpaper/paper.pdf 2020

[11] OpenEthereum, "The fast, light, and robust client for Ethereum-like networks," https://github.com/openethereum/openethereum 2020

[12] Beregszaszi, A. et al., "Ewasm Design Overview and Specification," https://github.com/ewasm/design 2020

[13] Second State, "The Second State VM," https://github.com/second-state/SSVM 2020

[14] Oasis Labs, "A web client for the Oasis platform," https://github.com/oasislabs/oasis.js/ 2020

[15] WebAssembly Community Group, "WebAssembly Instructions," https://webassembly.github.io/spec/core/text/instructions.html 2017

[16] Second State, "SOLL: a new compiler to generate Ewasm from Solidity or YUL," https://github.com/second-state/SOLL 2020

Second State Second State builds the next generation open source "operating system" for the cloud and the decentralized web. The Second State Virtual Machine provides a managed alternative to native code, and is ideal for building AI and big data microservices. It is also the execution engine for leading public blockchains.

Oasis Labs Oasis Labs is an international team made up of researchers, security experts, and privacy advocates — all working together to build a platform for a responsible data economy. Founded in 2018 by Dawn Song, an award-winning professor at the University of California at Berkeley, Oasis Labs is backed by investors including Andreessen Horowitz, Accel, Binance and many others.