Our idea is that the current privacy problems identified by confidential smart contracts have not well covered multi-party scenarios, thus also lacking corresponding practical solutions. Therefore, we first identify the Multi-party Transaction

(MPT), then propose a novel framework, CLOAK, for confidential smart contracts.

CLOAK is a pluggable and configurable framework for developing and deploying confidential smart contracts. CLOAK allows users to specify privacy invariants (what is private data and to who is the data private) in a declarative way. Then, it automatically generates runtime with verifiably enforced privacy and deploys it to the existing EVM-enabled platforms (e.g., Ethereum

) and TEE devices to enable the confidential smart contract.

Problem Definition

The public verifiability of public/private transaction

:

Currently, prominent blockchains (e.g., Ethereum

) are typically modeled as State Machines and require that every single step of state transitions be verified by all nodes. Thus, we say the state transition is publicly verifiable

and the transaction causing the transition has public verifiability

.

Look at the figure followed. when a normal transaction is packaged in a block, its transaction parameters, old states, function logic, return values, and new states are all public; therefore, we call it PUT

(public transaction). Similarly, as for a private transaction (PRT

) of confidential smart contract that takes secrets from a single party (considered in ZKRollup

, Ekiden

, and CCF

), although its components and even contract code can be private, it is also required to prove the validity of it caused state transition from old state commitment to new state commitment, i.e., a PRT also requires public verifiability.

PUT refers to a public transaction. PRT refers to a private transaction that takes secrets from a single party. MPT refers to a multi-party transaction that takes secrets from multiple parties. Dashed thick line arrows refer to reading blockchain while solid thick line arrows refer to writing blockchain. \sigma

refers to the world state. C_*

refers to a commitment to a secret *

.

Towards multi-party transaction

:

We expand the public verifiability of PRT to multi-party scenarios and call it Multi-party Transaction (MPT

). We model the MPT as the formula below. The proof

achieves an MPT's public verifiability by proving both variables' relation and privacy policy. Variables' relation means that anchored by public C_{x_i}, C_{s_i}, C_f, C_{s'*i}, C*{r_i}

, a MPT confidentially executes f(x_1, \dots, x_n, s_1, \dots, s_i)

and outputs private r_i, s'_i

to their corresponding P_i

. The privacy policy of an MPT means that secrets are exactly fed by and delivered to different parties P_i

and their confidentiality is kept in the whole process.

\begin{aligned} \mathrm{MPT}: &\ & C_{s_1}, \dots, C_{s_n} \overset{C_f,~C_{x_1}, \dots, C_{x_n}}{\Longrightarrow} \ &

\qquad C_{s'*1}, \dots, C*{s'*n}, C*{r_1}, \dots, C_{r_n}, proof \ & \quad \mid s_1, \dots, s_n \stackrel{f(x_1, \dots, x_n)} {\Longrightarrow} s'_1, \dots, s'_n, r_1, \dots, r_n \end{aligned}

Why does MPT matter?

We compare the MPT with blockchain-assistant MPC in our paper in detail. Here, we just highlight that the high-level meaning of MPT is like this: we can widely spread the trust of both MPC process and result, and thereby can reuse its result. As a result, we can build the trust of state transition caused by long sequences of transactions with mixed PUT, PRT and MPT. Just as the following figure.

[

Screen Shot 2022-02-23 at 14.46.31

775×100 2.28 KB

](https://ethresear.ch/uploads/default/original/2X/b/b311cfed4df17322fb44ad20c5697179f643ea0e.png)

CLOAK language: new privacy specification language against Solidty

While current frameworks only consider one of PRT (e.g. ZoKrates

, zkay

) and blockchain-assistant MPC (e.g. Hawk

, Fastkitten

), we wanna allow developers to specify all three kinds of transactions in a single contract.

Ownership is all you need

The core idea of CLOAK language is allowing developers to intuitively annotate the ownership (i.e., owners' addresses, inspired by zkay

) of private data without considering the type of transactions. Then, CLOAK analyzes how many parties' secrets are involved in each transaction to infer its type. Specifically, we allow four ownership annotations now.

Annotation

Meaning

@all

value is public

@me

the value is private to msg.sender

@tee

the value is private to TEE enclaves

!id

declare an temporary address variable

@id

the value is private to the declared address variable id

reveal

conciously reveal the private data to other

Demo: CLOAK smart contract

The following is an annotated smart contract demo, we called the CLOAK smart contract.

pragma cloak ^0.2.0;

contract Demo { final address@all _manager; // all mapping(address => uint256) pubBalances; // public mapping(address!x

```
=> uint256@x) priBalances; // private

constructor(address manager) public {
    _manager = manager;
    pubBalances[manager] = 1000;
}

//  PRT-me
//
// @dev Deposit token from public to private balances
// @param value The amount to be deposited.
//
function deposit(uint256 value) public returns (bool) {
    require(value <= pubBalances[me]);
    pubBalances[me] = pubBalances[me] - value;
    // me-owned priBalances[me] is read and writen, thus being a PRT.
    priBalances[me] = priBalances[me] + value;
    return true;
}

//  MPT
//
// @dev Transfer token for a specified address
// @param to The address to transfer to.
// @param value The amount to be transferred.
//
function multiPartyTransfer(address to, uint256 value)
    public
    returns (bool)
{
    require(value <= priBalances[me]);
    require(to != address(0));

    // the me-owned priBalanced[me] and to-owned priBalances[to] are both mutated, thus being an MPT
    priBalances[me] = priBalances[me] - value;
    priBalances[to] = priBalances[to] + value;
    return true;
}

}
```

Enforcing privacy need of all types of transactions

The problem then goes to how to enforce the privacy needs of these different type of transactions. Actually, current approaches mainly fall into two categories that are based on cryptography and TEE, respectively.

Cryptography-based solutions (e.g., zkay

, ZKRollup

) suffer from low performance, specified function, poor multi-party support, and error-prone implementation (the promising zkEVM

could mitigate this problem) since developers are required to implement a set of off-chain cryptographic protocols and on-chain verification smart contracts. Therefore, now, we enforce the PRT and MPT based on TEE-blockchain architecture, where TEE devices evaluate PRT and MPT inside enclaves, strictly control I/O of private dat according to their privacy needs, and generate proof to update the state transition on the blockchain.

Notably, as the performance of MPC/ZKP improves, CLOAK can compile the PRT and MPT to ZKP/MPC-based solutions since the data structure of our on-chain commitment has been designed to allow this potential expansion.

Cloak Overview

Cloak is designed to work with TEE

and EVM-enabled blockchain. It initializes the blockchain in a pluggable manner to become a new architecture, where the blockchain and its clients are enhanced to be the Cloak Blockchain and the Cloak Client respectively, to interact with the Cloak Network.

The figure followed shows the workflow of Cloak to develop, deploy the confidential smart contract and send PRT and MPT. It is mainly divided into three phases, development

, deployment

and transaction

.

[

The overall workflow of Cloak

2367×487 103 KB

](https://ethresear.ch/uploads/default/original/2X/5/5dcc15427ba264f8259ec0638d14151bd20f6247.png)

In the development

phase, we provide a domain-specific annotation language for developers to express privacy invariants. Developers can annotate privacy invariants in a Solidity smart contract intuitively to get a Cloak smart contract. The core of the development phase is Cloak Engine

, which checks the correctness and consistency of the privacy invariants annotation, then generates the verifier contract (V)

, private contract (F)

, privacy policy (P)

and the transaction class

.

In the deployment

phase, Cloak helps developers deploy generated code to specified the blockchain and the Cloak Executors (E)

in the Cloak Network, where the verifier contract is deployed to the blockchain, the private contract and privacy policy are deployed to Cloak Network and the transaction class is held in Cloak Client.

In the transaction

phase, users use the transaction class of Cloak Client to interact with the blockchain and Cloak Network to send private transactions, as well as the MPT.

The concern about TEE-based security

We know one may deem that “using TEE needs to trust its manufacture and unverifiable security model”. To solve this problem, we have systemically identified several problems in verifiably evaluating PRT and MPT based on TEE. Some problems, such as device integrity, public verifiability, negotiation, financial fairness, result delivering fairness, have been answered in our paper. There are also other problems we are working on, such as heterogeneous TEE network and eclipse attack (cheating inputs) resistance.

While plenty of approaches is proposed to strengthen the integrity of TEE and lose the trust assumption of a single manufacturer, the performance advantage of TEE will exist for a long time, especially for verifiable computation against big data. Therefore, it deserves to be used for contributing to the web3 economy.

Conclusion

To the best of our knowledge, CLOAK is the first to allow developers to specify PUT, PRT, and MPT in a single contract simultaneously. It is also the first to achieve public verifiable state transition caused by MPT and resist byzantine adversaries with normally only 2 transactions, which is definitely can be further optimized in practical scenarios.

The main features of Cloak include:

- CT/MPT-Enabled

, support the confidential transaction (CT) and Multi-Party Transaction (MPT), which take private functions parameters or states from different parties.

- Mixed Contract-Enabled

, support smart contracts with mixed public, confidential and multi-party transactions without violating any of their privacy policies.

- Easy to use

, help developers develop confidential smart contracts by only annotating the data owner in source code.

- Easy to develop

, provision a toolchain for developing, deploying confidential smart contracts, and transacting with them.

- Pluggable

, enable confidential smart contracts on EVM-enabled blockchain in a pluggable manner.

For more details, please visit our prototype description. Admittedly, it is far from prepared; very much looking forward to any feedback.

- Prototype: https://github.com/OxHainan/cloak-compiler
- Documentation: Welcome to Cloak's documentation! — Cloak 0.2.0 documentation
- Paper: [2106.13926] CLOAK: Enabling Confidential Smart Contract With Multi-Party Transactions