耐タンパー仮定に基づくオフライン・スマートコントラクト

Offline smart contract based on tamper resistance assumptions

松本彩花・ネットワーク分科会・大塚研究室

Blockchain has such disadvantages as high gas consumption during the execution and long latency to P2P transaction certification. To meet this challenge for blockchain as an execution architecture for smart-contracts, C. Raymond et al. (2019) proposed "Ekiden" as a technology to apply a trusted execution environment (TEE) to the blockchain. As Ekiden succeeds in reducing the calculation burden, it can process more smart-contracts in the same unit of time and resources. As a result, it can minimize the per-unit cost for the process and shorten latency. However, there is still a problem with introducing TEE, which this poster will discuss, as it takes a relatively long time to confirm that the internal state of TEE is securely stored in the blockchain as the blockchain fragment is used. This poster will assume a secure element (SE) of "Global Platform" specifications used in smartphones as tamper-resistant devices.

Outline

Problem : Disadvantages of smart-contract

Declining confidentiality and Performance.
⟹ Ekiden[1] proposed TEE & Block-chain technique.

Comparison items	Latency	Device	Improvement
Ekiden	**High**	**TEE**	**Confidentiality & Performance UP**
Proposal method	**Low**	**SE**	**Low Latency**

Comparison between proposal method and Ekiden.

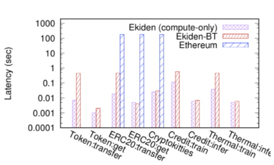


図 1: Latency[1]

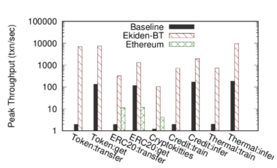
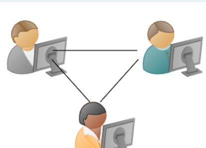


図 2: Throughput[1]

Problem

Block-chain

- As increase the amount of computation, gas-cost raises and be restricted feasible smart-contract.



Miners

Execute individually.

⟹ **Be restricted the feasible smart-contract.**

Goal

Goal of proposal method

- **Low latency**
  - Reduce the waiting time from settlement to contract execution (immediate execution).
- **Offline executable smart-contract**
  - Smart-contract temporarily runs offline and later connects online to expose the internal state.

Otimistic Proof of Publication

Definition (Publication)

An arbitrary string s is said to be publicized if and only if there exists a block satisfying the following:

$$^{\exists}B_i \in \mathcal{B}^{\lceil k} \quad s.t. \quad ^{\exists}\tau \in B_i \text{ and } s \in \tau$$

We write $s \in \mathcal{B}$ if s is publicized.

Work-flow

1. Player(Compute Node) gets a Token.
2. After SE check $s_{i-1} \in \mathbb{C}^{\lceil k}$, inputs Token and s_{i-1}.
3. SE decrypts s_{i-1} and updates the state. Verifier $\mathcal{V}$ sends s_i and Token.
4. Player registers the updated state with s_i as the state after the transition on the block-chain.
5. Send token to the next player.

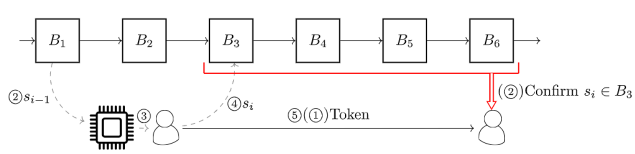
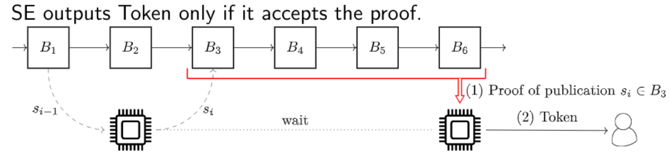


図 3: Optimistic Proof of Publication

Comparison for protocol

- **Proof of Publication[1]**
  SE checks to proof of publication $s_i \in B_3$.
  SE outputs Token only if it accepts the proof.



- **Optimistic Proof of Publication**
  SE output instantly Token.
  When receiver uses Token, receiver checks $s_i \in B_3$.