How to use your brain for cryptography without trustworthy machines

Wakaha Ogata[1], Toi Tomita[2], Kenta Takahashi[3], Masakatsu Nishigaki[4]

[1] Institute of Science Tokyo
[2] Yokohama National University
[3] Hitachi Ltd.
[4] Shizuoka University

Abstract. In this work, we study cryptosystems that can be executed securely without fully trusting all machines, but only trusting the *user's brain*. This paper focuses on signature scheme. We first introduce a new concept called "server-aided in-brain signature," which is a cryptographic protocol between a human brain and multiple servers to sign a message securely even if the user's device and servers are not completely trusted. Second, we propose a concrete scheme that is secure against mobile hackers in servers and malware infecting user's devices.

Keywords: Digital signature · untrusted machine · malware · mobile hacker.

1 Introduction

1.1 Motivation

Most traditional cryptosystems are designed to be secure on the assumption that secret information such as decryption keys and signing keys are kept secret against attackers, and the machine executing the algorithm is trustworthy. In reality, however, the security is not always guaranteed because the machine may be infected with malware that leaks secret information and computes in different ways than intended algorithm. Of course, if the user makes sure that the machine is not infected with malware before executing the algorithm, such problems will not occur. But it is a hard task for individual users to thoroughly check for malware infection on their devices.

One approach to address such problem is to use TPM and TEE. The device security of TPM and TEE guarantees the secrecy of secret information and the correct execution of cryptographic process. However, building a secure system and attacking it are always a cat-and-mouse game. Even highly protected systems may be compromised by extremely sophisticated malware or advanced attacks. Therefore, it is important to design a system whose security does not rely on the security of a particular device or machine.

Secure multiparty computation (MPC) and threshold cryptography are powerful tools to realize such a system, because they allow algorithms to be securely

and jointly executed by independent entities who do not trust each other. However, by using only MPC and threshold cryptography, we cannot solve an important issue; how to reflect human intention. For example, threshold signature can be used to generate signatures in the distributed manner, but how multiple signers agree to initiate the signing process is out of its scope. That is, it has no mechanism to reflect the signing intention of a (legitimate) human user.

The goal of this work is to develop a system in which only the legitimate human user can initiate the cryptographic process, and its security is guaranteed without trusting any particular device or machine.

Now consider an approach that simply combines an existing password-based user authentication scheme and an MPC; when a user wants to perform cryptographic process, first the user performs password-based user authentication using their password memorized in their brain, and if it passed, servers execute the MPC. This approach cannot achieve our goal, because in a password-based authentication, a user needs to type in their password on their terminal device and the device executes a protocol. This means that malware on the device can learn the password and may abuse it!

1.2 Our Contributions

In this paper, we focus on digital signature among various cryptographic processes. We summarize our contributions in the following.

Formal Definition of Server-Aided In-Brain Signature. As the first contribution, we introduce a concept called "server-aided in-brain signature," in-brain sig for short. Roughly speaking, an in-brain sig scheme consists of MPCs that are performed among a user and several servers to generate a public key and signatures. Unlike regular MPCs, which are performed by only machines, MPCs in in-brain sig allow a human user to participate, i.e, instead of user's device, a human user (or a human brain) performs the protocol with servers. Therefore, all calculations a user needs should be simple enough to be done in human brain (or only using a pencil and paper). A password memorized in user's brain is used to reflect human intention, but it does not have to be typed in any user's devices. This is possible because the user themselves participate in the MPCs. Therefore, in-brain sig is one of the solutions to achieve our goal.

As attackers against in-brain sig, we consider malware that infects a user's device and a hacker controlling a server, and they may behave actively, i.e., not follow protocols. We also consider *mobile hackers*, where a hacker corrupts several servers at different times. To ensure the security against active malware in a user device, we require each user to use two devices. See Sec. 3 for details.

Constructing an In-Brain Signature Scheme. The second contribution is proposing a concrete in-brain sig scheme based on threshold (EC)DSA. The overview of our in-brain sig scheme is as follows.

- Servers share a secret information (called master secret) in advance.

- A signing key for a user is determined by the user's password and the master secret using a pseudo-random function (PRF). The human user and servers jointly compute it by running an MPC. Further, servers jointly compute the corresponding public key using the threshold key generation protocol of (EC)DSA.
- In case servers are hacked, each server does not keep the user's signing-key share in its storage.
- When signing is requested, the user and servers re-generate the signing key and the corresponding public key. The password-based authentication is done by comparing the re-generated public key and the original one. Only after the successful authentication, servers perform the threshold signing protocol of (EC)DSA to compute a signature.

Our MPCs are carefully designed to minimize the calculation done by a human user. The security against malware that controls one of user devices and against a mobile hacker that controls predetermined number of servers, e.g., one-out-of three servers, is ensured through the use of secure MPCs and threshold signature scheme. By using a PRF to derive a signing key, offline dictionary attacks on users' passwords is prevented. Therefore, passwords only need to be long enough to resist online dictionary attacks.

1.3 Related Works

There are many proposals to solve the shortcomings of password-based user authentication mentioned above.

Human identification schemes and leakage-resilient password systems, first introduced in [12], are challenge-response authentication protocols that allow human users to compute responses in their brains, and passwords are never leaked to the user's device, unlike typical password-based schemes. However, these schemes are symmetric systems, that is, the server that authenticates the user verifies the authenticity based on the password shared with the user. Therefore, once the server is hacked, the password leaks.

Boldyreva et al. [2] proposed a system that allows a human user to be authenticated based on a password and exchange a session key. Similar to human identification schemes, any devices between the user and the authentication server cannot learn the password but the server needs to share the password to authenticate the user. So, it is vulnerable to server hacking.

Many systems that utilize human brainwave (e.g., brainwave communication, brainprint, passthoughts) have also been proposed. They, however, need special devices to measure human brainwave, and the system security relies on the device security as in the system with TEEs.

The survey by Halunen and Latvala [9] introduced many systems in which human's intention can be reflected into the systems. However, we cannot use them, because in these systems, user's device is assumed not to be infected, i.e., the system rely on a security of a single device.

2 Preliminaries

In this paper, $\lambda \in \mathbb{N}$ denotes the security parameter, $\mathbb{F}_q$ denotes the field with prime order q. For a non-negative function $f : \mathbb{N} \to \mathbb{R}$, we say that $f(x)$ is negligible in x (or simply f is negligible) if for any polynomial function p, there exists $n_0 \in \mathbb{N}$ s.t. $f(x) \leq 1/p(x)$ holds for all $x \geq n_0$. Let negl be a negligible function.

2.1 Secret Sharing and Multi-party Computation

Secret sharing schemes, especially threshold schemes, introduced by Shamir [15] and Blakley [1], is a basic tool to realize secure distributed systems. A secret sharing scheme[5], a secret value s is divided into n pieces, called "shares," each one is given to a "shareholder." From $t+1$ shares, the original secret can be uniquely reconstructed, where t is the predetermined threshold, while t or less shares give no information about s. Following many literature, we denote i-th share of s by $[s]_i$, and the list of n shares by $[s] = ([s]_1, \ldots, [s]_n)$. $[s]$ also represents the situation where s is secret-shared among shareholders.

In the additive scheme for $t = n-1$, n shares of a secret $s \in \mathbb{G}$ are random elements that satisfy $s = \sum_i [s]_i$ in some additive group $\mathbb{G}$. In Shamir's (t, n) scheme for $t < n-1$, each share is represented by $[s]_i = f_s(i)$, where f_s is a t-degree random polynomial in some finite field $\mathbb{F}$ such that $f_s(0) = s$.

Multiparty computation (MPC) is a protocol used to compute $f(x_1, \ldots, x_m)$ for a given function f and secret inputs $x_1, \ldots, x_m$. Among several approaches to construct MPCs, we focus on MPCs based on secret sharing schemes.

In an MPC for computing a function f, for each input value x_j, a party who knows the value of x_j first generates $[x_j]$ and shares them among all parties. Next, the parties jointly compute $[y] = [f(x_1, \ldots, x_m)]$ by performing a certain protocol π_f. Then, the computation result y is reconstructed from shares $[y]$. For various functions, efficient PMCs have been proposed so far. Theoretically any functions theoretically can be evaluated by MPCs. Especially, the MPC (denoted by π_{RSS}) that generates a random element allows to securely evaluate even probabilistic functions.

Efficiency of MPCs highly depends on compatibility of the evaluated function and the base field/group of the underlying secret sharing scheme. MPCs in which each value is shared over $\mathbb{F}_q$ for large prime q is suitable for arithmetic functions. On the other hand, binary MPCs in which each value is shared in binary representation, that is, shared over $(\mathbb{F}_2)^\ell$, is suitable for logical operations and integer comparisons. In order to evaluate a complex function or algorithm, MPCs for share conversion (e.g. [10]) can be used to convert shares over a domain (e.g., $\mathbb{F}_q$) into shares over another domain (e.g., $(\mathbb{F}_2)^\ell$).

As the security of MPCs, there are various attack models. They are classified based on several points of view. In this paper, we consider the model so-called honest majority with abort [11]. In this model, adversaries can be active (i.e.,

[5] In this paper, we consider only threshold schemes.

they may deviate from the protocol), but majority of parties are honest, and if a dishonest behavior is detected, parties abort the protocol and the correct output does not need to be output.

To define the security of an MPC π_f for a function $f(x_1, \ldots, x_m)$, the ideal/real simulation paradigm is often used. In the real world, an adversary A, controlling up to t parties $P_{\mathsf{A}}(\subset \{P_1, \ldots, P_n\})$, and honest parties execute π_f. Let $Real_{\mathsf{A}(z)}(x_1, \ldots, x_m)$ denote the list of output of the honest parties and A in the real execution of π_f. In this execution, A can follow any arbitrary strategy, while honest parties follow the instructions of π_f. On the other hand, in the ideal world, there are a simulator Sim controlling P_{A}, honest parties, and a functionality F_f that receives $x_1, \ldots, x_m$ from parties/simulator and outputs $f(x_1, \ldots, x_m)$. In this world, roughly, each party sends input to F_f (via a secure channel), and receives $y = f(x_1, \ldots, x_m)$ from F_f. For modeling "with abort," there are some special instructions. For detail definition, refer [11]. Let $Ideal_{Sim(z)}(x_1, \ldots, x_m)$ denote the list of output of the honest parties (that is equal to y from F_f) and Sim in the execution of the ideal world. We way π_f is a secure MPC for f if for every ppt adversary A, there exists a ppt simulator Sim s.t. $Real_{\mathsf{A}(z)}(x_1, \ldots, x_m)$ and $Ideal_{Sim(z)}(x_1, \ldots, x_m)$ are indistinguishable, where z is an auxiliary input.

If there exists a secure MPC π_f for a function f and a secure MPC π_h for a function h, we can easily construct a secure MPC $\pi_{h \circ f}$; (1) compute $[y] = [f(x)]$ from secret-shared x using π_f, (2) without reconstructing y, compute $[z] = [h(y)]$ using π_h, (3) reconstruct z from shares $[z]$.

2.2 Digital Signature

A signature scheme consists of three algorithms, $(\mathsf{SigKGen}, \mathsf{SigSign}, \mathsf{SigVerify})$.

$\mathsf{SigKGen}(1^\lambda) \to (pk, sk)$: The probabilistic key generation algorithm takes the security parameter λ as input, and generates a public (verification) key pk and a secret (signing) key sk.

$\mathsf{SigSign}(m, sk) \to \sigma$: The signing algorithm takes a message $m \in \mathcal{M}$ and a signing key sk as input, and outputs a signature σ.

$\mathsf{SigVerify}(m, \sigma, pk) \to$ acc/rej : The verification algorithm takes a message m, a signature σ, and a public key pk as input, and outputs acc or rej.

Correctness. For all $m \in \mathcal{M}$ and $(pk, sk) \leftarrow \mathsf{SigKGen}(1^\lambda)$,

$$\Pr[\mathsf{SigVerify}(m, \mathsf{SigSign}(m, sk), pk)) = \text{acc}] = 1$$

must hold.

Security. The typical security notion of digital signature scheme is existential unforgeability against chosen message attack (EUF-CMA) defined as follows.

Definition 1 (EUF-CMA for digital signature). *Let* $\Pi_{\mathrm{DS}} = (\mathsf{SigKGen}, \mathsf{SigSign}, \mathsf{SigVerify})$ *be a digital signature scheme. The EUF-CMA game between a challenger* C *and an adversary (or forger)* B *is defined as follows.*

1. *The challenger* C *generates* $(pk, sk) \leftarrow \mathsf{SigKGen}(1^\lambda)$*, and sends* pk *to* B*.*
2. B *asks a query* m_j *to the sign oracle* q_{sig} *times, where* $q_{\mathrm{sig}} = q_{\mathrm{sig}}(\lambda)$ *is a polynomial in* λ*. As the response,* C *computes* $\sigma_j \leftarrow \mathsf{SigSign}(m_j, sk)$*, and sends* σ_j *to* B*.*
3. B *outputs a forged message-signature pair* $(\tilde{m}, \tilde{\sigma})$*.*
4. *If* $\tilde{m} \neq m_j$ *for all* $j \in \{1, \ldots, q_{\mathrm{sig}}\}$ *and* $\mathsf{SigVerify}(\tilde{m}, \tilde{\sigma}, pk) = \mathrm{acc}$*,* B *wins the game.*

B*'s advantage in this game is defined as*

$$\mathsf{Adv}^{\text{euf-cma}}_{\Pi_{\mathrm{DS}},\mathsf{B}}(\lambda) := \Pr[\tilde{m} \notin \{m_1, \ldots, m_{q_{\mathrm{sig}}}\} \wedge \mathsf{SigVerify}(\tilde{m}, \tilde{\sigma}, pk) = \mathrm{acc}].$$

If $\mathsf{Adv}^{\text{euf-cma}}_{\Pi_{\mathrm{DS}},\mathsf{B}}(\lambda)$ *is* $\mathsf{negl}(\lambda)$ *for any ppt algorithm* B*,* Π_{DS} *is said to be EUF-CMA.*

2.3 Threshold Signature

Threshold signature is an extension of signature schemes in which distributed signers, rather than a single signer, jointly generate a public key and signatures associated with the public key. For typical signature schemes such as DSA and BLS signature, threshold signature schemes have been proposed [8, 16].

Each scheme has some parameters — the number of signers, the number of signers required to sign, the maximum number of colluding signers for security. For simplicity, we consider threshold signature schemes in which there are 3 signers, $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$, all of them are required to join a signing protocol in order to successfully generate a signature, and one signer can be malicious (that is, we assume no collusion of signers).[6]

For a message space $\mathcal{M}$, a threshold signature scheme Π_{tDS} associated to a digital signature scheme $\Pi_{\mathrm{DS}} = (\mathsf{SigKGen}, \mathsf{SigSign}, \mathsf{SigVerify})$ consists of a pair of protocols $(\pi_{\mathsf{SigKGen}}, \pi_{\mathsf{SigSign}})$.

$\pi_{\mathsf{SigKGen}}(1^\lambda) \to (pk, sk_1, sk_2, sk_3)$: The key generation protocol is performed by signers $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. It takes the security parameter λ as input, and at the end of the protocol, each S_i obtains a signing-key share sk_i, and all of them agree the corresponding public key pk.

$\pi_{\mathsf{SigSign}}(m, sk_1, sk_2, sk_3) \to \sigma$: The signing protocol is performed by $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. It takes a message $m \in \mathcal{M}$ as a public input and a signing-key share sk_i as a private input from S_i, and outputs a signature σ.

Correctness. For all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$, if $(pk, sk_1, sk_2, sk_3) \leftarrow \pi_{\mathsf{SigKGen}}(1^\lambda), \sigma \leftarrow \pi_{\mathsf{SigSign}}(m, sk_1, sk_2, sk_3)$, then $\Pr[\mathsf{SigVerify}(m, \sigma, pk) = \mathrm{acc}] = 1$ must hold.

Security. We consider the existential unforgeability against chosen message attack (EUF-CMA).

[6] We can similarly define the system model and security for different parameters.

Definition 2 (EUF-CMA for threshold signature[8]). *Let* $\Pi_{\mathrm{tDS}} = (\pi_{\mathsf{SigKGen}}, \pi_{\mathsf{SigSign}})$ *be a threshold signature scheme, and* $\mathsf{SigVerify}$ *is the associated verification algorithm. The EUF-CMA game between a challenger* C_{th} *and an adversary* B_{th} *controlling one signer (denoted by* S_3 *WLOG) is defined as follows.*

1. *The challenger* C_{th} *(playing the role of* S_1 *and* S_2*) and* B_{th} *(playing the role of* S_3*) jointly perform* $\pi_{\mathsf{SigKGen}}(1^\lambda)$*, and* C_{th} *obtains* pk, sk_1, sk_2*.*
2. B_{th} *asks a query* m_j *to the sign oracle* q_{sig} *times, where* $q_{\mathrm{sig}} = q_{\mathrm{sig}}(\lambda)$ *is a polynomial in* λ*.* C_{th} *(playing the role of* S_1 *and* S_2*) and* B_{th} *(playing the role of* S_3*) run* $\pi_{\mathsf{SigSign}}(m_j, sk_1, sk_2, *)$ *to compute* σ_j*.*
3. B_{th} *outputs a forged message-signature pair* $(\tilde{m}, \tilde{\sigma})$*.*
4. *If* $\tilde{m} \neq m_j$ *for all* $j \in \{1, \ldots, q_{\mathrm{sig}}\}$ *and* $\mathsf{SigVerify}(\tilde{m}, \tilde{\sigma}, pk) = \mathrm{acc}$*,* B_{th} *wins the game.*

B_{th}*'s advantage in this game is defined as*

$$\mathsf{Adv}^{\text{t-euf-cma}}_{\Pi_{\mathrm{tDS}},\mathsf{B}_{\mathrm{th}}}(\lambda) := \Pr[\tilde{m} \notin \{m_1, \ldots, m_{q_{\mathrm{sig}}}\} \wedge \mathsf{SigVerify}(\tilde{m}, \tilde{\sigma}, pk) = \mathrm{acc}].$$

If $\mathsf{Adv}^{\text{t-euf-cma}}_{\Pi_{\mathrm{tDS}},\mathsf{B}_{\mathrm{th}}}(\lambda)$ *is* $\mathsf{negl}(\lambda)$ *for any ppt algorithm* B_{th}*,* Π_{tDS} *is said to be EUF-CMA.*

If Π_{DS} has EUF-CM security and Π_{tDS} associating to Π_{DS} satisfies the simulatability defined below, then Π_{tDS} also has EUF-CMA security[8].

Definition 3 (Simulatability[8]). *Let* $\Pi_{\mathrm{tDS}} = (\pi_{\mathsf{SigKGen}}, \pi_{\mathsf{SigSign}})$ *be a threshold signature scheme.* Π_{tDS} *is said to be simulatable if it satisfies the following conditions.*

1. *There exists a simulator* sim_1 *that takes* pk *and the public information of* π_{SigKGen} *as input, and simulates the view that the adversary* A *can see in the protocol execution of* π_{SigKGen}*.*
2. *There exists a simulator* sim_2 *that takes* (m, pk) *and* $t = 1$ *secret-key share and the public information of* π_{SigSign} *as input, and simulates the view that the adversary* A *can see in the protocol execution of* π_{SigSign}*.*

2.4 (EC)DSA and Threshold (EC)DSA

DSA and its elliptic-curve variant are popular digital signature schemes, in which a key pair is $(sk = x, pk = g^x)$ over a cyclic group $\langle g \rangle$ with prime order q.

Damgard et al [5] proposed a threshold signature scheme associated to (EC)DSA. It is shown that their π_{SigKGen} and π_{SigSign} have UC-security [4] that is stronger security notion than simulatability. Therefore, the threshold signature scheme is EUF-CMA secure. In their distributed key generation protocol π_{SigKGen} consists of two steps: (1) Run the MPC π_{RSS} to generate a random secret key $sk \in \mathbb{Z}_q$, (2) run the MPC π_{PowOpen} that jointly computes g^{sk} from $[sk]$ and a public g.

2.5 Pseudo-random Function

Pseudo-random function (PRF) is a keyed function used to generate a pseudo-random string (or an element in some domain). For a PRF PRF, let $\mathcal{K}_{\mathsf{PRF}}(\lambda)$ be its key space.

Definition 4 (Pseudorandom function). *The PRF game played by a distinguisher* D *is defined as follows: First,* $b \leftarrow\!\!\$\ \{0,1\}, k \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{PRF}}(\lambda)$ *are chosen.* D *is given the security parameter* λ *and access to the oracle* $O(\cdot)$*. If* $b = 1$*,* $O(\cdot) := \mathsf{PRF}_k(\cdot)$*, otherwise (*$b = 0$*),* $O(\cdot) = \mathsf{RND}(\cdot)$*, where* RND *is a random function whose range and domain are the same as* PRF*. After polynomial times of oracle access,* D *outputs* $b' \in \{0,1\}$*.*

If for any ppt distinguisher D*, the advantage*

$$\mathsf{Adv}^{\mathrm{pr}}_{\mathsf{PRF},\mathsf{D}}(\lambda) = \left|\Pr[\mathsf{D}^O(1^\lambda) = 1 \mid b = 1] - \Pr[\mathsf{D}^O(1^\lambda) = 1 \mid b = 0]\right|$$

is negligible in λ*, then* PRF *is said to be a pseudo-random function.*

In this paper, we consider a PRF that has the following properties. (i)PRF takes a bit string as input, and outputs $x \in \mathbb{Z}_q$ for a large prime. (ii) PRF can be computed by an efficient MPC secure against active attack in the honest majority with abort model. Naor-Reingold PRF [13] is a candidate.

3 Server-aided In-brain Signature

Our goal is to provide a mechanism in which only a password in a human brain can initiate the key-generation and signing process, and its security is guaranteed without trusting all machines, but only trusting the user's brain. In this section, we introduce a new framework, called Server-aided In-brain Signature, or in-brain sig for short, as an approach to our goal.

3.1 Definition of In-brain Sig

An in-brain sig is a system by which human users securely generate their public(verification) key and signatures supported by partially trusted servers. In case servers get hacked, we assume three servers $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. Further, we assume that each user U is assigned a unique user identifier uid, chooses a password $pw \in \mathcal{PW}$ randomly in advance, and memorizes it in their brain memory, where $\mathcal{PW}$ be a pre-determined password space.

A server-aided in-brain sig scheme consists of four protocols and one algorithm $(\pi_{\mathsf{Setup}}, \pi_{\mathsf{KGen}}, \pi_{\mathsf{Sign}}, \pi_{\mathsf{Refresh}}, \mathsf{Verify})$ as follows. Let $\mathcal{M}$ be the message space.

$\pi_{\mathsf{Setup}}(1^\lambda) \to (state_1, state_2, state_3)$: The setup protocol is performed by servers $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. Each S_i obtains a secret state $state_i$ as its output.

$\pi_{\mathsf{KGen}}((uid, pw), state_1, state_2, state_3) \to pk$: The key generation protocol is performed by a (human) user U and servers $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. U's input is uid and pw, each server's input is its state $state_i$. The output is user's public key pk. After this protocol, (uid, pk) is published for later reference.[7]

[7] We assume that if (uid, pk) has been published, π_{KGen} is never done.

$\pi_{\mathsf{Sign}}(pk, (uid, m, pw), state_1, state_2, state_3) \to \sigma$: The signing protocol is performed by $\mathsf{U}, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$. U's input is uid, a message $m \in \mathcal{M}$ to be signed, and pw, whereas each server's input is $state_i$. Further, this protocol takes a public value pk, which is published with specified uid. [8] The protocol outputs a signature σ as a public output. (uid, m, σ) is published.

$\pi_{\mathsf{Refresh}}(state_1, state_2, state_3) \to (state'_1, state'_2, state'_3)$: The refresh protocol is performed by $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$ to refresh servers' states $state_i$.

$\mathsf{Verify}(pk, m, \sigma) \to$ acc/rej : The verification algorithm takes pk, m, σ as input, and outputs acc or rej as in ordinary signature schemes.

Remark 1 (On User's Terminal Devices). We assume that each user U can use two terminal devices $\mathsf{T}_1, \mathsf{T}_2$. This is because if the user relies on a single terminal device, an active malware in it may impersonate servers. To detect such a malicious behavior, we need a server authentication mechanism. At this moment we have no idea of brain-friendly server authentication system.

In all above protocols, all data sent to/from U are transmitted via T_1 or T_2. The protocols specify T_1 or T_2 to be used for each value sent to/from U.

Remark 2 (On the Refresh protocol). To protect the system against "mobile hackers," we provide a refresh protocol performed constantly, say, once a week. Mobile hackers (or originally mobile viruses) was introduce in [14] to model the behavior of hackers in the real world; hackers are always trying to take over servers, while server administrators are managing to detect and eliminate them as soon as possible, so hackers cannot stay on one server and may move to another. E.g., the first server S_1 is hijacked first, next the second server S_2 is hijacked after S_1 is cleaned. Even if only one server is hijacked at any time, the mobile hacker may gather secret information from all servers. So, continuous refreshing of secret information is generally used in proactively-secure systems [14, 7, 3].

Feasibility. For feasibility, in addition to normal requirement (e.g., all algorithms must be polynomial time), we require that all calculations by the user are simple enough to be done in the brain or using only paper and pencils.

Correctness. An in-brain sig scheme must satisfy the correctness: For all $\lambda \in \mathbb{N}$ and $m \in \mathcal{M}$, if $pw \leftarrow\!\!\$\ \mathcal{PW}$, $(state_1, state_2, state_3) \leftarrow \pi_{\mathsf{Setup}}(1^\lambda)$, $pk \leftarrow \pi_{\mathsf{KGen}}((uid, pw), state_1, state_2, state_3)$, $\sigma \leftarrow \pi_{\mathsf{Sign}}(pk, (uid, m, pw), state_1, state_2, state_3)$, then

$$\Pr[\mathsf{Verify}(pk, m, \sigma) = \text{acc}] = 1$$

holds. Furthermore, this equation should hold even if $state_1, state_2, state_3$ are refreshed by π_{Refresh} protocol for any number of times.

[8] We assume that if (uid, pk) has not been published, π_{Sign} is never done.

3.2 Security Notions

For the security of in-brain signature, we consider existential forgery as the adversary's goal, and two types of adversaries, malware in user's devices and hackers controlling servers.

We set the following assumptions.

- During the execution of π_{Setup} and π_{KGen}, all malware in user's devices and hackers controlling servers are passive adversaries, i.e., they never deviate from the protocols.
- Malware controls at most one user terminal devices in each protocol execution. This assumption is necessary to take an advantage of the use of two devices.
- A secure channel can be established between each pair of servers based on mutual authentication. Therefore, no one can eavesdrop on them or send information to a server pretending to be another server.
- When we consider the security against malware (that can be active) in a device, the other terminal and all servers are considered being under *passive* malware/hackers. Similarly, when we consider the security against hacker (that can be active) controlling a server, user terminals and other servers are considered being under *passive* malware/hackers. We leave modeling the security in the presence of independent active adversaries as a future work.

We define *selective* security in which an adversary (malware or hacker) declares the target user at the beginning of the security game.

Security against malware in user's device is defined by EUF-CMA$_{\mathrm{dev}}$ game performed by an adversary A and a challenger C. In this game, A controls either of two terminal devices and corrupts any users other than one target user, and C plays the role of the target user, all servers, and a terminal device that is not controlled by A.

EUF-CMA$_{\mathrm{dev}}$ Game. First, the adversary A outputs a user identifier uid_{t}. Let U_{t} be the target user whose user ID is uid_{t}. The challenger C executes π_{Setup} protocol, randomly chooses a password $pw_{\mathrm{t}} \leftarrow\!\!\$\ \mathcal{PW}$ of U_{t}, and initializes $\mathcal{M}_{\mathrm{signed}} := \emptyset$. Next, A issues the following four types of queries. In the following, $role(\in \{1, 2\})$ indicates the terminal device controlled by A.

Start-key-generation query StartKGen$(sid, role)$: A must issue this query only once before all StartSign queries. U_{t} and servers perform $\pi_{\mathsf{KGen}}((uid_{\mathrm{t}}, pw_{\mathrm{t}}), state_1, state_2, state_3) \rightarrow pk_{\mathrm{t}}$. C sends A pk_{t} and all values what T_{role} can see.

Start-signing query StartSign$(sid, role, m)$: If sid has already been used, this query is ignored. Otherwise, U_{t} starts the protocol $\pi_{\mathsf{Sign}}(pk_{\mathrm{t}}, (uid_{\mathrm{t}}, m, pw_{\mathrm{t}}), state_1, state_2, state_3)$. The device T_j $(j \neq role)$ is not controlled by A, so it forwards all values correctly. The protocol π_{Sign} is performed until T_{role} is supposed to send a value in the protocol description, and all values T_{role} can see is sent to A.

Send-message query Send$(sid, dest, src, msg)$: Issuing this type of query, A sends msg to the destination $dest \in \{\mathsf{U}_{\mathrm{t}}, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3\}$ as sent from the source

$src \in \{\mathsf{U}_{uid}, \mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3\}$. ($uid$ is any user identifier.) If $\textsc{Send}(sid, \mathsf{U}_\mathrm{t}, src, msg)$ is issued after $\textsc{StartSign}(sid, role, *)$ query, msg is treated as the message that src sends to U_t through the controlled device T_{role}. (If corresponding StartSign does not exist, this query is ignored.) If $dest = \mathsf{S}_i$, $src = \mathsf{U}_{uid}$ must be hold. Upon receiving msg, $dest$ starts π_{KGen} or π_{Sign}, or continues it if it has already started, following the protocol description, i.e., computes something, sends something to some one. π_{KGen} or π_{Sign} is performed until T_{role} is supposed to send a value to someone in the protocol description, and all values T_{role} can see is sent to A.

Refresh query Refresh : C performs π_{Refresh}.

When (uid, m, σ) is published after $\textsc{StartSign}(sid, *, m)$ and a series of Send queries with the same sid, C adds m to $\mathcal{M}_{\mathrm{signed}}$.

Finally, A outputs a forged message-signature pair $(\tilde{m}, \tilde{\sigma})$. If $\tilde{m} \notin \mathcal{M}_{\mathrm{signed}}$ and $\mathsf{Verify}(\tilde{m}, \tilde{\sigma}, pk_\mathrm{t}) = \mathrm{acc}$, then we say A wins the game.

Security against a mobile hacker controlling a server is defined by EUF-CMA$_{\mathrm{server}}$ game performed by an adversary A and a challenger C. In this game, A controls servers (one at any given time) and corrupts any users other than one target user, and the challenger C plays the role of the target user, un-controlled servers. Both devices are considered being under passive adversaries . So, all values sent between the target user and servers are passed directly.

EUF-CMA$_{\mathrm{server}}$ Game. First, A outputs uid_t and $i_\mathsf{A} \in \{1, 2, 3\}$. Let U_t be the target user, $\mathsf{A}_{i_\mathsf{A}}$ be the corrupted server. The challenger C executes π_{Setup} protocol, randomly chooses a password $pw_\mathrm{t} \leftarrow\!\!\$\ \mathcal{PW}$ of U_t, and initializes $\mathcal{M}_{\mathrm{signed}} := \emptyset$.

Next, A issues the following four types of queries.

Start-key-generation query $\textsc{StartKGen}(sid)$: This type of query has to be issued only once and before all StartSign queries. U_t and servers perform $\pi_{\mathsf{KGen}}((uid_\mathrm{t}, pw_\mathrm{t}), state_1, state_2, state_3) \to pk_\mathrm{t}$. C sends A all values what $\mathsf{S}_{i_\mathsf{A}}$ can see in the protocol including pk_t.

Start-signing query $\textsc{StartSign}(sid, m)$: If sid has already been used, this query is ignored. Otherwise, U_t starts the protocol $\pi_{\mathsf{Sign}}(pk_\mathrm{t}, (uid_\mathrm{t}, m, pw_\mathrm{t}), state_1, state_2, state_3)$. The protocol π_{Sign} is performed until $\mathsf{S}_{i_\mathsf{A}}$ is supposed to send a value in the protocol description, and all values $\mathsf{S}_{i_\mathsf{A}}$ can see is sent to A.

Send-message query $\textsc{Send}(sid, dest, src, msg)$: Issuing this type of query, A sends msg to a destination $dest$ as a message from a source src. $(dest, src)$ has to be $(\mathsf{S}_i, \mathsf{U}_{uid})$ for some uid and $i \neq i_\mathsf{A}$, or $(\mathsf{U}_\mathrm{t}, \mathsf{S}_{i_\mathsf{A}})$. In the case of $src = \mathsf{U}_{uid} \neq \mathsf{U}_\mathrm{t}$, the user U_{uid} is considered controlled by A. Upon receiving msg, $dest$ starts π_{KGen} or π_{Sign}, or continues it if it has already started, following the protocol description. π_{KGen} or π_{Sign} is performed until $\mathsf{S}_{i_\mathsf{A}}$ or the controlled user U_{uid} is supposed to send a value in the protocol description, and all values $\mathsf{S}_{i_\mathsf{A}}$ and U_{uid} can see is sent to A.

Refresh query $\textsc{Refresh}(i'_\mathsf{A})$: C performs π_{Refresh}. Corrupted server is changed to $\mathsf{S}_{i'_\mathsf{A}}$.

When $(uid_\mathrm{t}, m, \sigma)$ is published after $\textsc{StartSign}(sid, *, m)$ and a series of Send queries the same sid, C adds m to $\mathcal{M}_{\mathrm{signed}}$.

At the end of the game, A outputs a forged message-signature pair $(\tilde{m}, \tilde{\sigma})$. If $\tilde{m} \notin \mathcal{M}_{\text{signed}}$ and $\mathsf{Verify}(\tilde{m}, \tilde{\sigma}, pk_{\text{t}}) = \text{acc}$, then we say A wins the game.

Definition 5. *If for any ppt adversary* A*,*

$$\Pr[\mathsf{A} \textit{ wins EUF-CMA}_{\text{dev}} \textit{ game}] \leq q_{\text{send}}/|\mathcal{PW}| + \mathsf{negl}(\lambda) \tag{1}$$

holds, then the in-brain sig scheme is said to be secure against malware in a device (EUF-CMA$_{\text{dev}}$*), where* q_{send} *is the number of* Send *queries. Further, if for any ppt adversary* A*,*

$$\Pr[\mathsf{A} \textit{ wins EUF-CMA}_{\text{server}} \textit{ game}] \leq q_{\text{send}}/|\mathcal{PW}| + \mathsf{negl}(\lambda) \tag{2}$$

holds, then scheme is said to be secure against hacker controlling a server (EUF-CMA$_{\text{server}}$*).*

Remark 3. The term $q_{\text{send}}/|\mathcal{PW}|$ is corresponding to the winning probability of the online dictionary attack. To check one password in the online attack, A has to issue Send queries several times, and the number of times depends on the description of π_{Sign} protocol. Here, we use q_{send} as the number of trials.

4 In-brain Sig Based on (EC)DSA

In this section, we give a concrete construction of in-brain sig based on (EC)DSA.

4.1 Key Idea

First of all, to prevent exposing pw to malware in user's device, U should never type pw in it. Instead, in our scheme, U receives a one-time-pad key x_1 from S_1 through the first terminal device, and then sends the one-time-pad ciphertext $x_2 := pw + x_1$ to S_2 through the second one. (x_1, x_2) can be seen as shares of $(2, 2)$-threshold scheme, so not only one device infected by malware but also a server controlled by hacker cannot know pw. By S_1 and S_2 re-sharing x_1 and x_2, respectively, we easily obtain the situation that pw is shared among servers.

Next, servers jointly generate user's key pair based on shared pw. Unfortunately, human memorable passwords are not complex enough. If we simply set $sk := pw$, attackers can conduct offline dictionary attack. Note that anyone can check if guessed password is correct or not by using the published pk. To prevent offline attack, in our scheme, sk is determined by pw, the user identifier uid, and a system master key msk as

$$sk := \mathsf{PRF}_{msk}(pw \| uid).$$

By secret-sharing msk among servers beforehand, servers are able to jointly generate sk from shared pw, while an attacker cannot evaluate PRF_{msk} without legitimately performing a protocol. In this way, off-line dictionary attack can be prevented.

4.2 Protocol Description

Let the password space be $\mathcal{PW} = \mathbb{Z}_n^{\ell_{pw}}$ for some n and ℓ_{pw}. Concrete example of these parameters will be discussed later. Let $\mathsf{SigSign}^{\mathrm{DSA}}$ and $\mathsf{SigVerify}^{\mathrm{DSA}}$ be the signing and verification algorithms of DSA (or ECDSA), $\pi^{\mathrm{DSA}}_{\mathsf{SigSign}}$ be the distributed signing algorithm given in [5], $\pi_{\mathsf{RSS}}, \pi_{\mathsf{ZSS}}, \pi_{\mathrm{PowOpen}}$ be protocols described above. Let $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \{0,1\}^L \to \mathbb{Z}_q$ be a pseudorandom function, where q is the order of g, and π_{PRF} be an MPC to compute $\mathsf{PRF}_{msk}(x)$ from shared msk and x. Let π_{comv} be an MPC that converts shares in $\mathcal{PW}$ into shares in $\{0,1\}^{\ell_{pw}\log n}$.

$\underline{\pi_{\mathsf{Setup}}(1^\lambda)}$. Servers run π_{RSS} to generate random $msk \in \mathcal{K}_{\mathsf{PRF}}$.

$\underline{\pi_{\mathsf{KGen}}((uid, pw), [msk])}$. The outline of this protocol is depicted in Fig. 1.

1. U randomly chooses a session ID sid, and sends (sid, uid) to S_1 using the first terminal device T_1.
2. S_1 chooses $x_1 \leftarrow\!\!\$\ \mathcal{PW}$ randomly, sends (sid, x_1) back to U, and sends (sid, uid) to $\mathsf{S}_2, \mathsf{S}_3$.
3. U calculates $x_2 := pw + x_1 \in \mathcal{PW}$ in his brain, and sends (sid, uid, x_2) to S_2 using the second device T_2.
4. If S_2 has not received the same (sid, uid) from S_1, S_2 discards (sid, uid) and aborts the protocol. Otherwise, sends (sid, uid) to $\mathsf{S}_1, \mathsf{S}_3$.
5. S_1 and S_2 secret-share x_1 and x_2, respectively, among three servers. Each server S_i computes $[pw]_i := [x_2]_i - [x_1]_i$.
6. Performing π_{comv}, servers convert shares $[pw]$ over $\mathcal{PW}$ into $[pw]^{\mathrm{bin}}$ over $\{0,1\}^{\ell_{pw}\log n}$. Servers compute $[sk] = [\mathsf{PRF}_{msk}(pw \| uid)]$ from $[msk], [pw]^{\mathrm{bin}}$ and uid by using π_{PRF}, compute $pk := g^{sk}$ from $[sk]$ by using π_{PowOpen}, and publish (uid, pk). Finally, servers discard all information obtained in the protocol.

$\underline{\pi_{\mathsf{Sign}}(pk, (uid, m, pw), [msk])}$.

1. U accesses S_1 and gets a random session identifier sid, and sends (sid, uid, m) to S_1 using the first terminal device T_1.
2. S_1 chooses $x_1 \leftarrow\!\!\$\ \mathcal{PW}$ randomly, sends (sid, x_1) back to U, sends (sid, uid, m) to $\mathsf{S}_2, \mathsf{S}_3$.
3. U calculates $x_2 := pw + x_1$ in the brain, and sends (sid, uid, x_2, m) to S_1 through T_2.
4. S_2 sends (sid, uid, m) to $\mathsf{S}_1, \mathsf{S}_3$. If the same (sid, uid, m) has not been sent from S_1, the protocol aborts.
5. Servers jointly compute $[sk]$ in the same way as in π_{KGen}, and perform π_{PowOpen} and obtain $pk' := g^{sk}$. If $pk' \neq pk$, where pk is the published one, then this protocol aborts. If not, servers perform π_{SigSign} to compute $\sigma := \pi_{\mathsf{SigSign}}(pk, m, [sk])$ and publish (uid, m, σ). Finally, servers discard all information they obained in the protocol.

$\underline{\mathsf{Verify}(pk, m, \sigma)}$. This is exactly the same as $\mathsf{SigVerify}^{\mathrm{DSA}}$.

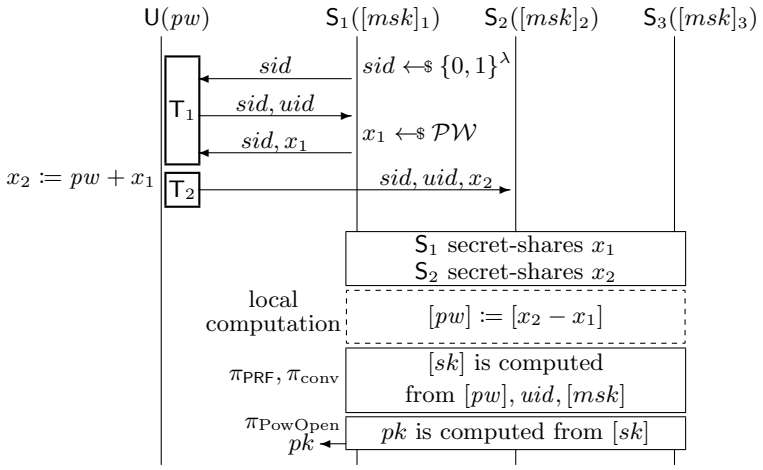

Fig. 1. Outline of π_{KGen}

$\pi_{\mathsf{Refresh}}(state_1, state_2, state_3)$. Servers jointly perform an MPC (denoted by π_{ZSS}) that generates shares of 0, and each S_i obtains z_i as a share. Then, Each S_i sets $state'_i := state_i + z_i$.

Correctness Clearly, the correctness of (EC)DSA and underlying MPCs lead the correctness of our scheme.

Feasibility It is clear that all servers' computation is done in polynomial time. What a human user needs to do in their brain is ℓ_{pw}-times addition over $\mathbb{Z}_n$. So, feasibility depends on the choice of $\mathcal{PW} = (\mathbb{Z}_n)^{\ell_{pw}}$. Here we give three choices;

- $n = 10$. As stated in [6], additive sharing over $\mathbb{Z}_{10}$ is human friendly. Even ℓ_{pw} is relatively long, addition over $\mathcal{PW}$ is feasible. The cons of this choice is that it is not so easy to memory a long decimal string, and an MPC to convert $pw \in (\mathbb{Z}_{10})^{\ell_{pw}}$ into $\{0,1\}^{\ell}$ can be costly.
- $n = 16$. We can consider addition over $\mathbb{Z}_{16}$ feasible enough in human brains, though it is a bit harder than that over $\mathbb{Z}_{10}$. Hexadecimal strings have the advantage of being shorter than decimal strings, and shares of $pw \in (\mathbb{Z}_{16})^{\ell_{pw}}$ can be converted into shares over $\{0,1\}^{\ell}$ using existing MPC such as [10].
- $n = 64$. In this case, we can use a string of ASCII characters that is generally used as a password, and realize shortest string among three choices. Also, shares of $pw \in (\mathbb{Z}_{64})^{\ell_{pw}}$ can be converted into shares over $\{0,1\}^{\ell}$ easily as the case of $n = 16$. The disadvantage of this option is that addition over $\mathbb{Z}_{64}$ is infeasible in human brains. We suggest the use of an addition table (64×64 table). All users can use the same table, and its validity is easily verified without trusting any machines.

Security For the security of our scheme is stated in the following theorem whose proof will appear in Appendix A.

Theorem 1. *If* PRF *is a pseudo-random function, all MPCs used as subprotocols are secure against active adversaries in the honest majority with abort model, then the scheme is EUF-CMA*$_{\mathrm{dev}}$ *and EUF-CMA*$_{\mathrm{server}}$*.*

5 Conclusion

The aim of our study is to realize the secure use of cryptography in the absence of fully trusted machines. In this paper, as the first attempt, we considered how to design secure signature schemes. We proposed one framework called server-aided in-brain signature, and gave one concrete construction. However, the scheme is secure against only one adversary, and we makes some assumptions on it as stated in Section 3.2. We need more study to eliminate them. In addition, it is important task to implement in-brain signature schemes and test its performance and usability.

Our another future work is construction of human-oriented cryptographic schemes other than signature schemes. An example is devising a method to encrypt messages in brain with server support but without disclosing any information about the messages to both the servers and own device terminals.

References

1. Blakley, G.R.: Safeguarding cryptographic keys. Proceedings of AFIPS 1979 National Computer Conference **48**, 313–317 (1979)
2. Boldyreva, A., Chen, S., Dupont, P.A., Pointcheval, D.: Human computing for handling strong corruptions in authenticated key exchange. In: Köpf, B., Chong, S. (eds.) CSF 2017 Computer Security Foundations Symposium. pp. 159–175. IEEE Computer Society Press (2017). https://doi.org/10.1109/CSF.2017.31
3. Boneh, D., Partap, A., Rotem, L.: Proactive refresh for accountable threshold signatures. Cryptology ePrint Archive, Paper 2022/1656 (2022), https://eprint.iacr.org/2022/1656, https://eprint.iacr.org/2022/1656
4. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). https://doi.org/10.1109/SFCS.2001.959888
5. Damgård, I., Jakobsen, T.P., Nielsen, J.B., Pagter, J.I., Østergaard, M.B.: Fast threshold ECDSA with honest majority. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 382–400. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6˙19
6. Erotokritou, S., Desmedt, Y.: Human perfectly secure message transmission protocols and their applications. In: Visconti, I., Prisco, R.D. (eds.) SCN 12. LNCS, vol. 7485, pp. 540–558. Springer, Berlin, Heidelberg (Sep 2012). https://doi.org/10.1007/978-3-642-32928-9˙30
7. Frankel, Y., Gemmell, P., MacKenzie, P.D., Yung, M.: Proactive RSA. In: Kaliski Jr., B.S. (ed.) CRYPTO'97. LNCS, vol. 1294, pp. 440–454. Springer, Berlin, Heidelberg (Aug 1997). https://doi.org/10.1007/BFb0052254
8. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Robust threshold DSS signatures. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 354–371. Springer, Berlin, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9˙31

9. Halunen, K., Latvala, O.M.: Review of the use of human senses and capabilities in cryptography. Computer Science Review **39**, 100340 (2021)
10. Kikuchi, R., Ikarashi, D., Matsuda, T., Hamada, K., Chida, K.: Efficient bit-decomposition and modulus-conversion protocols with an honest majority. In: Susilo, W., Yang, G. (eds.) ACISP 18. LNCS, vol. 10946, pp. 64–82. Springer, Cham (Jul 2018). https://doi.org/10.1007/978-3-319-93638-3˙5
11. Lindell, Y., Nof, A.: A framework for constructing fast MPC over arithmetic circuits with malicious adversaries and an honest-majority. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 259–276. ACM Press (Oct / Nov 2017). https://doi.org/10.1145/3133956.3133999
12. Matsumoto, T., Imai, H.: Human identification through insecure channel. In: Davies, D.W. (ed.) EUROCRYPT'91. LNCS, vol. 547, pp. 409–421. Springer, Berlin, Heidelberg (Apr 1991). https://doi.org/10.1007/3-540-46416-6˙35
13. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. Journal of the ACM **51**(2), 231–262 (Mar 2004). https://doi.org/10.1145/972639.972643
14. Ostrovsky, R., Yung, M.: How to withstand mobile virus attacks (extended abstract). In: Logrippo, L. (ed.) 10th ACM PODC. pp. 51–59. ACM (Aug 1991). https://doi.org/10.1145/112600.112605
15. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
16. Yu, J., Kong, F.: Forward secure threshold signature scheme from bilinear pairings. In: International Conference on Computational and Information Science. pp. 587–597. Springer (2006)

A Proof of Theorem 1

A.1 Security against Malware in User Devices

First, we show the security in the simple case in which A does not issue refresh queries. We prove with a game sequence.

$\mathbf{G}_0$: This is the EUF-CMA$_{\mathrm{dev}}$ game for our scheme. So,

$$\Pr[\mathsf{A} \text{ wins } \mathbf{G}_0] = \Pr[\mathsf{A} \text{ wins EUF-CMA}_{\mathrm{dev}} \text{ game}].$$

In our scheme, servers jointly generate a signature of some message m for U_{t} under the following three conditions.

- (i) S_1 receives $(sid, uid_{\mathrm{t}}, m)$,
- (ii) S_2 receives $(sid, uid_{\mathrm{t}}, x_2, m)$ for the same (sid, m), and
- (iii) x_1 that is chosen by S_1 as the response of (i) and x_2 that is received by S_2 in (ii) satisfy

$$\mathsf{PRF}_{msk}(pw_{\mathrm{t}} \| uid_{\mathrm{t}}) = \mathsf{PRF}_{msk}(x_2 - x_1 \| uid_{\mathrm{t}}). \tag{3}$$

In $\mathbf{G}_0$, there are three cases when these conditions are satisfied.

(a1) A receives $x_2' := pw_{\mathrm{t}} + x_1$ as the response of $\textsc{StartSign}(sid, 2, m)$, and makes a query $\textsc{Send}(sid, \mathsf{S}_2, \mathsf{U}_{\mathrm{t}}, (sid, uid_{\mathrm{t}}, x_2, m))$. In this case, a signature of m is generated only if x_1 chosen by S_1 satisfies Equation (3).

(a2) A receives x_1 as the response of $\textsc{StartSign}(sid, 1, m)$, and makes a query $\textsc{Send}(sid, \mathsf{U}_{\mathrm{t}}, \mathsf{S}_1, (sid, x_1'))$. In this case, a signature of m is generated only if $x_2 := pw_{\mathrm{t}} + x_1'$ computed by U_{t} satisfies Equation (3).

(a3) A does not make queries $\textsc{StartSign}(sid, *, m)$, but issues $\textsc{Send}(sid, \mathsf{S}_1, \mathsf{U}_{\mathrm{t}}, (sid, uid_{\mathrm{t}}, m))$, receives x_1 as the response, issues $\textsc{Send}(sid, \mathsf{S}_2, \mathsf{U}_{\mathrm{t}}, (sid, uid_{\mathrm{t}}, x_2, m))$ for some x_2. In this case, a signature of m is generated only if Equation (3) holds.

We note that if m's signature is successfully generated, m is added to $\mathcal{M}_{\mathrm{signed}}$ in the case of (a1) and (a2), but m is not added in the case of (a3).

From these observations, we consider the following game sequence.

$\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$ except that, when $\textsc{StartKGen}(*, 2)$ or $\textsc{StartSign}(*, 2, *)$ is queried, x_2 is randomly chosen from $\mathcal{PW}$ and set $x_1 := x_2 - pw_{\mathrm{t}}$, instead of U_{t} calculating $x_2 := pw_{\mathrm{t}} + x_1$. If $role = 2$, A cannot see x_1 and the relation $pw_{\mathrm{t}} = x_2 - x_1$ retains. So, this change does not affect to A's view, that is,

$$\Pr[\mathsf{A} \text{ wins } \mathbf{G}_1] = \Pr[\mathsf{A} \text{ wins } \mathbf{G}_0].$$

In this game, x_2 received from U_{t} is information theoretically independent from the password pw_{t}.

$\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$ except that C randomly chooses msk, evaluates $sk_{uid} := \mathsf{PRF}_{msk}(x_2 - x_1 \| uid)$, $pk_{uid} := g^{sk_{uid}}$ and $\sigma := \mathsf{SigSign}(m, sk_{uid})$ without using MPCs and π_{SigSign} protocol. From the correctness of the MPC and π_{SigSign}, this change does not affect the winning probability:

$$\Pr[\mathsf{A} \text{ wins } \mathbf{G}_2] = \Pr[\mathsf{A} \text{ wins } \mathbf{G}_1].$$

$\mathbf{G}_3$: In this game, $\mathsf{PRF}_{msk}(\cdot)$ is replaced with a (truly) random function $\mathsf{RND}(\cdot)$ whose domain and range are the same as PRF. A gets information about msk only through values $\mathsf{PRF}_{msk}(pw \| uid)$. Therefore there exists a distinguisher of PRF whose advantage is larger than $|\Pr[\mathsf{A} \text{ wins } \mathbf{G}_3] - \Pr[\mathsf{A} \text{ wins } \mathbf{G}_2]|$, and we have

$$|\Pr[\mathsf{A} \text{ wins } \mathbf{G}_3] - \Pr[\mathsf{A} \text{ wins } \mathbf{G}_2]| = \mathsf{negl}(\lambda)$$

if PRF is psendorandom.

In this game, Equation (3) that is the condition of generating a signature for m is replaced by the following equation.

$$\mathsf{RND}(pw_{\mathrm{t}} \| uid_{\mathrm{t}}) = \mathsf{RND}(x_2 - x_1 \| uid_{\mathrm{t}}) \tag{4}$$

In addition, all values associated to $uid (\neq uid_{\mathrm{t}})$ are independent from values associated to uid_{t}.

$\mathbf{G}_4$: In this game, C aborts the game if 1) either of (a1), (a2), (a3) defined above occurs, and 2) $x_2 - x_1 \neq pw_{\mathrm{t}}$ and Equation (4) holds.

Since RND is a random function with domain $\mathbb{Z}_q$, the abort probability is bounded by $|\mathcal{SID}|/q$, where $\mathcal{SID}$ is the set of *sid* used in the game, and we have

$$|\Pr[\mathsf{A} \text{ wins } \mathbf{G}_4] - \Pr[\mathsf{A} \text{ wins } \mathbf{G}_3]| \leq |\mathcal{SID}|/q = \mathsf{negl}(\lambda).$$

In this game, Equation (4) is replaced by the following equation.

$$pw_{\mathrm{t}} = x_2 - x_1 \tag{5}$$

Now let E_{quess} be the event that Equation (5) is satisfied in case of (a3).

$\mathbf{G}_5$: In $\mathbf{G}_5$, C aborts if E_{quess} occurs. A's view until E_{quess} occurs is independent from the value of pw_{t}. So, this event occurs with probability $|\mathcal{SID}|/|\mathcal{PW}|$ or less. Therefore, we have

$$|\Pr[\mathsf{A} \text{ wins } \mathbf{G}_5] - \Pr[\mathsf{A} \text{ wins } \mathbf{G}_4]| \leq |\mathcal{SID}|/|\mathcal{PW}|.$$

In this game, $\mathcal{M}_{\mathrm{signed}}$ includes all messages m for which C computes $\sigma = \mathsf{SigSign}(m, sk_{uid_{\mathrm{t}}})$ as the response of A's queries. So, A wins $\mathbf{G}_5$ only if forging a signature for a message $\tilde{m}$ C has never signed.

Finally, we show that A's winning probability in $\mathbf{G}_5$ is negligibly small provided that (EC)DSA is EUF-CMA by constructing an adversary B playing the EUF-CMA game using A. First, B receives pk from the challenger of EUF-CMA game. B sets $pk_{\mathrm{t}} := pk$, and runs A. Basically, B responds A's queries in the same way to the challenger of $\mathbf{G}_5$, but asks own sign oracle to compute $\sigma := \mathsf{SigSign}(m, sk_{uid_{\mathrm{t}}})$. If A outputs a forged pair $(\tilde{m}, \tilde{\sigma})$, B outputs it. It is clear that A's environment is perfectly simulated by B, and if A wins $\mathbf{G}_5$, then $\tilde{m}$ is not included in $\mathcal{M}_{\mathrm{signed}}$. Therefore, we have

$$\Pr[\mathsf{A} \text{ wins } \mathbf{G}_5] \leq \mathsf{Adv}^{\text{euf-cma}}_{\Pi_{\mathrm{DSA}},\mathsf{B}}(\lambda).$$

From the above argument, we have Equation (1).

Next, we consider the case that A issues refresh queries. Since π_{Refresh} is performed by servers, A cannot see any information by these queries. So, Equation (1) holds even in this case.

A.2 Security against Mobile Hacker controlling Servers

We use the following game sequence.

$\mathbf{G}_0$: This is the EUF-CMA$_{\mathrm{server}}$ game for our scheme.

$\mathbf{G}_1$: This game is the same as $\mathbf{G}_0$ except that all executions of MPCs are replaced by ideal functionalities as follows; In π_{Setup}, C randomly chooses $msk \leftarrow\!\!\$\ \mathcal{K}_{\mathsf{PRF}}$. In π_{KGen} and π_{Sign} protocols, each serial execution of sharing x_1, x_2, π_{conv}, π_{PRF}, π_{PowOpen} is replaced with the access to the ideal functionality $F(x_1, x_2, uid) = g^{\mathsf{PRF}_{msk}(x_2 - x_1 \| uid)}$ that is simulated by C. As the result, all servers obtain $pk = g^{\mathsf{PRF}_{msk}(x_2 - x_1 \| uid)}$. In the execution of π_{KGen}, C keeps the intermediate result $sk_{uid} := \mathsf{PRF}_{msk}(x_2 - x_1 \| uid)$ as well as pk. All executions of $\pi_{\mathsf{Sign}}(pk_{\mathrm{t}}, (uid_{\mathrm{t}}, m, pw_{\mathrm{t}}), [msk])$

are replaced with the access to the ideal functionality of $\mathsf{SigSign}(m, sk_{uid_\mathrm{t}})$. All servers receive σ as the result. When A issues REFRESH, C does nothing.

If π_{RSS}, π_{conv}, π_{PRF}, π_{PowOpen}, π_{SigSign}, π_{ZSS} are secure MPCs, there exists a simulator for each MPC that simulates the execution of the MPC. Since A can run these simulators by itself, these replacements do not affect A's winning probability except negligible probability.

In this game, a signature of m for U_t is generated only if (x_1, x_2, uid) satisfying the next equation is input to the ideal functionality to compute $F(x_1, x_2, uid)$.

$$g^{\mathsf{PRF}_{msk}(pw_\mathrm{t} \| uid_\mathrm{t})} = g^{\mathsf{PRF}_{msk}(x_2 - x_1 \| uid_\mathrm{t})} \tag{6}$$

$\mathbf{G}_2$: This game is the same as $\mathbf{G}_1$ except that the ideal functionality of $F(x_1, x_2, uid) = g^{\mathsf{PRF}_{msk}(x_2 - x_1 \| uid)}$ is replaced with $g^{\mathsf{RND}(x_2 - x_1 \| uid)}$, where RND is a random function. The difference between A's winning probabilities in $\mathbf{G}_1$ and $\mathbf{G}_2$ is bounded by the advantage of PRF, which is assumed to be negligible.

In this game, a signature of m for U_t is generated only if (x_1, x_2, uid) satisfying the next equation is passed to the ideal functionality to compute $F(x_1, x_2, uid)$.

$$g^{\mathsf{RND}(pw_\mathrm{t} \| uid_\mathrm{t})} = g^{\mathsf{RND}(x_2 - x_1 \| uid_\mathrm{t})} \tag{7}$$

$\mathbf{G}_3$: This game is the same as $\mathbf{G}_2$ except that C aborts the game if $x_2 - x_1 \neq pw_\mathrm{t}$ and $\mathsf{RND}(x_2 - x_1 \| uid_\mathrm{t}) = \mathsf{RND}(pw_\mathrm{t} \| uid_\mathrm{t})$ hold. The difference between A's winning probabilities in $\mathbf{G}_2$ and $\mathbf{G}_3$ is upper bounded by the abort probability, which is estimated as $q_{\mathrm{send}}/q = \mathsf{negl}(\lambda)$, where q_{send} is the number of *Send*.

In this game, a signature of m for U_t is generated only if (x_1, x_2, uid) satisfying the next equation is passed to the ideal functionality to compute $F(x_1, x_2, uid)$.

$$pw_\mathrm{t} = x_2 - x_1 \tag{8}$$

Let E_{guess} be the event that

- for some m, A does not issue $\textsc{StartSign}(sid, m)$ but
- issues $\textsc{Send}(sid, \mathsf{S}_1, \mathsf{U}_{uid}, (sid, uid_\mathrm{t}, m))$ and $\textsc{Send}(sid, \mathsf{S}_2, \mathsf{U}_{uid}, (sid, uid_\mathrm{t}, x_2, m))$, and
- $x_2 = pw_\mathrm{t} + x_1$ holds, where x_1 is the response of $\textsc{Send}(sid, \mathsf{S}_1, \mathsf{U}_{uid}, (sid, uid_\mathrm{t}, m))$.

If (and only if) E_{guess} occurs, m is not added to $\mathcal{M}_{\mathrm{signed}}$ even if a signature of m is generated by $\mathsf{SigSign}(m, sk_{uid_\mathrm{t}})$.

$\mathbf{G}_4$: This game is the same as $\mathbf{G}_3$ except that it is aborted if E_{guess} occurs. The A's view until this event occurs is independent from pw_t. So, the probability this event occurs is upper bounded by $|\mathcal{SID}|/|\mathcal{PW}|$. $|\mathcal{SID}|$ is upper bounded by the number of SEND queries, q_{send}. Therefore, the difference between A's winning probabilities in $\mathbf{G}_3$ and $\mathbf{G}_4$ is upper bounded by $q_{\mathrm{send}}/|\mathcal{PW}|$.

In $\mathbf{G}_4$, $\mathcal{M}_{\mathrm{signed}}$ includes all messages for which C computes $\sigma = \mathsf{SigSign}(m, sk_{uid_\mathrm{t}})$ as the response of A's queries. So, A wins $\mathbf{G}_4$ only if forging a signature for $\tilde{m}$ C has never signed.

Finally, we show that A's winning probability in $\mathbf{G}_4$ is upper bounded by the UF-CMA advantage of (EC)DSA in the following way.

Let consider the following adversary algorithm B that uses A as a subroutine. First, B receives pk^* from the challenger of EUF-CMA game. B chooses random $pw_{\mathrm{t}} \leftarrow\!\!\$\ \mathcal{PW}$, sets $pk_{\mathrm{t}} := pk^*$, and considers $\mathsf{RND}(pw_{\mathrm{t}} \| uid_{\mathrm{t}}) = sk^*$, where sk^* is the unknown secret key corresponding to pk^*. Then, B runs A. For A's queries, B can respond almost all queries simulating U_{t}, honest servers, and the functionalities, except evaluation of $F(x_1, x_2, uid_{\mathrm{t}})$ such that $x_2 - x_1 = pw_{\mathrm{t}}$ and calculation of $\mathsf{SigSign}(m, sk_{\mathrm{t}})$. In the former case, instead of evaluating $F(x_1, x_2, uid_{\mathrm{t}}) = g^{\mathsf{RND}(x_2 - x_1 \| uid_{\mathrm{t}})}$, B sets $F(x_1, x_2, uid_{\mathrm{t}}) := pk^*$. In the latter case, B makes use of own signing oracle to compute $\mathsf{SigSign}(m, sk^*)$. When A outputs a forged pair $(\tilde{m}, \tilde{\sigma})$, B outputs it.

From the modification of $\mathbf{G}_4$, if A wins the game, $(\tilde{m}, \tilde{\sigma})$ is not in $\mathcal{M}_{signed}$, that means B has never queried on $\tilde{m}$ to the signing oracle. Therefore, B's winning probability is the same as A's winning probability in $\mathbf{G}_4$.

Consequently, we have Equation (2).