# Scrappy
## SeCure Rate Assuring Protocol with PrivacY

Kosei Akama*1*†, Yoshimichi Nakatsuka*2,
Masaaki Sato*1*3, Kesuke Uehara*1

*1 Keio University, *2 ETH Zurich, *3 Tokai University

*† Corresponding author
akama[at]keio.jp

## Introduction

# Problem 1/2

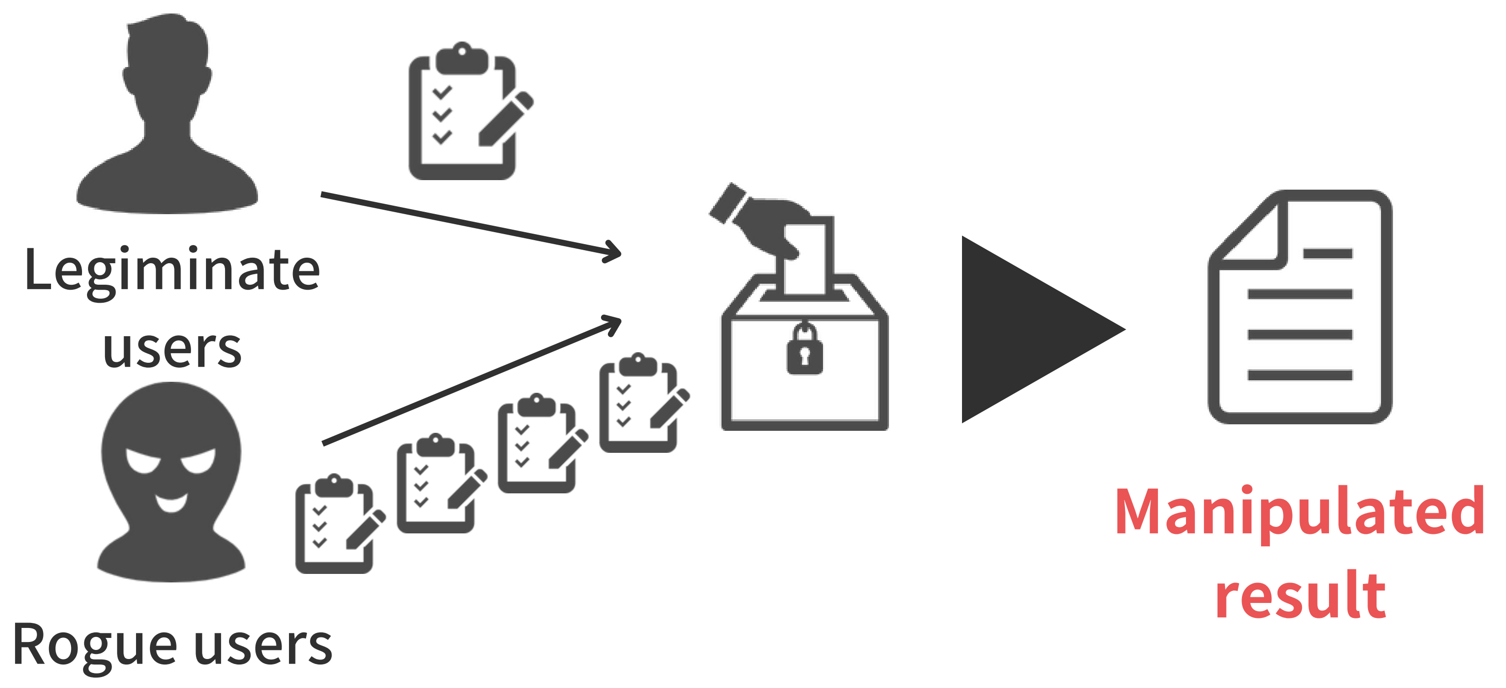
Abuses by access at a rate exceeding services expectations are the problem

→Need to limit the access rate, namely Rate-Limiting

## Example 1: Large number of submissions in questionaire

Malicious user submits many responses

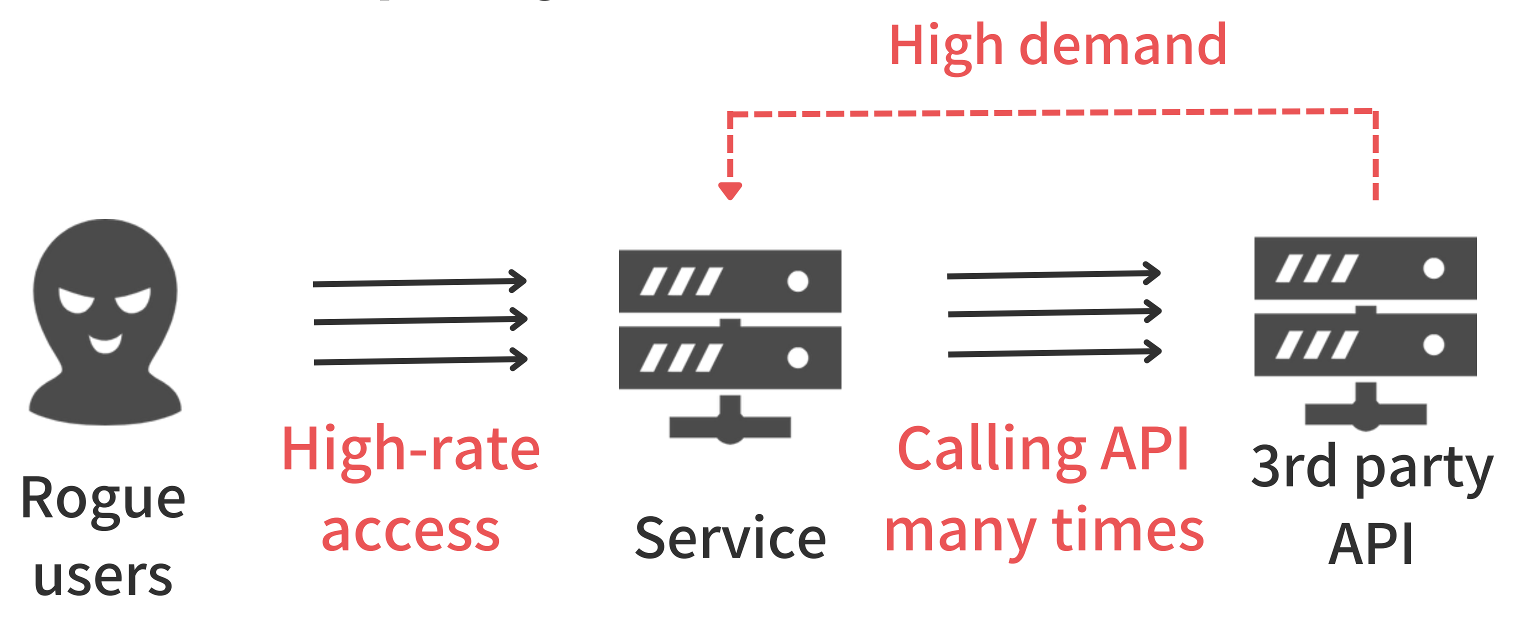
→The result will be manipulated



## Example 2: Excessive consumption of 3rd party APIs

Malicious user causes web services to call 3rd party API many times

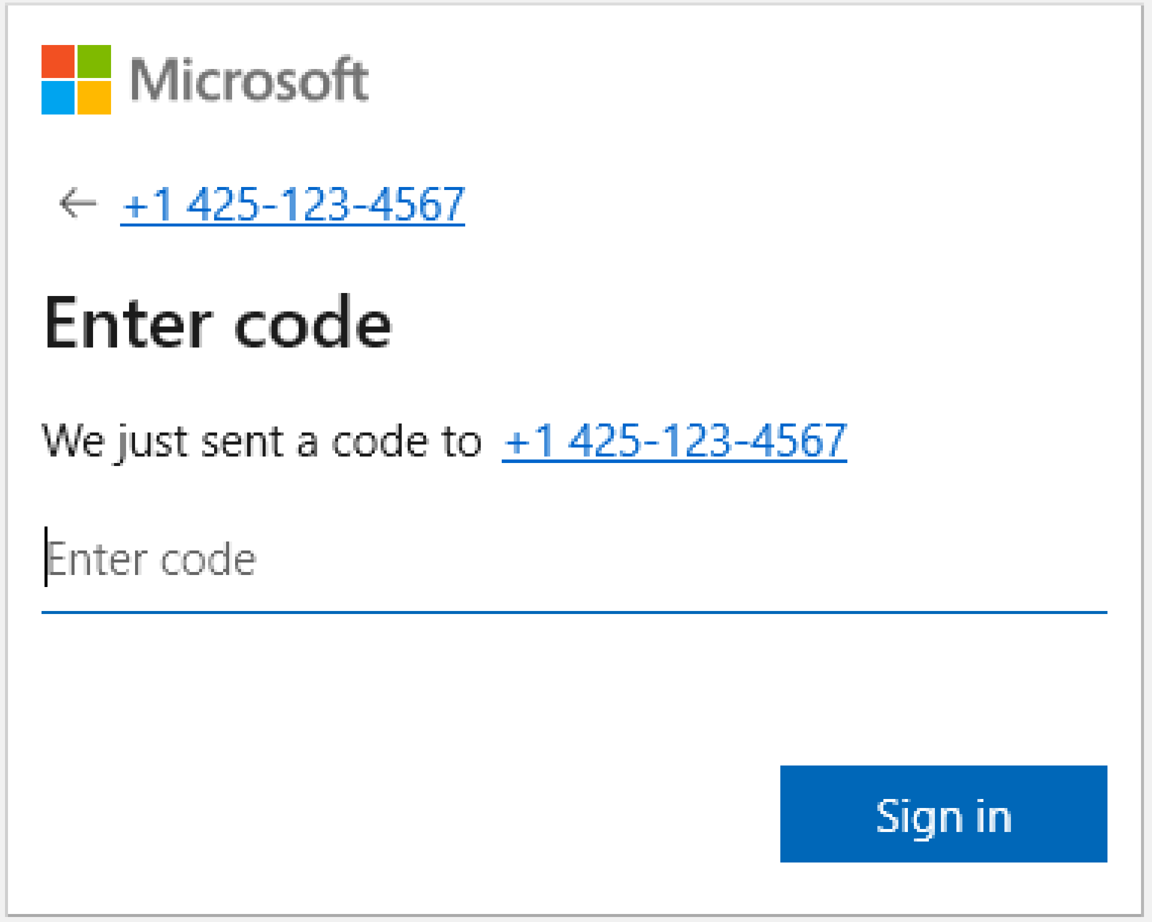
→Service will get an expensive bill by the 3rd party



# Problem 2/2

SMS authentication is the most straightforward approach to limit access rate
→Issue with users‘ privacy (i.e., tracking )

## Example of SMS authentication


Microsoft

← +1 425-123-4567

Enter code

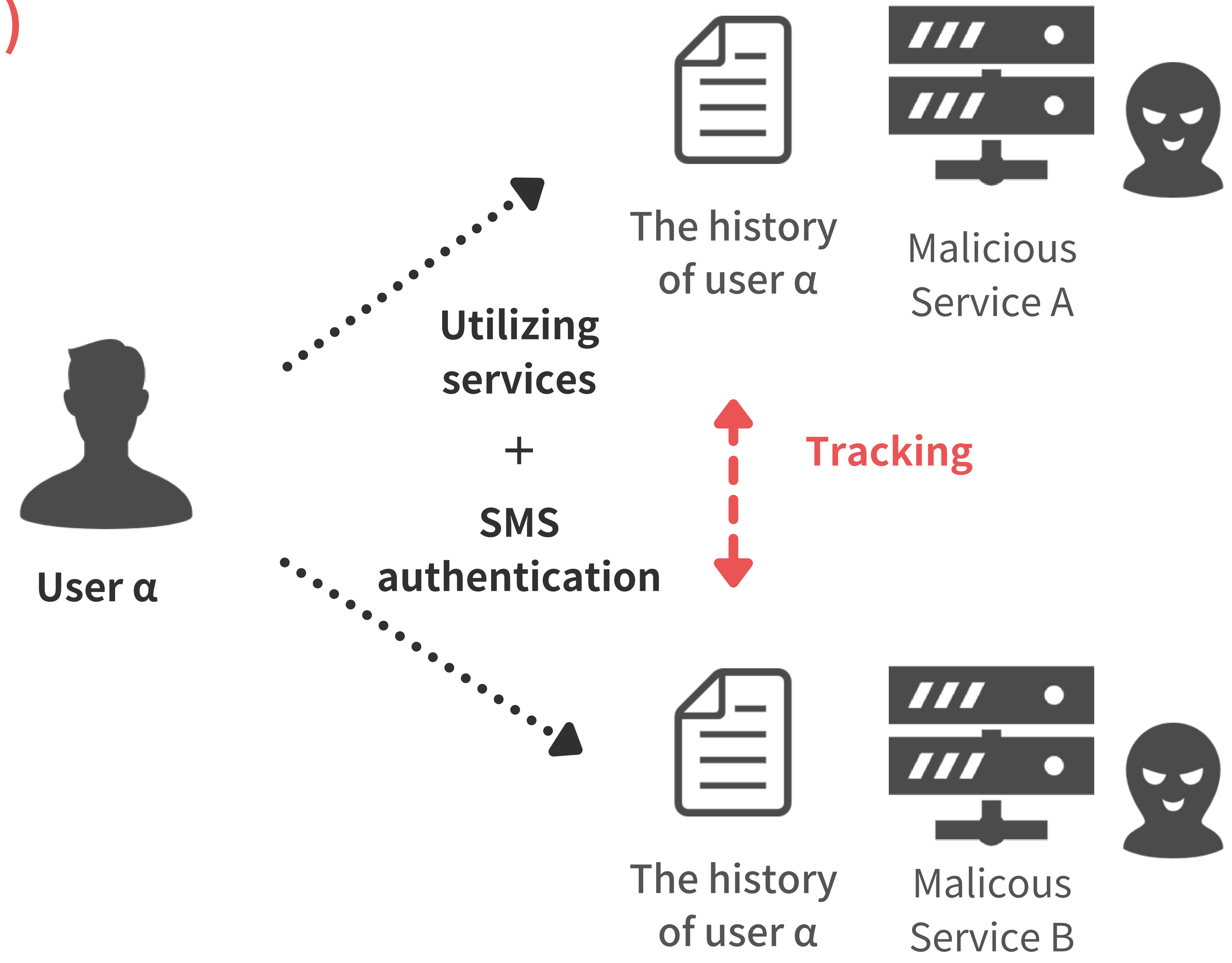
We just sent a code to +1 425-123-4567

Enter code

Sign in

https://learn.microsoft.com/en-us/entra/identity/authentication/howto-authentication-sms-signin

## Privacy concerns by the SMS authentication



## Related Work

- Many related work propose systems limiting access rate (called rate-limiter)
- **They have limitations regarding security or privacy**

| | SMS Authentication | CAPTCHA | CAP | CACTI | Privacy Pass | Opaak |
|---|---|---|---|---|---|---|
| **Private key storage** | - | - | SE | TEE | Undefined | **Files encrypted with a master password** |
| **Resource for uniqueness** | Phone number | - | - | Provisioning Key | Undefined | Phone number |
| **Resistance to timing correlation attacks** | - | - | Strong | Strong | **Weak** | Strong |
| **Rate-limiting depends on device security** | No | No | **Yes** | **Yes** | No | No |
| **Rate-limiting capability is vulnerable to evolution of AI** | No | **Yes** | No | No | No | No |

# Proposal：Scrappy

**Secure and Privacy-friendly rate-limiter**

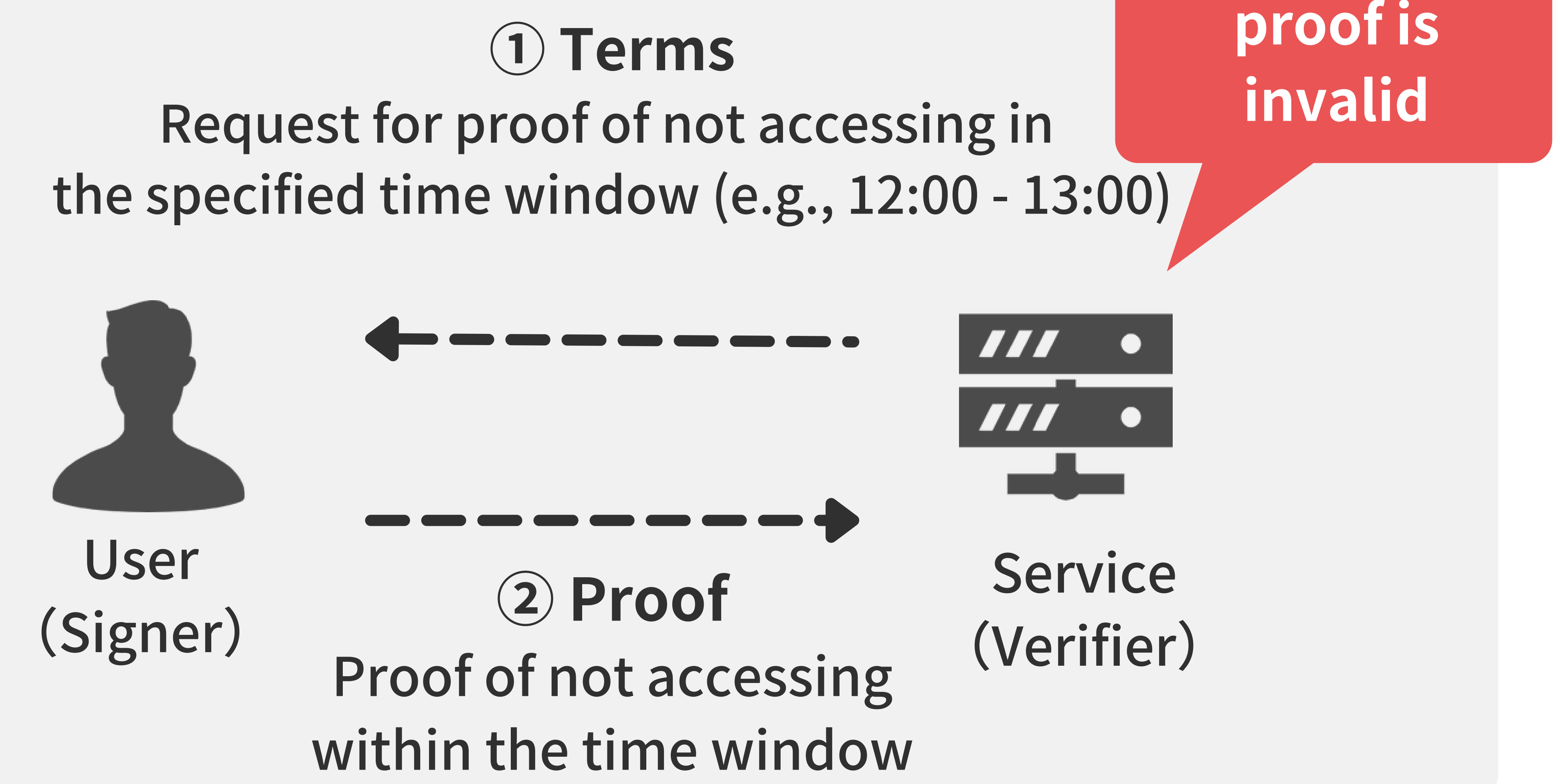
A cryptographic protocol blocking

(i) **multiple accesses** (ii) **from individual users** (iii) **within the same time window**

## Features

- Service cannot track users (i.e., privacy-friendly)
- Rate-limiting capability does not depend on users' device security
- The private key is stored in widely available secure hardware (TPM)

## Protocol Overview

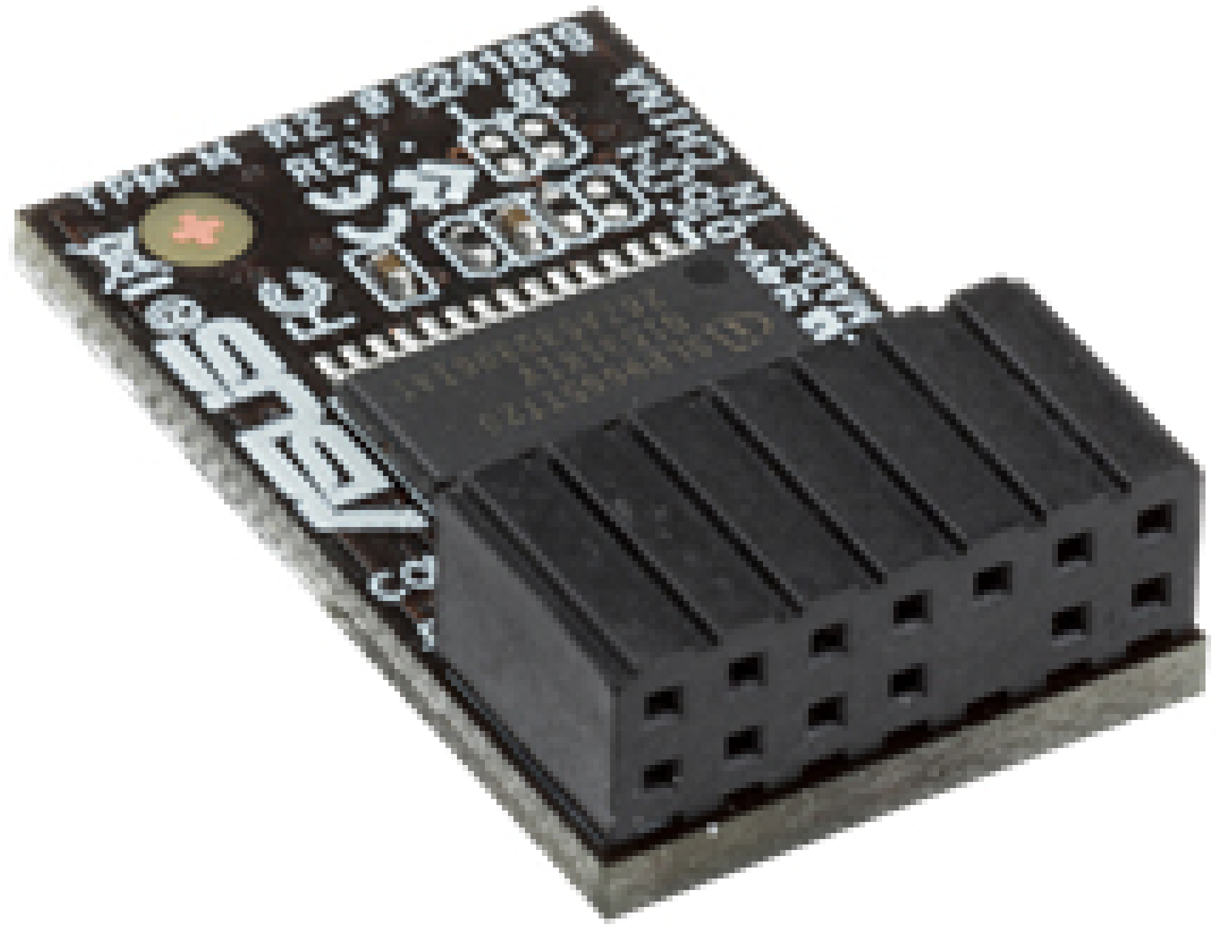

① Terms
Request for proof of not accessing in the specified time window (e.g., 12:00 - 13:00)
Block if the proof is invalid
User (Signer)
Service (Verifier)
② Proof
Proof of not accessing within the time window

## Background

## Background：TPM

**TPM(Trusted Platform Module) is a well-known security hardware chip.**

- **Secure storage of Key(: usk)**
  - Protected from side-channel attack
  - Supported scheme: RSA, ECDSA, and DAA
- **Attestation of the genuineness of the platform**
  - Manufacturer-installed unique secret: Endorsement Key (EK)



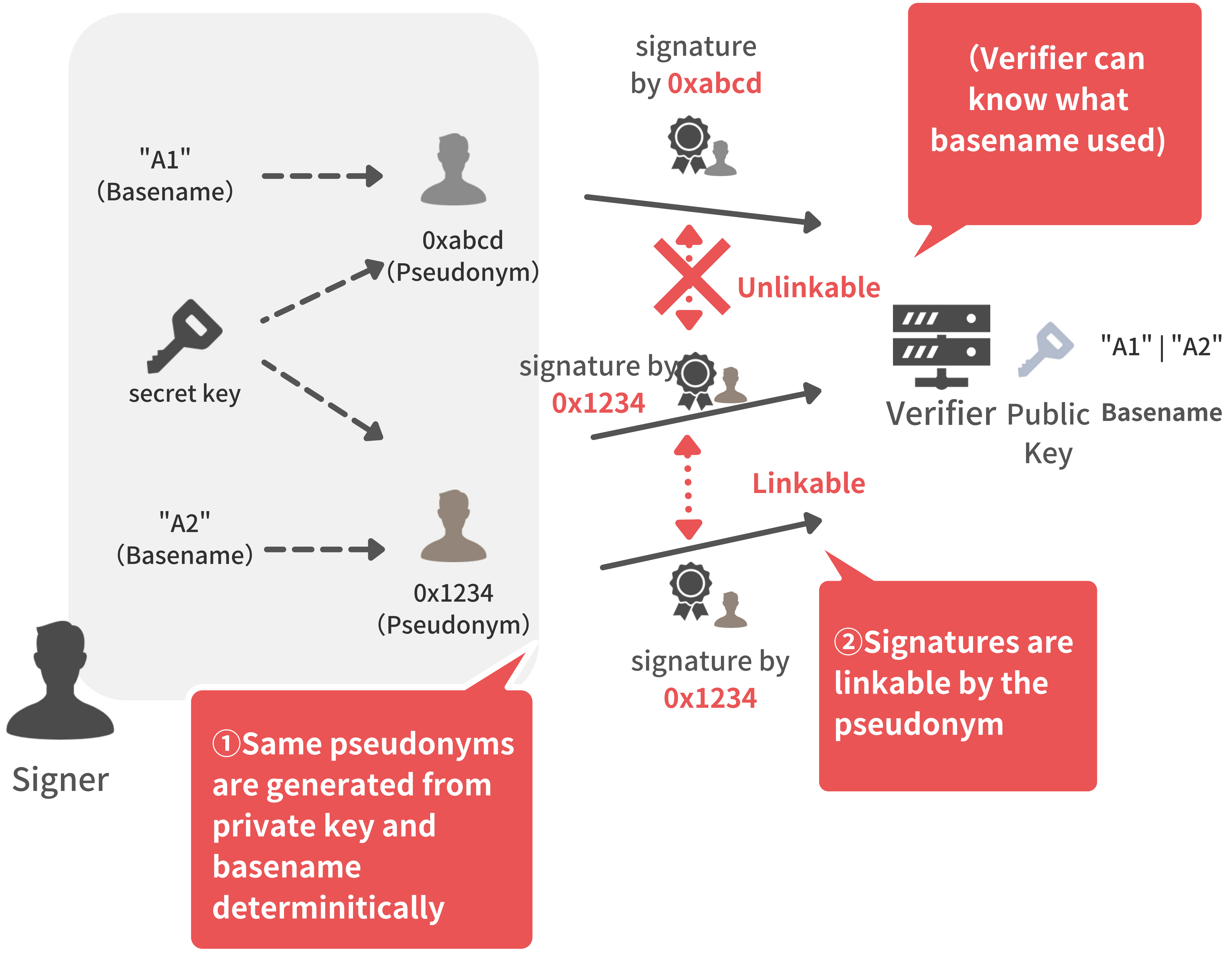
https://pc.watch.impress.co.jp/docs/topic/feature/1334277.html

## Background：DAA

**DAA(Direct Anonymous Attestation) is a privacy-friendly signature scheme.**

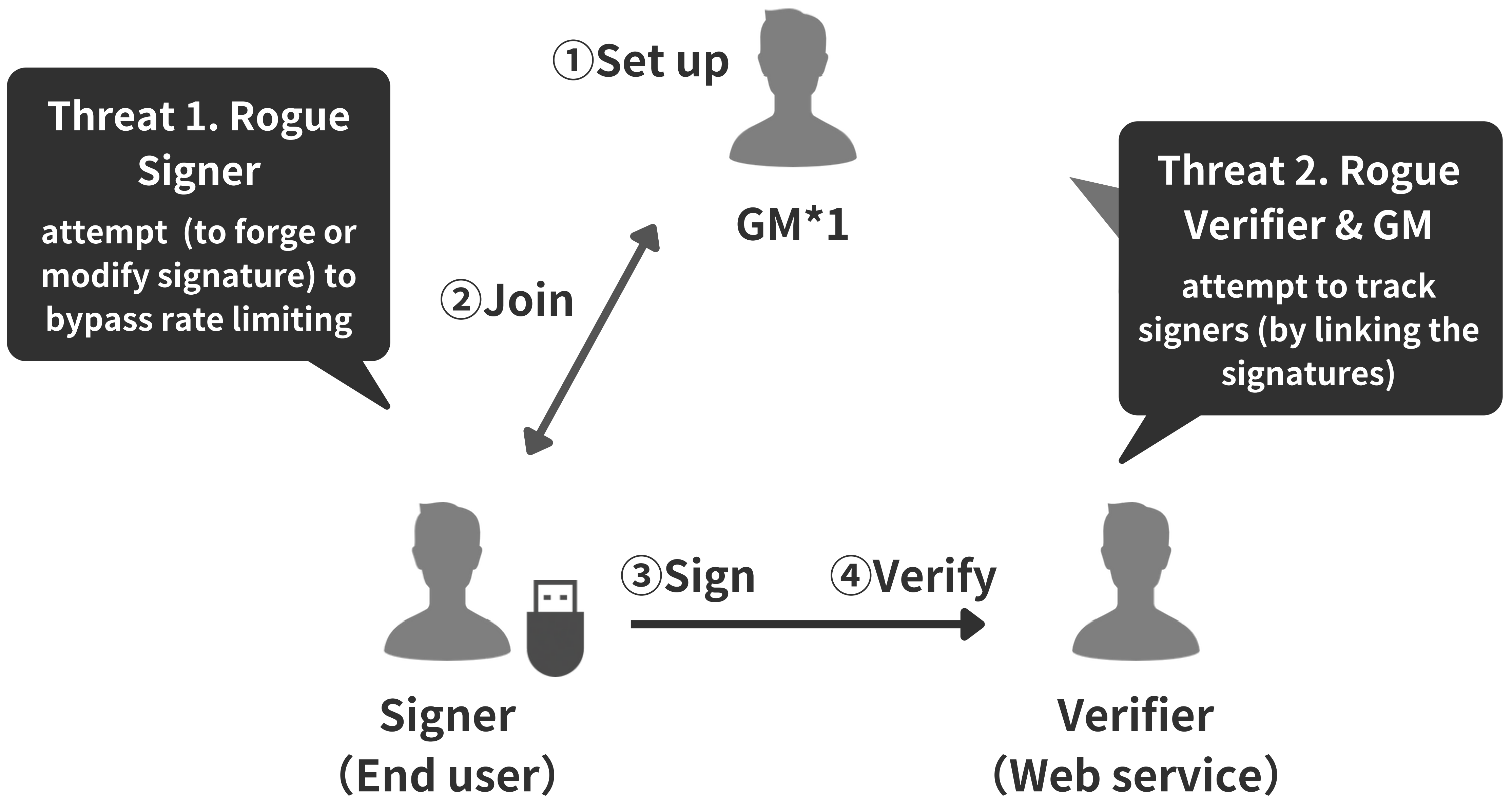
- Used for remote attestation of hardware chips
- Ensuring unforgeability and unlinkability
- DAA has a feature for signers to control their anonymity
  **→Verifier can link signatures using pseudonyms***

*A part of the signature



## Scrappy

# System & Threat Model



Threat 1. Rogue Signer
attempt (to forge or modify signature) to bypass rate limiting
①Set up
GM*1
②Join
Threat 2. Rogue Verifier & GM
attempt to track signers (by linking the signatures)
③Sign
④Verify
Signer (End user)
Verifier (Web service)

### Security Requirement

**Req a. Rate-Limit**
Signers cannot send requests that exceed the verifiers' threshold

**Req b. Unforgeability**
Signers cannot forge or modify signatures

**Req c. Unlinkability**
Verifier, even if colluding with GM, cannot track users

*1 GM: Group Manager, which is the authority to limit the number of users' credential

## Design: Challenge and Solution

### Challenge 1. Choosing the cryptographic protocol

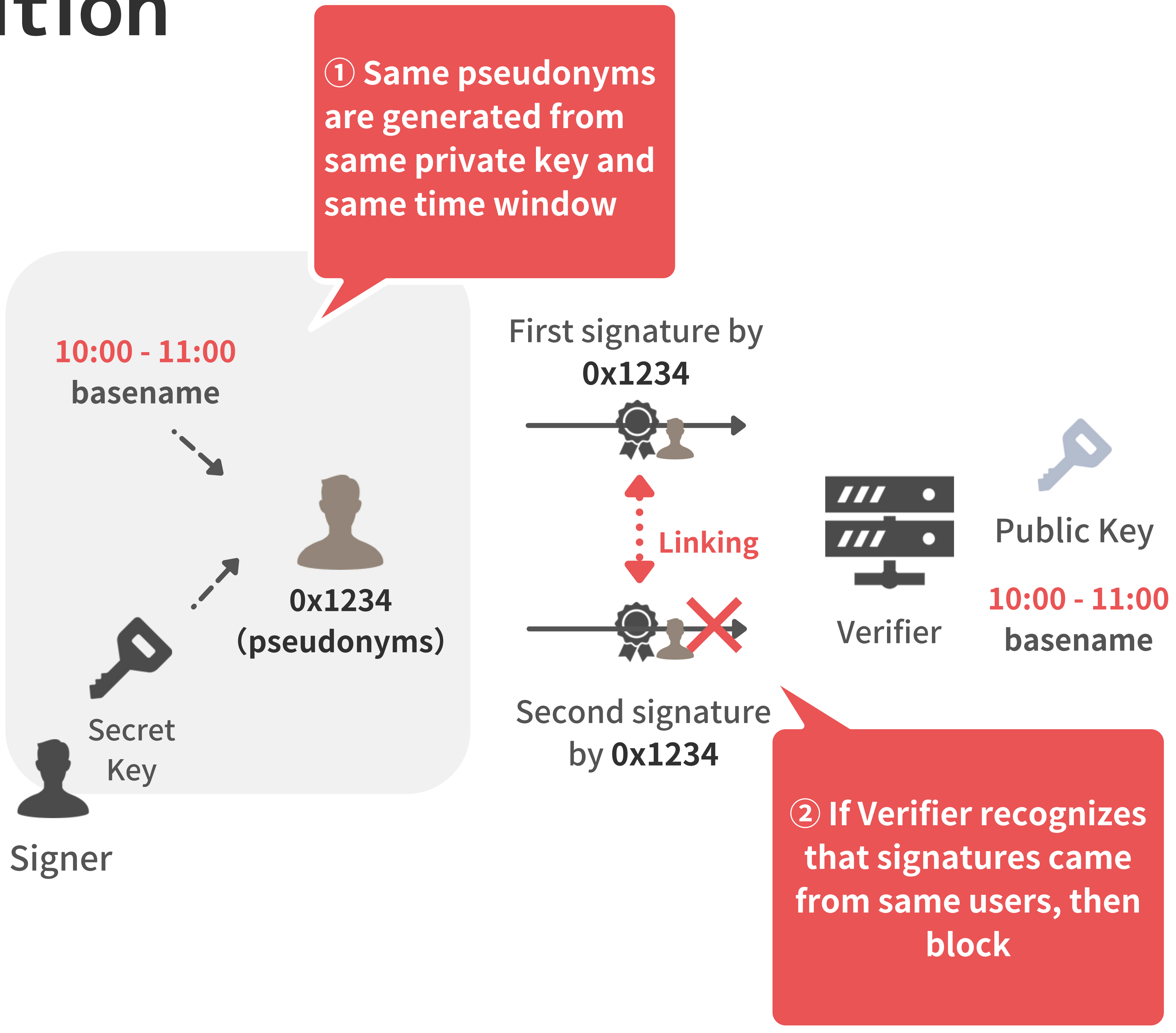
- The existing secure hardware does not support storing the key of Opaak(k-TAA).

→Solution. Based on DAA, a widely available protocol on secure hardware

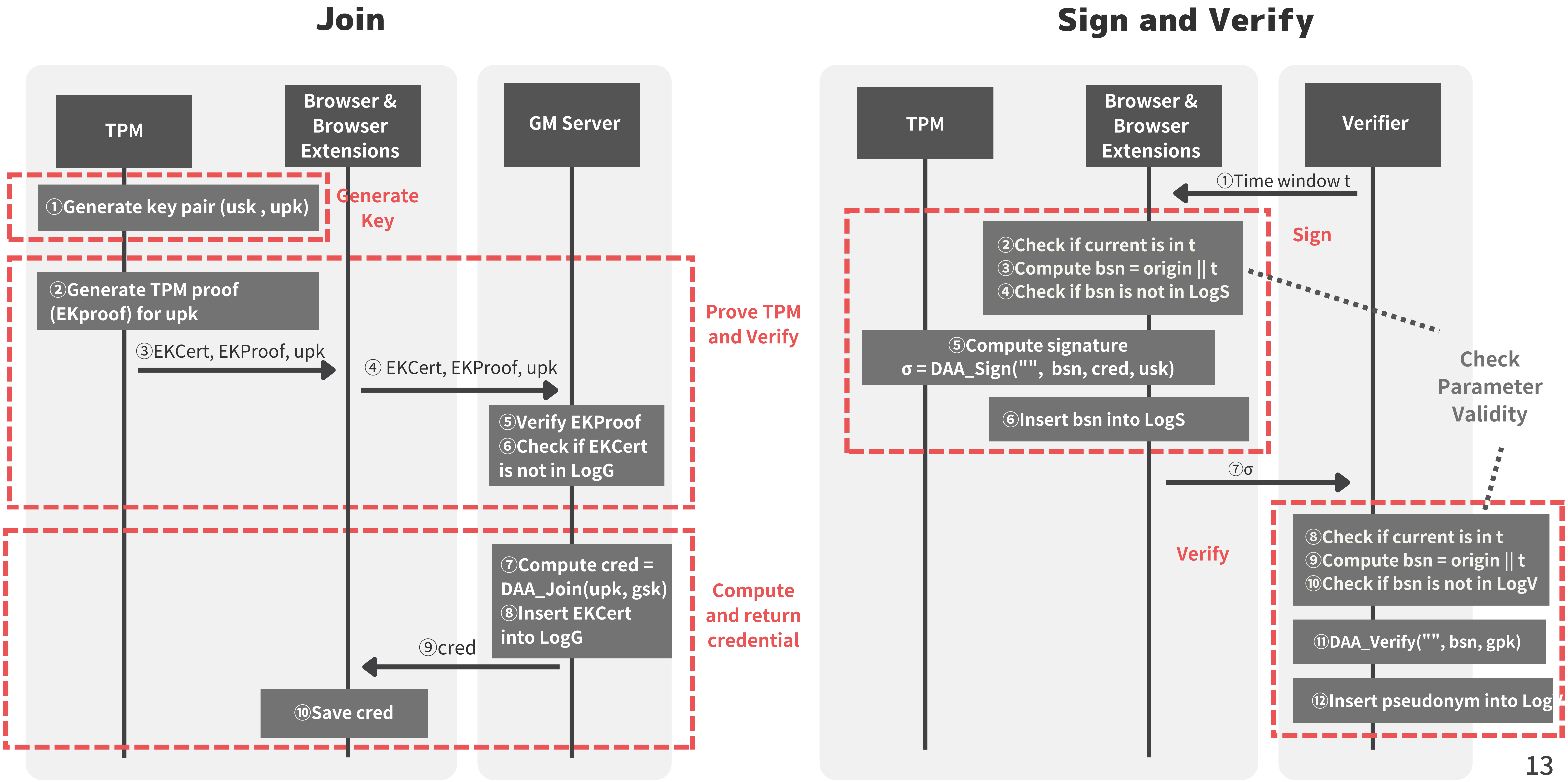
### Challenge 2. DAA has no rate-limit capability

→Solution. Generating pseudonyms from current time-windows

(Setting the time window to basename)


① Same pseudonyms are generated from same private key and same time window
10:00 - 11:00
basename
0x1234
(pseudonyms)
Secret Key
Signer
First signature by 0x1234
Linking
Second signature by 0x1234
Verifier
Public Key
10:00 - 11:00
basename
② If Verifier recognizes that signatures came from same users, then block

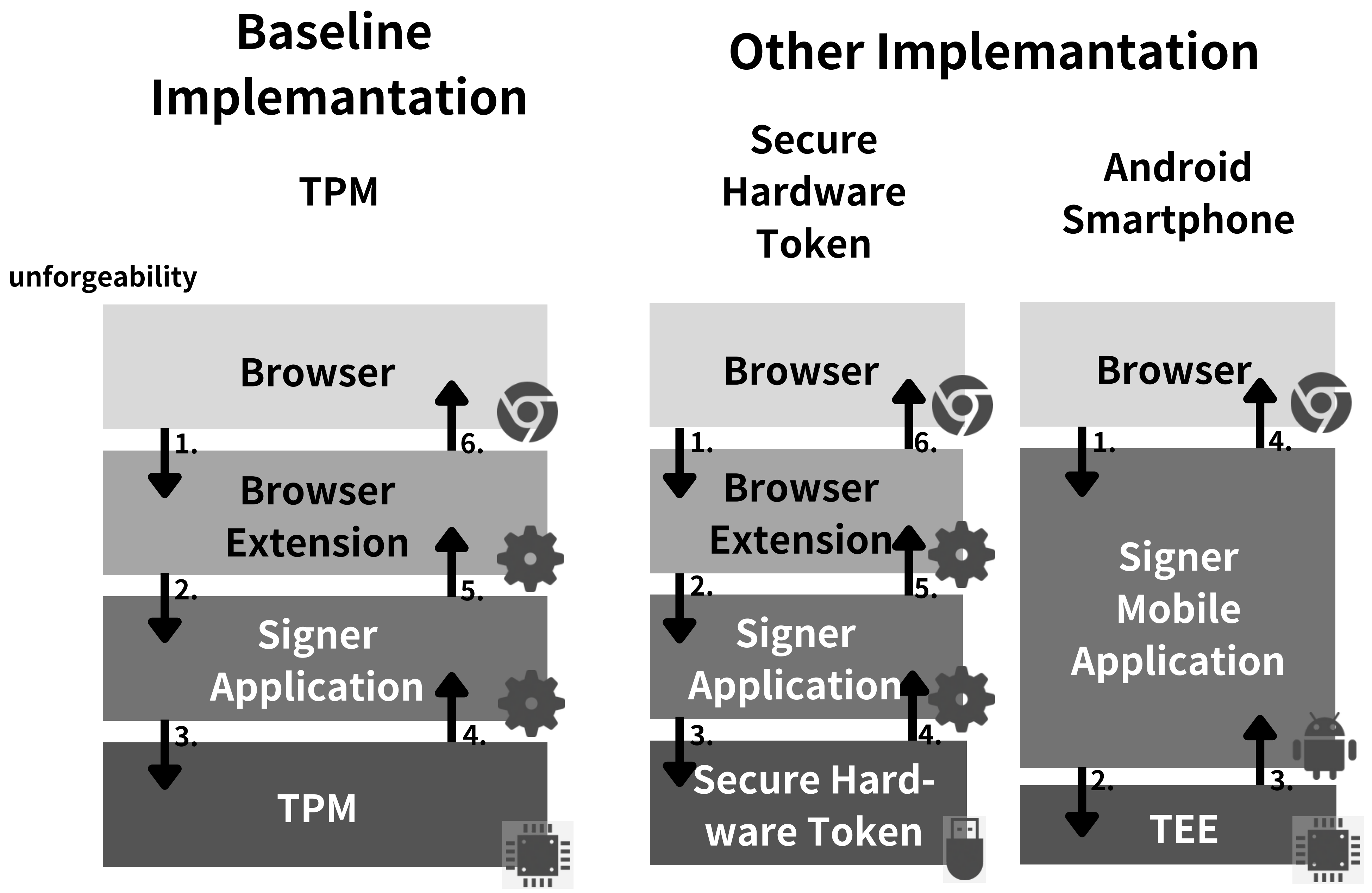
## Protocol


Join
TPM
Browser & Browser Extensions
GM Server
①Generate key pair (usk , upk)
Generate Key
②Generate TPM proof (EKproof) for upk
③EKCert, EKProof, upk
④ EKCert, EKProof, upk
⑤Verify EKProof
⑥Check if EKCert is not in LogG
Prove TPM and Verify
⑦Compute cred = DAA_Join(upk, gsk)
⑧Insert EKCert into LogG
⑨cred
⑩Save cred
Compute and return credential
Sign and Verify
TPM
Browser & Browser Extensions
Verifier
①Time window t
②Check if current is in t
③Compute bsn = origin || t
④Check if bsn is not in LogS
Sign
⑤Compute signature σ = DAA_Sign("", bsn, cred, usk)
⑥Insert bsn into LogS
Check Parameter Validity
⑦σ
Verify
⑧Check if current is in t
⑨Compute bsn = origin || t
⑩Check if bsn is not in LogV
⑪DAA_Verify("", bsn, gpk)
⑫Insert pseudonym into Log

# Implementation

- Mainly implemented on PC with TPM
  →Checked Scrappy functionality

- We also implemented using Secure Hardware Token and Android Smartphone
  →Checked that Scrappy works regardless the device



## Evaluation & Analysis

## Performance Evaluation: Baseline Latency

Scrappy is efficient enough to be deployed to the real world.

We measure the latency of signing and verifying in Scrappy and compare it with related work
→**The latency is smaller than related work**

| Related Work | Signing Latency[ms] | Verifying Latency[ms] |
|---|---|---|
| **CACTI[2]** | 211.9 | 27.3 |
| **Privacy Pass** (N tokens)[6] | 341.48 + 180.87 × N | 57.8 |
| **Opaak[3]** | 2550 (combined measurement) | |
| **Scrappy(TPM)** | **243.16** | **84.1** |
| **Scrappy(Secure Hardware Token)** | **2771** | |
| **Scrappy(Smart Phone)** | **26.4** | |

## Security Analysis - Compromised keys

We analyzed the security of Scrappy
→**We showed that Scrappy fulfills the requirements under certain assumptions.**

**Threat 1. Compromised usk**

- The signing rate is the same as that of an eligible user
- Although adversaries can link users by compromised usk, extracting key from TPM is extremely challenging

**→Impossible to violate rate-limit ability and extremely hard to link users**

**Threat 2. Compromised EK**

- The signing rate is the same as that of an eligible user
- Leaked EK does not violate the victim's privacy since the usk cannot be linked to EK

**→Either violating rate-limit ability or unlinkability is impossible**

**Threat 3. Compromised usk & EK**

**→The rate-limiting property remains intact and extremely hard to link users** (owing to above reasons).

**etc…**

## Conclusion

## Conclusion

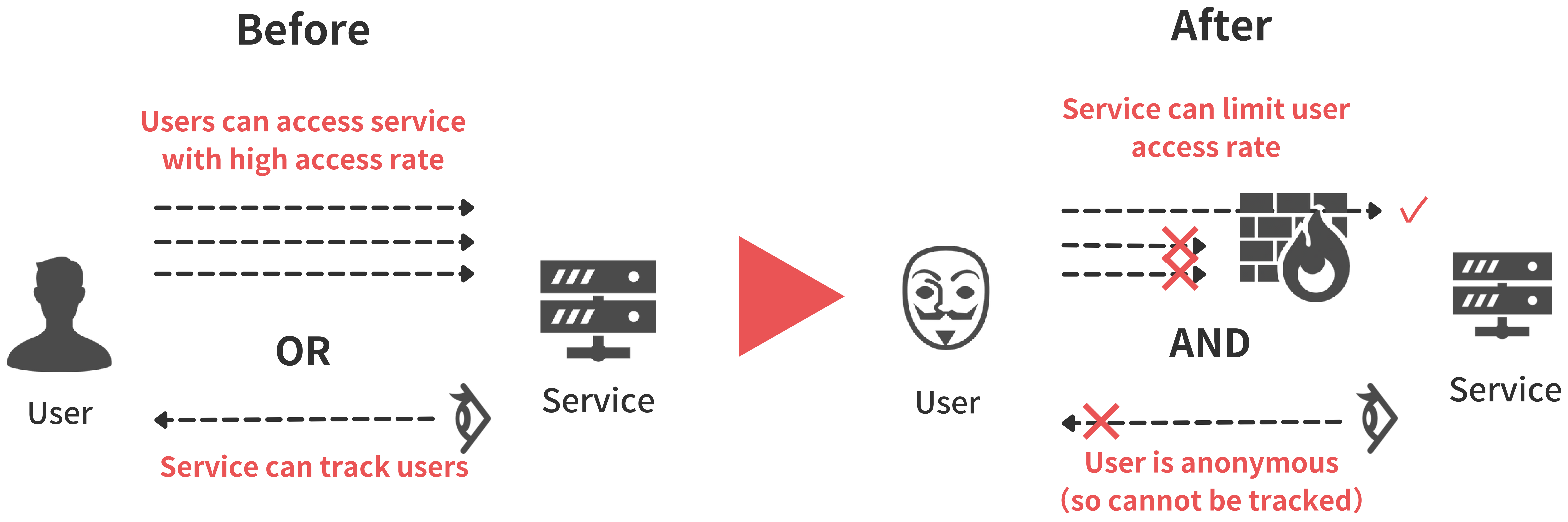
Contact: akama[at]keio.jp

- Scrappy: Privacy-preserving rate limiting protocol combining hardware security devices & DAA
- Baseline implementation requires no changes to the hardware or specification
- Unforgeability & Unlinkability features hold if tamper-resistant feature holds
- The rate-limiting feature holds no matter what
- Minimal latency
    - Proof generation: 243 ms
    - Proof verification: 84 ms

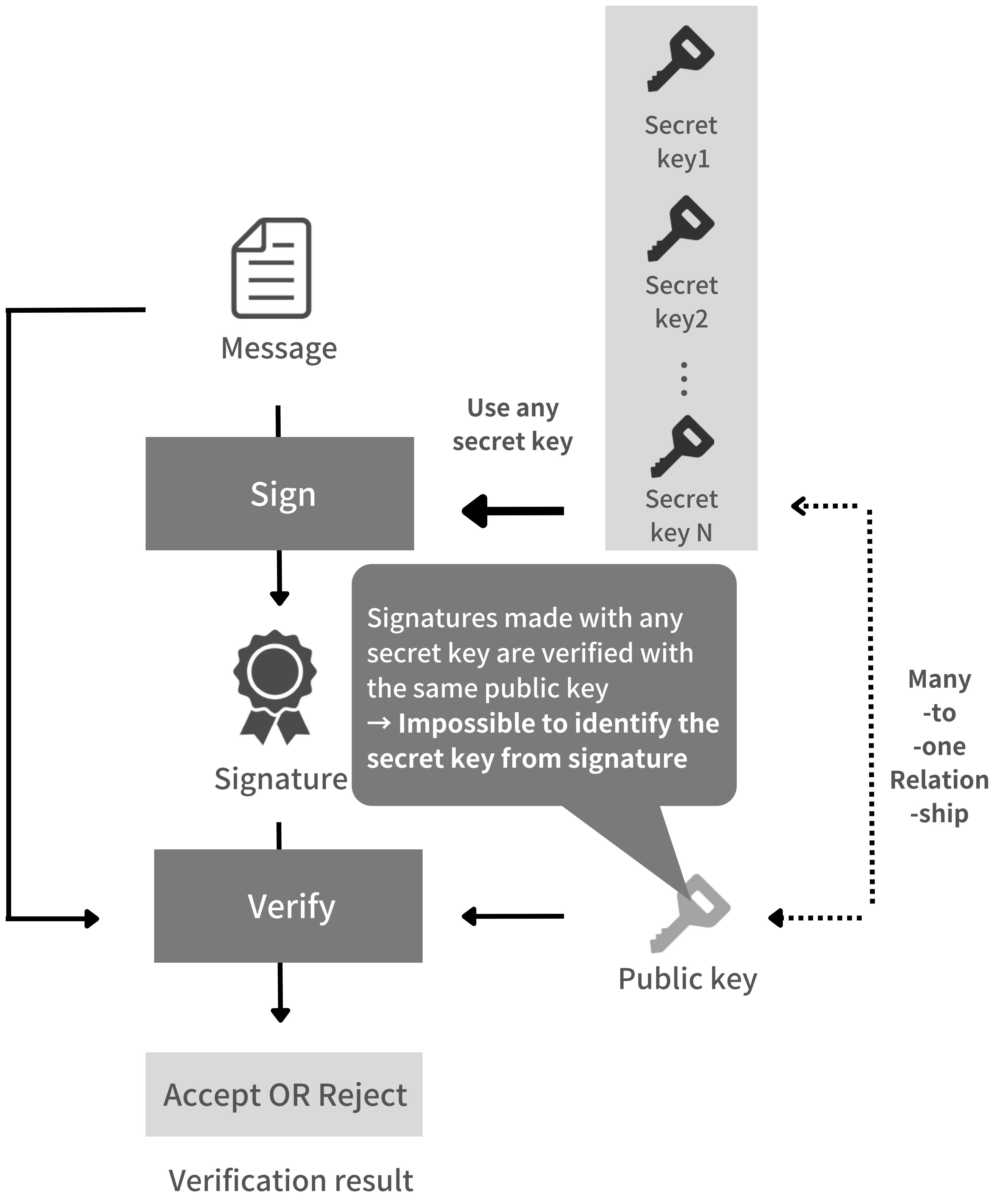
## Appendix

## Purpose

**To limit user access rate while protecting user privacy (specifically the anonymity).**


Before
Users can access service with high access rate
OR
Service can track users
User
Service
After
Service can limit user access rate
✓
AND
User is anonymous (so cannot be tracked)
User
Service

# How DAA holds Anonymity


Secret key1
Secret key2
Secret key N
Message
Use any secret key
Sign
Signature
Signatures made with any secret key are verified with the same public key
→ **Impossible to identify the secret key from signature**
Many -to -one Relation -ship
Verify
Public key
Accept OR Reject
Verification result

https://speakerdeck.com/akakou/privacy-enhanced-revocation-and-view-of-applying-to-web

# Performance Evaluation

## Latency（Corner case)

On the corner case, Scrappy also works with short latency

| | Latency of signing[ms] | Latency of Verifying[ms] |
|---|---|---|
| A case the signer having meny logs（1000) | 243.23 (236 + 7.23) | 84.1 (73.7 + 10.4) |
| A case where the revocation list is large（50) | 243.16 (236 + 7.16) | 151.4 (141 + 10.4) |

## Storage Consumption

The logs of Scrappy are not large.
→**Scrappy does not pressure the storage**

| | Logs total size |
|---|---|
| A case the signer having meny logs（1000) | 94.2 KB |
| A case the verifier having meny logs（100,000) | 6.64 MB |

## Bandwidth

The signature size is 261B.
→ **Scrappy does not pressure the bandwidth**

## Comparsion of each implemantations

| Device | **Baseline (TPM)** | Secure Hardware Token | Android Smartphone |
|---|---|---|---|
| **Private key storage** | **TPM** | Hardware Token | File encrypted using TEE |
| **Unique Resource** | **Endorsement Key** | Preinstalled secret key | Serial number, IMEI, MEID |
| **Method of Proving Unique Resource** | **TPM Attestation[8]** | Challenge & Response Authentication | Android ID Attestation |
| **Rate-limiting Depends on Device Security** | **No** | No | Yes |

# The Parameters of Scrappy Protocol

## Common

| Notation | Description |
|---|---|
| **gsk** | GM’s secret key |
| **gpk** | GM’s public key (generated from gsk) |
| **usk** | Signer’s secret key |
| **cred** | Signers credential |
| **upk** | Signers public key |

## Join

| Notation | Description |
|---|---|
| **EKCert** | Unique certificate for each TPM by vendor |
| **EKProof** | Proof of EKCert |
| **LogG** | List of EKCert |

## Sigin / Verify

| Notion | Description |
|---|---|
| **bsn** | basename |
| **t** | Time window e.g., 10:00 - 11:00 |
| **σ** | Signature |
| **origin** | Origin of site（e.g., www.example.com) |
| **LogS** | Signer’s log of bsn |
| **LogV** | Verifier’s log of psuedonym |
| **psuedonym** | Psuedonym. A part of signatures |