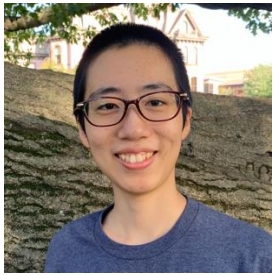
CrudiTEE:
A Stick-and-Carrot Approach to Building Trustworthy Cryptocurrency Wallets with TEEs

Authors



Lulu Zhou
Yale University

Zeyu Liu
Yale University

Fan Zhang
Yale University

Michael K. Reiter
Duke University

Background: blockchain and cryptocurrency

- Blockchain: a decentralized, distributed ledger that executes and records transactions.
- Cryptocurrency: digital currency built on blockchain, where the transaction is authorized by digital signatures.
- Smart Contract: self-executing code on blockchain.

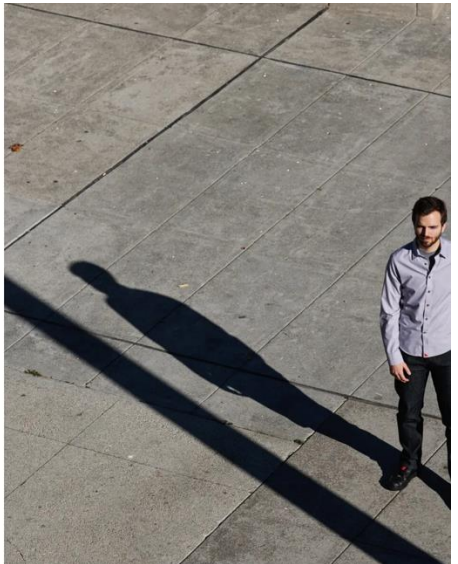
Cryptocurrency Wallet

- As cryptocurrencies gain popularity, more and more people use cryptographic signatures to authorize transactions.
- With inexperienced users often struggling with lost or leaked keys, a natural tendency is to outsource the task to specialized service providers.

The New York Times

Lost Passwords Lock Millionaires Out of Their Bitcoin Fortunes

Bitcoin owners are getting rich because the cryptocurrency has soared. But what happens when you can't tap that wealth because you forgot the password to your digital wallet?



TEE Wallet & Challenges

- However, custodial wallet solutions introduce extra trust assumptions to the service provider.
- Employing TEEs is a natural solution, but we still need to mitigate side channel attacks.

Insider vs. Outsider

- Insider Attacker: wallet provider who have physical access to the TEE

- Outsider Attacker: anyone who have access to less privileged side channels (e.g., remote time-based side channel).

Our Solution

- We develop a wallet key management solution with TEEs and OAuth.
- However, it is not possible to eliminate all the side channels of TEEs.
- Therefore, **economic incentives** are utilized to mitigate the side channels.

Insurance and Bounty

- The insurance ensures the **service provider** delivers service, or faces penalty.
- The bounty incentivizes the **outsider attackers** to submit the stolen key to the bounty, rather than using them or selling them.

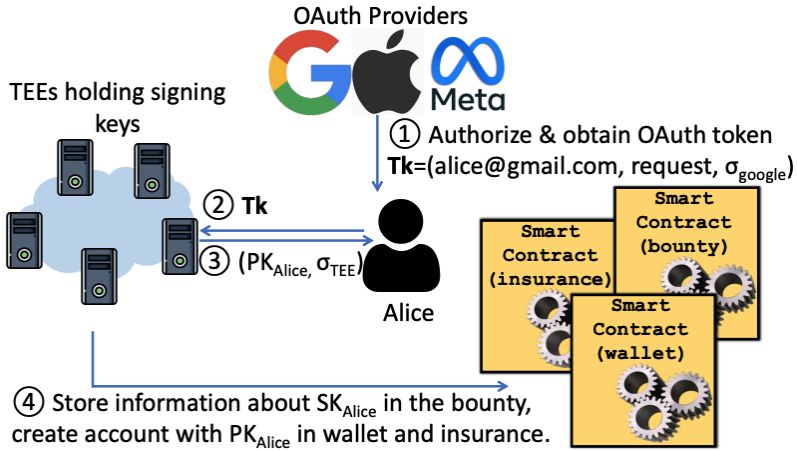
Accountable Wallet Key Management Service

CrudiTEE combines **TEEs** and an **authentication protocol** (OAuth), such that:

- Secret keys are generated within the TEEs and **never exported outside**. They can only be used by owners after authentication.
- The OAuth authentication protocol not only makes the authentication process **user-friendly**, but also make it **accountable**, since the OAuth token is a proof of authentication which the service provider cannot forge.

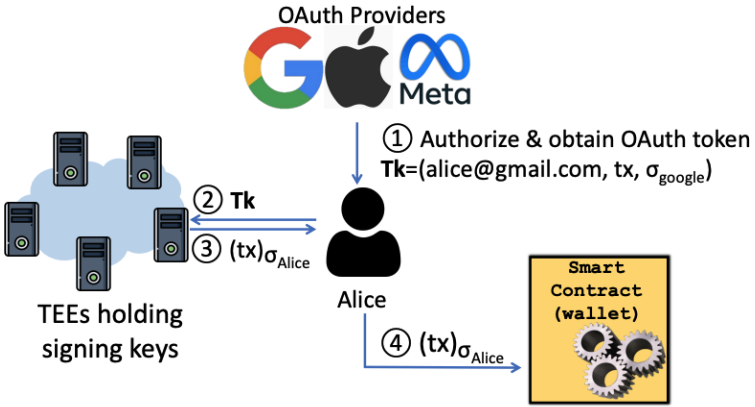
Registration

- The user chooses a set of OAuth providers they trust and requests tokens for registration.
- During registration, a key pair is generated for the user and associated with her OAuth ID(s). The secret key is secret-shared among multiple TEEs.
- A smart-contract wallet is deployed for the user on the blockchain.



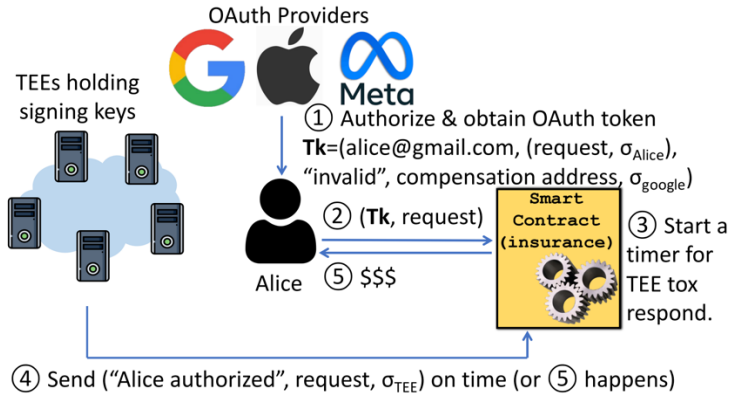
Transaction Signing

- The OAuth token is used as a method of authentication for transaction request, providing **accountability** for the authentication process.
- The OAuth token is stored in case of any dispute in the future.



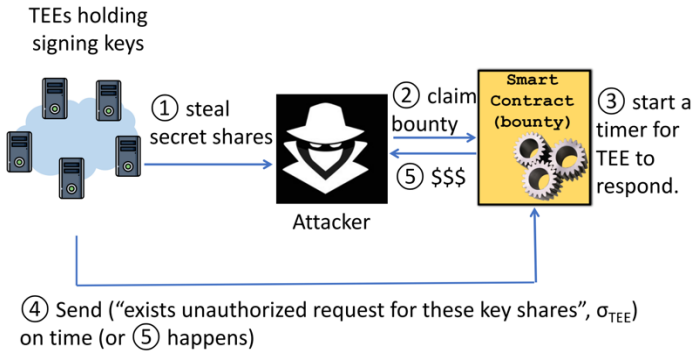
The Stick: Hold Service Provider Accountable

- Convert the service provider from a potential attacker to a **defender**.
- The **service provider** is a powerful attacker: many side channel attacks require root access to the host of TEEs, which the service provider can easily get.
- Automatically penalize the service provider if **unauthorized transactions are found.**
  - If the user finds an unauthorized transaction, they submit a insurance claim.
  - The service provider must look up the OAuth token for the corresponding transaction, let the TEE sign a message to prove the existence of the token, and submit the proof to the insurance smart contract. Otherwise the user is automatically compensated.



The Carrot: Make Remote Attackers Benign

- Enables “graceful degradation”: **detection** of TEE breaches before the entire system is breached.
- By **secret-sharing** the wallet key among TEEs, attackers must break multiple TEEs to steal a single key.
- The attacker has two options: steal the money or claim the bounty, but **cannot do both**.
- A bug bounty program with a carefully designed reward function **incentivizes attacker to flag side channels** rather than stealing the whole key.



Goals for reward function design

1. An attacker prefers submitting the key share(s) to bounty over selling them in the market / stealing the money .

2. An attacker submits the key share as soon as they get the first key share, instead of continuing the attack.

3. The defender’s cost is minimized.

f Score: Performance Metric for Reward Function

- Metric 1 (for goal 1): The probability that the attacker sells the key shares at the end of the game (p_e).
- Metric 2 (for goal 2): The average time between the attacker acquiring the first share and the termination of the game (t_h, normalized by the total period T).
- Metric 3 (for goal 3): The cost of the defender (c_d, normalized by the total value of the wallet v).
- We combine the 3 metrics with a weighted average and construct score f :

$$f = \alpha_1 \cdot p_e + \alpha_2 \cdot \frac{t_h}{T} + (1 - \alpha_1 - \alpha_2) \cdot \frac{c_d}{v}$$

Our proposed reward function

- The reward function $R(k,t)$ is a function of the number of key shares submitted to the bounty k and the time when they are submitted t.
- To achieve goal 1, we make $R(k) = V(N)^{\epsilon} \cdot V(k)^{1-\epsilon} + \eta > V(k)$ for all $k \in [1, N]$, since $\epsilon \in [0, 1]$ and $\eta > 0$.
- To achieve goal 2, we make the **extra reward** given to the attacker $g(k)$ **decay** with time.
- We leave the variable ϵ tunable, where we could compute an optimal one to minimized the f score.

$$g(k) \coloneqq V(N)^{\epsilon} \cdot V(k)^{1-\epsilon} + \eta - V(k)$$

$$R(k,t) \coloneqq V(N)^{\epsilon} \cdot V(k)^{1-\epsilon} + \eta - g(k) \cdot \frac{t}{T}$$

MDP to Model Attacker's Behavior

State Tuple (k, t, d):

- k: number of shares gained by the attacker.
- t: time slot.
- $d \in \{0, 1\}$: sub-step in each time slot for decision making.
  - At sub-step 0: the attacker chooses the number of TEEs to attack.
  - At sub-step 1: the attacker chooses to end the game by submitting to the bug bounty or to continue the game in the next time slot.

MDP to Model Attacker's Behavior

- At each step, attacking one TEE cost the attacker c_a, with a probability p_s of successfully breaching the key share inside the TEE.
- The **reward** of of the attacker is calculated when they turn in the key shares to the bounty or sell the key shares.

State × Action	State	Probability	Reward
$(k,t,0)\times$attack n TEEs	$(k+i,t,1)$	$Pr(\Delta k=i)$	$-n\cdot c_a$
$(k,t,1)\times$ wait $(t<T)$	$(k,t+1,0)$	1	0
$(k,t,1)\times$ wait $(t=T)$	termination	1	0
$(k,t,1)\times$ turn in	termination	1	$R(k,t)$
$(k,t,1)\times$ selling key	termination	1	$V(k)$

Evaluation of Reward Function

- We solve the MDP problems and use the results to **assess defense efficacy** of the reward function with f score.
- Here is a comparison of the f scores for different attackers and different functions (a smaller f score indicates better performance):

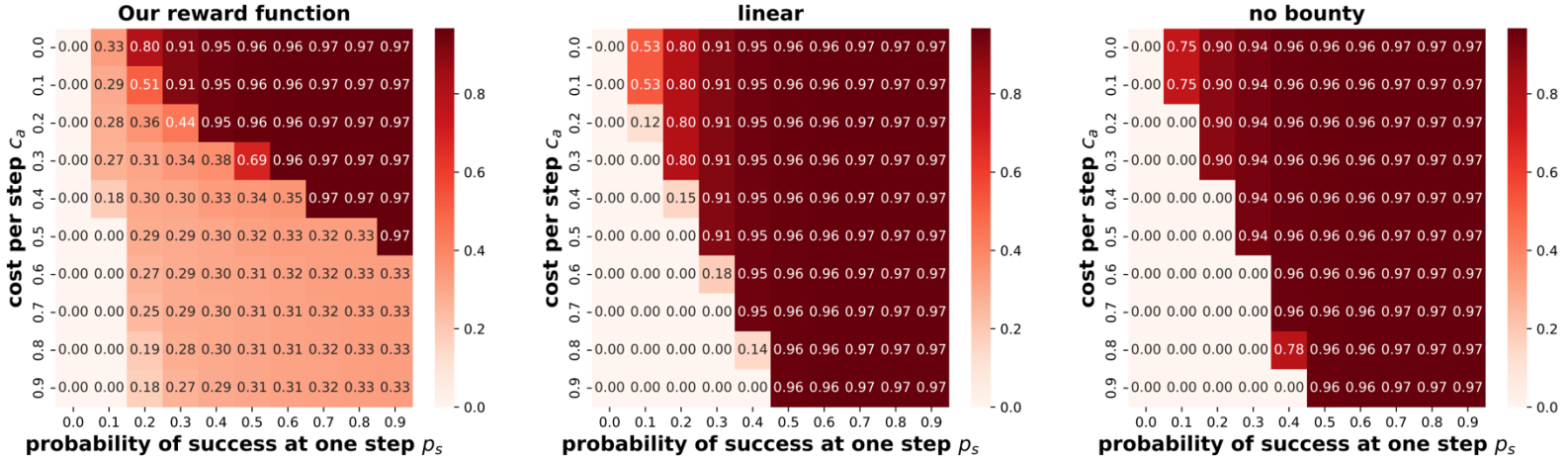


Figure 4 f score for different reward functions. $\alpha_{\text{cap}} = 0.8$. $\alpha_1 = \alpha_2 = 1/3$, $c_a = 0.4$, $p_s = 0.4$, $N = 3$, $\mathsf{v} = 6$. Optimal $\epsilon = 0.95$.

Summary

We leverage **economic incentives** to defend against side-channel attacks to cryptocurrency wallets, with insurance and bounty:

- The **insurance** automatically compensates the users for unauthorized transactions.
- The **bounty** has a well-designed reward function, whose effectiveness is evaluated by a model capturing the non-deterministic behavior of attacker.

I’m on the Job Market

- I will graduate with a Master’s degree in December 2024.
- 4 year of experience in blockchain research (Blockchain consensus, TEE, Mechanism design, ZK proof).
- Open to researcher / software engineer jobs

Thank you!

Contact: lulu.zhou@yale.edu

Paper

Code