Lightweight Delegated Authentication with Identity Fraud Detection for Cyber-physical Systems

Zheng Yang
Singapore University of Technology and Design, Singapore
zheng_yang@sutd.edu.sg

Chao Yin*
Chongqing University of Technology
Chongqing, China
cy_yin@2019.cqut.edu.cn

Chenglu Jin
CWI Amsterdam
Amsterdam, The Netherlands
chenglu.jin@cwi.nl

Jianting Ning
Singapore Management University
Singapore
jtning88@gmail.com

Jianying Zhou
Singapore University of Technology and Design, Singapore
jianying_zhou@sutd.edu.sg

ABSTRACT

Delegated authentication is a very popular and effective paradigm to deal with entity authentication problems for resource-constrained clients in cyber-physical systems; namely, the authentication between two clients is proxied by a trusted authentication server. However, an attacker may compromise the authentication server to impersonate the clients for sabotaging the cyber-physical systems. To detect the identity fraud attacks caused by an authentication server compromise, we propose two mutual authentication protocols by using a pseudo-random function family and a one-time signature (OTS) scheme. Our idea is to leverage the continuously evolving OTS signing and verifying keys at the signer and the verifier sides respectively for identity fraud detection because an identity fraud attack would violate the victim's honest OTS key update procedure. The proposed protocols are proven secure under a new mutual authentication security model that formulates the identity fraud detection.

CCS CONCEPTS

• **Security and privacy** → **Key management**; **Multi-factor authentication**.

KEYWORDS

Authentication, Identity fraud detection, Intrusion detection, Key management

ACM Reference Format:
Zheng Yang, Chao Yin, Chenglu Jin, Jianting Ning, and Jianying Zhou. 2021. Lightweight Delegated Authentication with Identity Fraud Detection for Cyber-physical Systems. In *Proceedings of the 7th ACM Cyber-Physical System Security Workshop (CPSS '21), June 7, 2021, Virtual Event, Hong Kong*. ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3457339.3457984

*Corresponding author

1 INTRODUCTION

A cyber-physical system (CPS) is a system comprised of various physical devices that are interconnected over a network to exchange data autonomously. Nowadays, CPS can not only facilitate the daily lives of people but also improve industrial efficiency, where the concrete CPS includes railway systems, water treatment systems, and many other Internet-of-Things systems. However, due to the computational limitation of those client devices, the security of CPS is always not a priority, which results in that malicious attackers can attack CPS in many ways [14, 21]. The first step to prevent unauthorized access to those critical devices is to authenticate the identity of communication principles before they can accept commands or exchange data. Practical cross-domain communication scenarios in CPS include: multi-branch communications (e.g., communication between two stations of a railway system), and edge-computing driven communication between two companies' CPS. In these scenarios, each entity is typically registered to a domain-specific server, such as a station management server (for railway communications) or enterprise authentication server (for edge-computing communications). The two communicating entities from different CPS domains typically neither share a symmetric key nor possess each other's public key certificate, so the authentication between them is usually delegated by their own domain servers.

Many lightweight authentication protocols have been proposed in the context of delegated authentication [4, 10, 11, 19, 23, 24, 37, 38]. Specifically, the authentication of clients (including users and constrained devices) is outsourced to a trustworthy third-party authentication server, which relies on a symmetric key framework. In the traditional framework, there is no shared key between clients, but each client may have a pre-shared key with the authentication server. However, an obvious drawback is that the compromise of an authentication server implies a compromise of the whole system since the attacker would know all authentication keys. That is, an attacker who compromised the authentication server can impersonate any client. In practice, the compromise of an authentication server may also be caused by malicious insiders who try to sabotage (e.g., impersonate clients to send malicious instructions to client devices in a critical CPS). To the best of our knowledge, there is no existing technique to identify which authentication server is compromised in such a delegated authentication framework. Due to the existence of zero-alarm attacks [3, 29], it may be even harder for

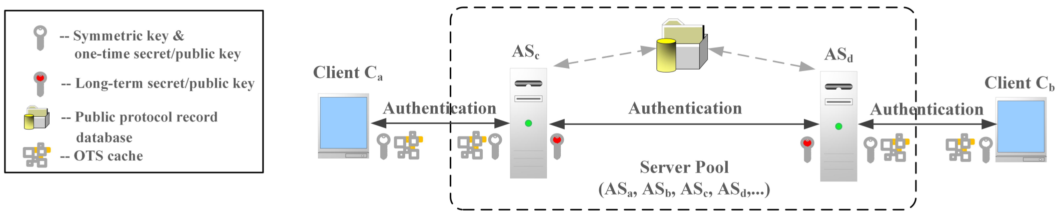
-- Symmetric key & one-time secret/public key
-- Long-term secret/public key
-- Public protocol record database
-- OTS cache
Client C_a
Authentication
AS_c
Authentication
AS_d
Authentication
Client C_b
Server Pool
(AS_a, AS_b, AS_c, AS_d,...)

Figure 1: Overview of Our Delegated Mutual Authentication Protocol

administrators to identify the compromised server since there is no obvious trace to track such attacks. To this end, we are motivated to seek a more affirmative way to detect the identity fraud events caused by a compromised authentication server, so that we can not only fix the compromised server but also help administrators identify the responsible staff for the identity fraud events (e.g., in the case of insider attacks).

In a large CPS, like manufacturing systems and water treatment systems, multiple controllers (called Programmable Logic Controller or PLC in such systems) sometimes need to communicate with each other to cooperate to complete a complex process. Since each controller only has very limited computational power for security enhancement, it is particularly interesting to shift the burden of authentication to the servers, which leads to the delegated authentication paradigm. Moreover, due to the air-gapped nature of these systems, the servers, which are connected to the enterprise network and sometimes the Internet, are more vulnerable to remote attacks. Thus, we are also interested in solving the security issues after a fraction of servers are compromised and impersonate other controllers in the network. By introducing identity fraud detection, we will be able to audit the system and detect such an attack.

Our Work. To mitigate the compromise of an authentication server in the delegated authentication framework, we focus on studying identity fraud detection (IFD) in this work. Our goal is to design lightweight mutual authentication protocols that enable an honest authentication server to detect not only a compromised client but also the identity fraud events caused by other compromised authentication servers. An overview of our proposed protocol is presented in Fig. 1. Note that we adopt a similar system model standardized for cross-domain authentication [34].[1]

In our protocol, each client has a separate authentication delegator which is selected from a pool of authentication servers. Our main idea for achieving IFD is to leverage evolving one-time signature (OTS) keys (of clients and servers). We allow a party to commit (sign) its next public key in a signature using its current OTS secret key, which would lead to a signature chain on the fly. The authentication delegator selected by a client is responsible for the update of its OTS key once so that it will be marked as unavailable after the corresponding session is over. The authentication delegator will become available after its authentication task is checked by the other authentication servers. For a pool with size z, every OTS key update procedure in a session will be checked by $z - 1$ authentication servers in the subsequent sessions.

[1]In contrast, we here focus on delegated authentication for machine-to-machine, not password-based setting.

To resist network failures, a party should locally 'cache' the currently unchecked signature-chain and 'replay' it in the next session. When an authentication server finishes its delegated authentication task, it should faithfully include the OTS public key evolving procedure into a public database to ensure that all sessions should be auditable. Hence each of the $z - 1$ authentication servers should check the OTS key update procedure of the client by comparing the client's OTS cache and the historical protocol transcripts recorded in the database to figure out identity fraud attacks. However, using only OTS for authentication is subject to network attacks. This is because an attacker can interfere with the OTS public key update procedure and force the cached signature-chain to be over-long. For example, the attacker can keep initiating sessions on the client-side to generate and cache many new OTS public keys (in the signature chain) without sending back the authentication server's acknowledgment messages.

To prevent network attackers, we adopt a hybrid approach such that the mutual authentication protocol is achieved by using both long-term symmetric key-based message authentication code and asymmetric key-based one-time signature (OTS). Firstly, most of the protocol messages are protected by the message authentication code computed based on the long-term pre-shared key. The long-term key-based authentication is useful in the IFD. This is because, if the message authentication codes are valid in a session, but the relevant one-time signature is invalid, then it implies that the honest party must have been impersonated before. Secondly, the OTS ensures the following two security properties (as defined in [20]): 1) forward security, which protects the compromised OTS secret keys from affecting the security of previous sessions using uncorrupted keys; 2) backward security, which means that if a party runs the mutual authentication to update the compromised OTS secret key before the adversary uses it in an impersonation attack, then the compromised key would become invalid (due to its one-time feature). Note that the backward security cannot be achieved by using forward secure signature [18]. And OTS is more computationally efficient.

To evaluate the performance of our proposed methods in a real-world scenario, we implemented a two-party system. The server in the system is a laptop, while a Raspberry Pi based OpenPLC is used as a CPS device. We choose to use OpenPLC because the commercial PLCs on the market are all closed-source, and do not allow us to modify their implementations. OpenPLC is an open-source implementation of a programmable logic controller (PLC) [5], which is widely used in the community of CPS security researchers. The experimental results show that our proposed framework is efficient in terms of communication, computation, and memory utilization in a cyber-physical system if the parameters are chosen properly.

Our contributions can be summarized as follows:

- A two-party mutual authentication protocol 2MA, which provides weak impersonation detection when the victim is not controlled by the attackers.
- A delegated mutual authentication protocol 4MA, which is built from 2MA and provides the server's strong identity fraud detection even when the server is a malicious insider.
- A new security model that is proposed to capture the mutual authentication with the server's strong identity fraud detection.
- The proposed protocols are proven secure in the standard model.

Organization. We introduce necessary preliminaries in Section 2. The security model of mutual authentication with identity fraud detection is presented in Section 3. Section 4 presents two mutual authentication protocols of which the second protocol can provide the server's identity fraud detection. The security of the proposed protocols is analyzed in Section 5. Performance analysis and evaluation results are presented in Section 6. We review the literature related to our work in Section 7. The paper concludes in Section 8.

2 PRELIMINARIES

Notations. Let $\kappa \in \mathbb{N}$ be the security parameter and 1^κ be a string that consists of κ ones. We denote with $[n] = \{1, \dots, n\} \subset \mathbb{N}$ the set of integers between 1 and n. If X is a set, then $x \overset{\$}{\leftarrow} X$ denotes the action of sampling a uniformly random element from X. If X is a probabilistic algorithm, then $x \overset{\$}{\leftarrow} X$ denotes that X is run with fresh random coins and returns x. Assuming that X is a set, we denote with $|X|$ the operation to obtain the number of elements in X. We may use the operation $X[i]$ to get access to the i-th element of X. We let the operation $\|$ denote the concatenation of two strings.

In the following, we review the cryptographic primitives [16] used in our constructions.

Digital Signature Schemes. We define a (regular) digital signature scheme SIG with three probabilistic polynomial time (PPT) algorithms (SIG.Gen, SIG.Sign, SIG.Vfy). A one-time signature (OTS) scheme is denoted by OTS = (OTS.Gen, OTS.Sign, OTS.Vfy), which is a special case of SIG. We assume that a signature scheme is associated with public and secret key spaces $\{\mathcal{PK}_{\mathsf{SIG}}, \mathcal{SK}_{\mathsf{SIG}}\}$, a randomness space $\mathcal{R}_s$ for key generation, message space $\mathcal{M}_{\mathsf{SIG}}$, and signature space $\mathcal{S}_{\mathsf{SIG}}$ in the security parameter κ. We denote the bit-length of space $\mathcal{R}_s$ by ℓ_s which is determined by κ. The algorithms of SIG are defined as follows:

- $\mathsf{SIG.Gen}(1^\kappa, rs)$: This algorithm takes as input the security parameter κ and a random value $rs \overset{\$}{\leftarrow} \mathcal{R}_{\mathsf{SIG}}$, and outputs a (public) verification key $vk \in \mathcal{PK}_{\mathsf{SIG}}$ and a signing key $sk \in \mathcal{SK}_{\mathsf{SIG}}$.
- $\mathsf{SIG.Sign}(sk, m)$: This is the signing algorithm that generates a signature $\sigma \in \mathcal{S}_{\mathsf{SIG}}$ for a message $m \in \mathcal{M}_{\mathsf{SIG}}$ using sk.
- $\mathsf{SIG.Vfy}(vk, m, \sigma)$: This is the verification algorithm that takes as input a verification key vk, a message m and a signature σ, outputs 1 if σ is a valid signature for m under vk, and 0 otherwise.

Let $\mathcal{SIG}(sk, \cdot)$ be a signing oracle which on input message m returns a signature $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$. We use a list SR to record all tuple (m_i, σ_i) where m_i and σ_i are the input and output of i-th $\mathcal{SIG}$ oracle query respectively.

Definition 1. *For a signature scheme* SIG = (SIG.Gen, SIG.Sign, SIG.Vfy) *and an adversary $\mathcal{F}$, we define the following experiment:*

$\mathsf{EXP}^{\text{seuf-cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, q)$: $(sk, vk) \overset{\$}{\leftarrow} \mathsf{SIG.Gen}(1^\kappa)$; $(\sigma^*, m^*) \leftarrow \mathcal{F}^{\mathcal{SIG}(sk,\cdot)}(vk)$, *where* $\mathcal{SIG}(sk, \cdot)$ *can be asked at most q times; Return 1, if* $\mathsf{SIG.Vfy}(pk, m^*, \sigma^*) = 1$, *and* $(m^*, \sigma^*) \notin \mathsf{Slist}$; *Output 0 otherwise.*

We define the advantage of $\mathcal{F}$ in the above experiment as: $\mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, q) := \Pr[\mathsf{EXP}^{\text{seuf-cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, q) = 1]$. *We say that* SIG *is secure against strong existential forgeries $\mathcal{F}$ under adaptive chosen-message attacks (SEUF-CMA), if for all PPT adversaries $\mathcal{F}$ the advantage* $\mathsf{Adv}^{\text{seuf-cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, q)$ *is negligible. If $q = 1$ then* SIG *is called as a SEUF-CMA secure OTS scheme.*

Collision-Resistant Hash Functions. Let $\mathsf{CRH} : \mathcal{K}_{\mathsf{CRH}} \times \mathcal{M}_{\mathsf{CRH}} \to \mathcal{Y}_{\mathsf{CRH}}$ be a family of keyed-hash functions where $\mathcal{K}_{\mathsf{CRH}}$ is the key space, $\mathcal{M}_{\mathsf{CRH}}$ is the message space and $\mathcal{Y}_{\mathsf{CRH}}$ is the hash value space. The public key $hk_{\mathsf{CRH}} \in \mathcal{K}_{\mathsf{CRH}}$ defines a hash function, denoted by $\mathsf{CRH}(hk_{\mathsf{CRH}}, \cdot)$. On input a message $m \in \mathcal{M}_{\mathsf{CRH}}$, this function $\mathsf{CRH}(hk_{\mathsf{CRH}}, m)$ generates a hash value $y \in \mathcal{Y}_{\mathsf{CRH}}$. When the hash key hk_{CRH} is obvious from the context, we write $\mathsf{CRH}(m)$ for $\mathsf{CRH}(hk_{\mathsf{CRH}}, m)$.

Definition 2. *For a hash function* $\mathsf{CRH} : \mathcal{K}_{\mathsf{CRH}} \times \mathcal{M}_{\mathsf{CRH}} \to \mathcal{Y}_{\mathsf{CRH}}$ *and an adversary $\mathcal{H}$, we define the following experiment:*

$\mathsf{EXP}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{F}}(\kappa)$: $hk_{\mathsf{CRH}} \overset{\$}{\leftarrow} \mathcal{K}_{\mathsf{CRH}}$; $(m, m') \leftarrow \mathcal{F}(hk_{\mathsf{CRH}})$; *Return 1, if* $(m, m') \in \mathcal{M}_{\mathsf{CRH}}$ *and* $\mathsf{CRH}(m) = \mathsf{CRH}(m')$; *Output 0 otherwise.*

We define the advantage of $\mathcal{F}$ in the above experiment as $\mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) := \Pr[\mathsf{EXP}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) = 1]$. *We say that* CRH *is* collision-resistant *if for all PPT adversary $\mathcal{H}$ the advantage* $\mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa)$ *is negligible.*

Pseudo-Random Functions. We denote by $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ a pseudo-random function family, where $\mathcal{K}_{\mathsf{PRF}}$ is the key space, $\mathcal{D}_{\mathsf{PRF}}$ is the domain and $\mathcal{R}_{\mathsf{PRF}}$ is the range of PRF for security parameter κ. Let PList be a list to store the messages queried in the PRF oracle $\mathcal{FN}(k, x)$ which just returns $\mathsf{PRF}(k, x)$.

Definition 3. *Given a* $\mathsf{PRF} : \mathcal{K}_{\mathsf{PRF}} \times \mathcal{D}_{\mathsf{PRF}} \to \mathcal{R}_{\mathsf{PRF}}$ *and an adversary* $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$, *we define the following experiment:*

$\mathsf{EXP}^{\text{IND-CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, q)$: $b \overset{\$}{\leftarrow} \{0, 1\}, k \overset{\$}{\leftarrow} \mathcal{K}_{\mathsf{PRF}}$; $(x^*, st) \leftarrow \mathcal{B}_1^{\mathcal{FN}(k,\cdot)}$, *s.t.* $x^* \notin \mathsf{PList}$; *if* $x \in \mathsf{PList}$ *then return a failure* $\perp$; $V_1^* := \mathsf{PRF}(k, x^*), V_0^* \overset{\$}{\leftarrow} \mathcal{R}_{\mathsf{PRF}}$; $b' \leftarrow \mathcal{B}_2^{\mathcal{FN}(k,\cdot)}(st, V_b^*)$; *If $b = b'$ then return 1; Otherwise return 0;*

We define the advantage of $\mathcal{B}$ in the above game as: $\mathsf{Adv}^{\text{IND-CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, q) := \left|\Pr[\mathsf{EXP}^{\text{IND-CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, q) = 1] - \frac{1}{2}\right|$, *where q is the number $\mathcal{FN}$ queries that can be asked by $\mathcal{B}$. The pseudo-random function family* PRF *is said to be a secure, if for all PPT adversaries $\mathcal{B}$ the advantage* $\mathsf{Adv}^{\text{IND-CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, q)$ *is negligible.*

3 SECURITY MODEL

In this section, we define a security model for mutual authentication with identity fraud detection (MA-IFD) protocols, which is adapted from [1, 7, 26]. In particular, we formulate two security properties of MA-IFD protocols, including mutual authentication (MA) and identity fraud detection (IFD).

Execution Environment. Let $u, v, d \in \mathbb{N}$ be positive integers. We consider an environment with multiple parties that include u clients and v authentication servers, where $u \geq v \geq 2$. We assume that each client has a unique identity C and each server is uniquely identified by an identity S. We will denote with $\mathcal{ID}$ the general identity for a party such that $\mathcal{ID} \in \{\mathsf{C}, \mathsf{S}\}$. Each party $\mathcal{ID}_i$ may carry out at most $\rho \in \mathbb{N}$ sessions that are simulated by a set of oracles $\{\Pi^s_{\mathcal{ID}_i} : i \in [\ell], s \in [\rho]\}$, where $\ell = u$ and $\rho = d$ if $\mathcal{ID}_i$ is client, otherwise $\ell = v$ and $\rho = u \cdot v \cdot d + v \cdot d$. We assume that an oracle would terminate a session with acceptance if and only if it has sent or received the last protocol message and all received messages are valid according to the protocol specification; otherwise it would terminate with rejection. Meanwhile, each client can only sequentially execute the protocol, i.e., the $s+1$-th oracle can be activated if and only if the s-th oracle of that party is terminated. The authentication server S can only sequentially run oracles with a specific client but execute oracles with different clients concurrently. That is, $\Pi^s_{\mathsf{S}_i}$ and $\Pi^t_{\mathsf{S}_j}$ can run simultaneously with a different communication partner.[2] For entity authentication purpose, we assume that each party has a secret/public key pair (sk, pk) which is evolvable (so that each oracle may have a distinct secret/public key pair), but we do not require the client and the authentication server has the same type of secret/public key pair, e.g., the authentication server's secret/public key pair $(sk_{\mathsf{S}}, pk_{\mathsf{S}})$ may have many sub-secret/public key pairs each of which is used to authenticate itself to a specific client. Each oracle can have access to its own secret key, all public keys of other parties, and any other public information. Moreover, each oracle $\Pi^s_{\mathcal{ID}}$ maintains a list of independent internal state variables including:

- $\Phi^s_{\mathcal{ID}}$ – execution-state $\Phi^s_{\mathcal{ID}} \in \{\mathtt{accept}, \mathtt{reject}\}$;
- $\mathsf{pid}^s_{\mathcal{ID}}$ – identities of the communication partners;
- $T^s_{\mathcal{ID}}$ – messages orderly sent and received by $\Pi^s_{\mathcal{ID}}$.

Here we review a notion regarding the partnership of two oracles during online interaction. We denote with $\mathsf{PTF} : (\mathcal{ID}_i, s, \mathsf{pid}^s_{\mathcal{ID}_i}, T^s_{\mathcal{ID}_i}) \rightarrow (\mathcal{ID}_j, t)$ a partner function [7] which takes input the internal states $\mathsf{pid}^s_{\mathcal{ID}_i}$ and $T^s_{\mathcal{ID}_i}$ of an oracle $\Pi^s_{\mathcal{ID}_i}$, and outputs its partner oracle (session) $\Pi^t_{\mathcal{ID}_j}$, such that $\mathcal{ID}_j \in \mathsf{pid}^s_{\mathcal{ID}_i}$. We realize the partner function PTF which outputs $(\mathcal{ID}_j, t)$ if and only if all the following conditions hold: i) both $\Pi^s_{\mathcal{ID}_i}$ and $\Pi^t_{\mathcal{ID}_j}$ have sent their last protocol message without rejection; ii) $\mathcal{ID}_j \in \mathsf{pid}^s_{\mathcal{ID}_i}$ and $\mathcal{ID}_i \in \mathsf{pid}^t_{\mathcal{ID}_j}$; iii) $T^s_{\mathcal{ID}_i} \subseteq T^t_{\mathcal{ID}_j}$ or $T^t_{\mathcal{ID}_j} \subseteq T^s_{\mathcal{ID}_i}$; iii) $\Pi^t_{\mathcal{ID}_j}$ is the unique oracle satisfying all above conditions.

Adversarial Model. We define an active adversary $\mathcal{A}$ as a probabilistic polynomial time (PPT) algorithm which can ask the following queries.

- $\mathsf{Send}(\mathcal{ID}_i, s, m)$: $\mathcal{A}$ can use this query to send any message m to an initialized oracle $\Pi^s_{\mathcal{ID}_i}$. The oracle will respond to the next message m^* (if any) to be sent according to the protocol specification and its internal states.
- $\mathsf{Corrupt}(\mathcal{ID}_i)$: This query returns the current secret key $sk_{\mathcal{ID}_i}$ to $\mathcal{A}$. We say that the party $\mathcal{ID}_i$ submitted to this query is *corrupted*. Once a party $\mathcal{ID}_i$ is corrupted, no oracle of $\mathcal{ID}_i$ will be initialized any more.

[2]These restrictions are required by our OTS public key evolving scenario.

Security Definitions. In the following, we formulate the security properties for mutual authentication protocols, and, especially for server identity fraud detection. To formulate identity fraud detection, we first define a class of clients that are deceived by a compromised authentication server.

Deceived Clients. We say a client C_i is *deceived* if there is an oracle $\Pi^s_{\mathsf{C}_i}$ that accepts without a partner oracle at the uncorrupted client $\mathsf{C}_j \in \mathsf{pid}^s_{\mathsf{C}_i}$ (if any) which will be called as *impersonated* client. This issue also implies an identity fraud event.

Direct Delegator. We say an authentication server S_j is a direct delegator for C_i in an oracle $\Pi^s_{\mathsf{C}_i}$ if $\mathsf{S}_j \in \mathsf{pid}^s_{\mathsf{C}_i}$ and S_j is supposed to directly verify the authentication proof (e.g., a signature) of C_i.

For security definition, we first define a security experiment that is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ based on an MA-IFD protocol Σ. Let MP be a variable denoting one of the security properties in $\{\mathsf{MA}, \mathsf{MA\text{-}IFD}\}$, where MA is mutual authentication, and MA-IFD is mutual authentication with (server) identity fraud detection.

Security Experiment $\mathsf{EXP}_{\Sigma,\mathcal{A}}(q, \kappa, \mathsf{MP})$: the security experiment is represented as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on a target AKA protocol Σ, where the following steps are performed: i) *Setup*: $\mathcal{C}$ initializes the execution environment by realizing the oracles $\{\Pi^s_{\mathsf{C}_i} : i \in [u], s \in [d]\}$ for clients and $\{\Pi^t_{\mathsf{SID}_j} : j \in [v], t \in [2u \cdot d]\}$ for authentication servers. $\mathcal{C}$ returns all identities and public keys to $\mathcal{A}$; ii *Query Phase*: $\mathcal{A}$ may issue a polynomial number of Send and Corrupt queries; iii) *Finalization*: $\mathcal{A}$ terminates the game and outputs a tuple $(\mathcal{ID}^*_i, s^*, t^*)$. The experiment outputs 1 if and only if $\Pi^{s^*}_{\mathcal{ID}^*_i}$ accepts and one of the following conditions is held:

- **C1**: $\mathsf{MP} \in \{\mathsf{MA}, \mathsf{MA\text{-}IFD}\}$, and all parties in $\mathsf{pid}^{s^*}_{\mathcal{ID}^*_i}$ are uncorrupted, and there is a party $\mathcal{ID}_j \in \mathsf{pid}^{s^*}_{\mathcal{ID}^*_i}$ such that $\mathcal{ID}_j$ has no unique partner oracle to $\Pi^{s^*}_{\mathcal{ID}^*_i}$;
- **C2**: $\mathsf{MP} = \mathsf{MA\text{-}IFD}$ and $\mathcal{ID}^*_i = \mathsf{C}^*_i$, and all the following conditions are satisfied: i) $\mathcal{A}$ only asked q times Corrupt query to authentication servers; ii) C^*_i is deceived in $\Pi^{t^*}_{\mathcal{ID}^*_i}$ such that $t^* < s^*$; iii) C^*_i's direct delegator $\mathsf{S}_c \in \mathsf{pid}^{s^*}_{\mathsf{C}^*_i}$ is not corrupted and S_c has a partner oracle $\Pi^w_{\mathsf{S}_c}$ to $\Pi^{s^*}_{\mathsf{C}^*_i}$, and the other client $\mathsf{C}_j \in \mathsf{pid}^{t^*}_{\mathcal{ID}^*_i}$ is not corrupted before $\Pi^w_{\mathsf{S}_c}$ accepts.[3]

Definition 4 (MA-IFD Security). *We define the advantage of an adversary $\mathcal{A}$ against a MA-IFD protocol Σ with parameter $q < v$ as* $\mathsf{Adv}_{\Sigma,\mathcal{A}}(q, \kappa, \mathsf{MP}) := \Pr[\mathsf{EXP}_{\Sigma,\mathcal{A}}(q, \kappa, \mathsf{MP}) = 1]$. *We say that MA-IFD protocol Σ is secure, if for all PPT adversaries $\mathcal{A}$, the advantage* $\mathsf{Adv}_{\Sigma,\mathcal{A}}(q, \kappa, \mathsf{MP})$ *is negligible.*

4 A MUTUAL AUTHENTICATION FRAMEWORK WITH IDENTITY FRAUD DETECTION

In this section, we will introduce a new authentication framework that can provide identity fraud detection (IFD) capability. We start

[3]This IFD rule indicates that the honest authentication server S_c fails to identify that C^*_i is deceived.

with the construction for two-party mutual authentication (2MA), which relies on both PRF and OTS. The proposed 2MA can provide weak client identity fraud detection. The 'weak' means that the identity fraud event is caused by outsider attackers who compromise the authentication keys without controlling the victim's device. Next, we show how to generically leverage on the proposed 2MA protocol as a building block to construct a MA-IFD protocol 4MA that can provide strong server identity fraud detection (i.e., even if the compromised server is an insider attacker which takes control of the server). Our MA-IFD protocol 4MA involves four parties including two clients and two authentication servers (AS) for simplicity. Each client's authentication is outsourced to an authentication server. In our protocol constructions, we include three standard building blocks: i) a one-time signature scheme OTS = (OTS.Gen, OTS.Sign, OTS.Vfy); ii) a regular (multiple-time) signature scheme SIG = (SIG.Gen, SIG.Sign, SIG.Vfy); iii) a collision-resistant hash function CRH which has a random hash key $hk_{\mathsf{CRH}} \overset{\$}{\leftarrow} \mathcal{K}_{\mathsf{CRH}}$; and iv) a pseudo-random function family PRF.

4.1 Two-party Mutual Authentication with Weak Identity Fraud Detection

Design Rationale. Our 2MA protocol is adapted from the traditional challenge-response paradigm with several important modifications. Essentially, we combine symmetric key-based PRF with asymmetric key-based OTS for authentication. First, the pre-shared symmetric key can be compromised from either key sharer, so it is hard to identify the party who leaks the key. To mitigate the drawback of symmetric key-based authentication and achieve identity fraud detection, we leverage OTS for authentication that each OTS key pair needs to be updated to a new one in each session. If the OTS key update procedure is abnormal, then there must be some identity fraud events.

Since the OTS secret key can be used only one-time, we adopt a cached signature-chain technique to update the OTS public key. Specifically, a party commits its next (nx) OTS public key in the signature generated based on the current OTS secret key, so that the OTS public keys of a party are chained along with the generation of signatures. In particular, the transmitted signatures may be lost due to network failures. Each party should cache the signature relevant data (e.g., the public key used to verify the signature and sign hashed message) until it is verified. A party can send its cached data for verification in the next session. That is, the current unverified public key of a client can be validated by checking whether there is a valid signature-chain starting from using the secret key of the last verified public key (see more details in Algorithm 1). To track the protocol executions, the authentication server must record the transcript of protocol messages in a session, which is used later for identity fraud detection. According to the recorded transcript and the evolving OTS public key, a client can identify whether it has been impersonated before.

In practice, an adversary who eavesdrops on the network may disrupt the communication so that the cached signature-chain may grow very fast on the client-side. To deal with this problem, we rely on the symmetric key-based authentication to guarantee that network adversary cannot manipulate the exchanged messages (including nonce and one-time signatures). Meanwhile, the result of symmetric key-based authentication can help a party identify the key compromise events. Namely, the proposed protocol can ensure that only the compromised party leads to its communication partner's signature-chain being over-long.

Protocol Description. We let $2\mathsf{MA}(\mathcal{ID}_a, \mathsf{S}_c, \mathsf{AI}_{\mathcal{ID}_a}, \mathsf{AI}_{\mathsf{S}_c,\mathcal{ID}_a})$ denote a protocol instance of our two-party MA protocol executed between two parties $\mathcal{ID}_a$ and S_c with auxiliary inputs $\mathsf{AI}_{\mathcal{ID}_a}$ and $\mathsf{AI}_{\mathsf{S}_c,\mathcal{ID}_a}$ from them respectively. Each execution of 2MA outputs the protocol transcript $T_{\mathcal{ID}_a}$ and $T_{\mathsf{S}_c,\mathcal{ID}_a}$ from the view of $\mathcal{ID}_a$ and S_c respectively, where $\mathcal{ID}_a$ can be either client or server. Here we mainly depict the protocol from the view of the client, i.e., $\mathcal{ID}_a = \mathsf{C}_a$ for instance. As for $\mathcal{ID}_a = \mathsf{S}_a$, the protocol execution is similar, but no OTS public key update is required since both parties will use long-term keys for authentication. We denote by ast $\in$ {No-Client-Partner, Client-Partner-Auth} the authentication assertion issued by an authentication server, where No-Client-Partner means there is no partnered client, and Client-Partner-Auth means the partner's identity is authenticated. For 2MA, ast is always No-Client-Partner. If ast = No-Client-Partner, then the communication entities include the client and its direct delegator. Let $\mathcal{R}_N = \{0,1\}^{\ell_r}$ be a nonce space, where ℓ_r is the bit-length determined by κ.

We describe the proposed two-party protocol 2MA as follows.

Parties. In our construction, we consider that there are u clients which could be either users or devices (e.g., programmable logic controller), and v authentication servers (AS) (which form a pool). Each client is identified by a unique identifier C_a for $a \in [u]$, and each authentication server has an identity S_c for $c \in [v]$. For the IFD purpose, we will let these authentication servers work in turns for a client (as mentioned in Section 1).

Setup. To track the OTS public key update procedure, we assume each party exploits a queue data structure called 'SCache', such that each item in this list is a tuple $(\mathcal{ID}_a, pk^{ots}, h, pk^{ots,nx}, \sigma)$, where $\sigma := \mathsf{OTS.Sign}(sk^{ots}, h||pk^{ots,nx})$ is a signature of the hashed message h concatenating with the public key $pk^{ots,nx}$, pk^{ots} is the public key used for verifying the current signature, and $\mathcal{ID}_a$ is the party which is supposed to verify the signature (i.e., the intended communication partner of the owner of pk^{ots}). We assume that an element can be only popped up from the queue's head and pushed into the tail of the queue.

Let ClearSC($parm$, SCache) be a function that takes as input a cache SCache and a parameter $parm \in (pk^{ots}, \mathcal{ID}_a)$, and it does the following steps: i) If $parm$ is a public key pk^{ots}, then it deletes all tuples before the one containing pk^{ots} from SCache; ii) If $parm$ is an identity $\mathcal{ID}_a$, it deletes all tuples identified by $\mathcal{ID}_a$ from SCache.

The authentication server S_c first generates a set of pre-shared symmetric keys $\{K_{\mathsf{S}_c,\mathsf{C}_a}\}_{a\in[u]}$ and securely distributes them to the corresponding client over a secure channel, e.g., the client C_a has the key $K_{\mathsf{S}_c,\mathsf{C}_a}$. Second, S_c generates OTS secret/public pairs, which are used for authenticating itself to each client, i.e., for $a \in [v]$ it runs $(sk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}) \overset{\$}{\leftarrow} \mathsf{OTS.Gen}(1^\kappa, rs_{\mathsf{S}_c,\mathsf{C}_a})$, where $rs_{\mathsf{S}_c,\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_s$. Meanwhile, the public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ will be sent to the client C_a. Third, S_c generates a long-term secret/public pair $(sk^{sig}_{\mathsf{S}_c}, pk^{sig}_{\mathsf{S}_c}) \overset{\$}{\leftarrow}$

$\mathsf{SIG.Gen}(1^\kappa)$. In particular, the authentication server S_c would keep variables:

- $\{\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}\}_{a\in[u]}$: a set of queues, each of which stores the one-time signature relevant data of S_c in sessions communicating with the client C_a. Each used OTS public key of the signer S_c, which is not checked by the intended verifier C_a (i.e., without receiving the acknowledgement from C_a), will be stored into the corresponding $\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}$.

Each client C_a initiates the protocol by generating a secret/public pair $(sk^{ots}_{\mathsf{C}_a}, pk^{ots}_{\mathsf{C}_a}) \overset{\$}{\leftarrow} \mathsf{OTS.Gen}(1^\kappa, rs_{\mathsf{C}_a})$, where $rs_{\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_s$. C_a initializes and locally keeps the following variables:

- $\mathsf{AS\_PK}_{\mathsf{C}_a} = \{pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}\}_{c\in[v]}$: a set of authentication servers's OTS public keys. Each public key will be used to verify the signature from the corresponding authentication server S_c.
- $\{\mathsf{C}_a\_\mathsf{S}_c.\mathit{Available}\}_{c\in[v]}$: a set of variables regarding available status of authentication servers, such that $\mathsf{C}_a\_\mathsf{S}_c.\mathit{Available} \in \{0,1\}$, where 1 denotes that AS_a is available for the client C_a, and 0 otherwise. These variables are initially set to be 1. Note that when $\mathsf{C}_a\_\mathsf{S}_c.\mathit{Available} = 0$, another party $\mathsf{C}_y \neq \mathsf{C}_a$ can still choose S_c for authentication if $\mathsf{C}_y\_\mathsf{S}_c.\mathit{Available} = 1$.
- $\mathsf{ASQueue}_{\mathsf{C}_a}$: a queue that stores the identities of direct authentication servers, which are currently selected to do both authentication and identity fraud detection for C_a. We implicitly assume that the queue is initialized with size $z = |\mathsf{ASQueue}_{\mathsf{C}_a}|$ by a queue initialization algorithm, and z available authentication servers' identities are pushed into this queue initially.
- $\mathsf{IFDCache}_{\mathsf{C}_a}$: a queue that stores the information regarding the OTS key update procedure, which is used for identity fraud detection. Each record in $\mathsf{IFDCache}$ is a tuple in the form of $(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a})$, where $\mathsf{S}_c \in \mathsf{ASQueue}_{\mathsf{C}_a}$, and $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ is the public key used to verify the received signature $\sigma_{\mathsf{S}_c,\mathsf{C}_a}$, $h_{\mathsf{S}_c,\mathsf{C}_a}$ is the signed hash message, and $pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}$ is supposed to be the next OTS public key of S_c for authenticating to C_a.
- $\mathsf{SCache}_{\mathsf{C}_a}$: OTS signature relevant data kept by C_a, which has the data structure described above.

In addition, we assume that there is a public database $\mathcal{PDB}$, such that each record in $\mathcal{PDB}$ is a tuple $(\mathsf{S}_c, h, T, \sigma_{\mathsf{S}_c})$, where S_c is the authentication server which submitted the record, h is a hash value related to T, T is a transcript including all protocol messages sent and received by S_c in a session, and σ_{S_c} is a signature $\sigma_{\mathsf{S}_c} := \mathsf{SIG.Sign}(sk^{sig}_{\mathsf{S}_c}, T)$. We assume that an authentication server inserts new items into $\mathcal{PDB}$.

Authentication. The online authentication steps are shown in Figure 2. Two parties would first generate two new OTS public keys respectively, and exchange two nonces $(N_{\mathsf{C}_a}, N_{\mathsf{S}_c,\mathsf{C}_a})$, where S_c is the last the authentication server in the queue $\mathsf{ASQueue}_{\mathsf{C}_a}$. Meanwhile, S_c should authenticate the identities of participants and exchange nonces with a message authentication code A_1 generated by PRF based on the pre-shared key $K_{\mathsf{S}_c,\mathsf{C}_a}$, which convinces the client regarding the existence of S_c in the corresponding session.

Next, in addition to signing the protocol transcript of messages exchanged, the client C_a would sign the data in $\mathsf{IFDCache}_{\mathsf{C}_a}$ and $\mathsf{SCache}_{\mathsf{C}_a}$ which include the used OTS public keys from both client and authentication servers respectively. Namely, C_a generates the

Algorithm 1: SChainVfy: Signature-Chain Verification

```
Input: $\mathcal{ID}, pk, \sigma, h, pk^{nx}$, SCache
Output: 0, 1
$L := |\mathsf{SCache}|$
if $pk$ equals to current unused one recorded either in $\mathcal{PDB}$ or locally by a
  client then
    goto Current
if $\mathcal{ID} = \mathsf{C}$ and $pk^1 \in \mathsf{SCache}$ is not recorded locally then
    return 0          \\ $pk^1$ is the first public key of $\mathcal{ID}$ in SCache
if $\mathcal{ID} = \mathsf{S}$ and $pk^1 \in \mathsf{SCache}$ is not recorded in $\mathcal{PDB}$ then
    return 0
for $t = 1$ to $L$ do
    $vr := \mathsf{SIG.Vfy}(pk^t, h^t||pk^{nx,t}, \sigma^t)$    \\ $t$ is the item index of SCache

    if $pk^{nx,t} \in \mathcal{PDB}$ and $sk^{nx,t}$ has been used to sign $h'$ s.t. $h' \neq h^t$ then
        $vr := 0$          \\ i.e., there is valid signature related to $h'$;
    if $vr = 0$ then
        return 0
Current: $vr := \mathsf{SIG.Vfy}(pk, h||pk^{nx}, \sigma)$
return $vr$
```

signature $\sigma_{\mathsf{C}_a} := \mathsf{OTS.Sign}(sk^{ots}_{\mathsf{C}_a}, h_{\mathsf{C}_a}||pk^{ots,nx}_{\mathsf{C}_a})$, where $h_{\mathsf{C}_a} := \mathsf{CRH}(T_{\mathsf{C}_a})$ and $T_{\mathsf{C}_a} := \mathsf{C}_a||\mathsf{S}_j||N_{\mathsf{C}_a}||\mathsf{AI}_{\mathsf{C}_a}||N_{\mathsf{S}_c,\mathsf{C}_a}||\mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a}||\mathsf{A}_1|| pk^{ots}_{\mathsf{C}_a}||\mathsf{SCache}_{\mathsf{C}_a}||\mathsf{IFDCache}_{\mathsf{C}_a}||pk^{ots,nx}_{\mathsf{C}_a}||pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}$. The client would also record the tuple $(pk^{ots}_{\mathsf{C}_a}, h_{\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{C}_a}, \sigma_{\mathsf{C}_a})$ into $\mathsf{SCache}_{\mathsf{C}_a}$ (in case of any message loss), so that C_a can show the OTS public key evolving procedures recorded in $\mathsf{SCache}_{\mathsf{C}_a}$ next time. The signature σ_{C_a} should also be protected by a message authentication code A_2 generated based on $K_{\mathsf{S}_c,\mathsf{C}_a}$ and PRF. Then, the signature σ_{C_a}, C_a's current public key $pk^{ots}_{\mathsf{C}_a}$, the client cached signature relevant data set $\mathsf{SCache}_{\mathsf{C}_a}$, the cached servers' signature relevant data set $\mathsf{IFDCache}_{\mathsf{C}_a}$, its next (nx) public key $pk^{ots,nx}_{\mathsf{C}_a}$, the latest OTS public key of $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ stored at C_a, and the message authentication code A_2, are sent to the authentication server S_c.

Upon receiving the message m_3, S_c would first check the message authentication code A_2. If A_2 is valid, then S_c runs the signature-chain verification algorithm SChainVfy which is defined by Algorithm 1, to verify the signature based on the data cached in $\mathsf{SCache}_{\mathsf{C}_a}$. If C_a's signature is valid and the corresponding public key is recorded correctly, then SChainVfy returns 1. After this, S_c would sign the hash of the protocol transcript $T_{\mathsf{S}_c,\mathsf{C}_a} = T_{\mathsf{C}_a}||\sigma_{\mathsf{C}_a}|| pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}||\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}||pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}||\mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a}$. Next, S_c would clear its signature cache by running $\mathsf{ClearSC}(pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}', \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}$ according to the public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}'$ stored at C_a, i.e., all tuples before the one containing $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}'$ will be deleted from $\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}$. Furthermore, S_c would push the tuple $(pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a})$ into $\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}$, and update its current OTS public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ to the next one $pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}$. Finally, $\sigma_{\mathsf{S}_c,\mathsf{C}_a}$ and its relevant message authentication code A_3, and the messages in $T_{\mathsf{S}_c,\mathsf{C}_a}$ which are not known by C_a are sent to C_a in m_4. Upon receiving m_4, the client C_a verifies A_3 and $\sigma_{\mathsf{S}_c,\mathsf{C}_a}$ analogously.

Post-acceptance. This phase is shown in Figure 2. For identity fraud detection, we need first to set the available status of S_c at C_a to be 0, i.e., $\mathsf{C}_a\_\mathsf{S}_c.\mathit{Available} := 0$. Before confirming that S_c is not a cheater, it should be no longer selected to authenticate for C_a.

C_a $(K_{\mathsf{S}_c,\mathsf{C}_a}, sk^{ots}_{\mathsf{C}_a}, pk^{ots}_{\mathsf{C}_a}, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}')$		S_c $(\mathsf{S}_c = \mathsf{ASQueue}_{\mathsf{C}_a}[z])$ $(K_{\mathsf{S}_c,\mathsf{C}_a}, sk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a})$
	Online Authentication	
$rs_{\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_s$		$rs_{\mathsf{S}_c,\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_s$
$(sk^{ots,nx}_{\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{C}_a}) \leftarrow \mathsf{OTS.Gen}(1^\kappa, rs_{\mathsf{C}_a})$		$(sk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}) \leftarrow \mathsf{OTS.Gen}(1^\kappa, rs_{\mathsf{S}_c,\mathsf{C}_a})$
$N_{\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_N$		$N_{\mathsf{S}_c,\mathsf{C}_a} \overset{\$}{\leftarrow} \mathcal{R}_N$
$m_1 := \mathsf{C}_a \Vert \mathsf{S}_c \Vert N_{\mathsf{C}_a} \Vert \mathsf{AI}_{\mathsf{C}_a}$	$\xrightarrow{m_1}$	
		$\mathsf{A}_1 := \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, m_1 \Vert N_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a})$
Reject if $\mathsf{A}_1 \neq \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, m_1 \Vert N_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a})$	$\xleftarrow{m_2}$	$m_2 := N_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{A}_1$
$T_{\mathsf{C}_a} := m_1 \Vert m_2 \Vert pk^{ots}_{\mathsf{C}_a} \Vert \mathsf{SCache}_{\mathsf{C}_a} \Vert \mathsf{IFDCache}_{\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{C}_a} \Vert pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}'$		
$h_{\mathsf{C}_a} := \mathsf{CRH}(T_{\mathsf{C}_a})$		
$\sigma_{\mathsf{C}_a} := \mathsf{OTS.Sign}(sk^{ots}_{\mathsf{C}_a}, h_{\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{C}_a})$		
$\mathsf{A}_2 := \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{C}_a})$		
$m_3 := \sigma_{\mathsf{C}_a} \Vert pk^{ots}_{\mathsf{C}_a} \Vert \mathsf{SCache}_{\mathsf{C}_a} \Vert \mathsf{IFDCache}_{\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{C}_a} \Vert pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}' \Vert \mathsf{A}_2$	$\xrightarrow{m_3}$	Reject if $\mathsf{A}_2 \neq \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{C}_a})$
$(pk^{ots}_{\mathsf{C}_a}, h_{\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{C}_a}, \sigma_{\mathsf{C}_a}) \overset{\text{push}}{\rightarrow} \mathsf{SCache}_{\mathsf{C}_a}$		Reject if $\mathsf{SChainVfy}(\mathsf{S}_c, pk^{ots}_{\mathsf{C}_a}, \sigma_{\mathsf{C}_a}, h_{\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{C}_a}) \neq 1$
$pk^{ots}_{\mathsf{C}_a} := pk^{ots,nx}_{\mathsf{C}_a}$		$T_{\mathsf{S}_c,\mathsf{C}_a} = T_{\mathsf{C}_a} \Vert \sigma_{\mathsf{C}_a} \Vert pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a}$
		$h_{\mathsf{S}_c,\mathsf{C}_a} := \mathsf{CRH}(T_{\mathsf{S}_c,\mathsf{C}_a})$
		$\sigma_{\mathsf{S}_c,\mathsf{C}_a} := \mathsf{OTS.Sign}(sk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a})$
		$\mathsf{A}_3 := \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a})$
Reject if $\mathsf{A}_3 \neq \mathsf{PRF}(K_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a})$	$\xleftarrow{m_4}$	$m_4 := \mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \sigma_{\mathsf{S}_c,\mathsf{C}_a} \Vert pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a} \Vert pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a} \Vert \mathsf{A}_3$
Reject if $\mathsf{SChainVfy}(\mathsf{C}_a, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a}) \neq 1$		$\mathsf{ClearSC}(pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}{}', \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a})$
		$(pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}) \overset{\text{push}}{\rightarrow} \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a}$
		$pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a} := pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}$
accept		accept
	Post-acceptance	
$\mathsf{C}_a\_\mathsf{S}_c.\mathit{Avaliable} := 0$		$T_{\mathsf{S}_c,\mathsf{C}_a} = T_{\mathsf{S}_c,\mathsf{C}_a} \Vert \sigma_{\mathsf{S}_c,\mathsf{C}_a}$
$(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a},$		$\sigma_{\mathsf{S}_c} := \mathsf{SIG.Sign}(sk^{sig}_{\mathsf{S}_c}, T_{\mathsf{S}_c,\mathsf{C}_a})$
$\mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a}) \overset{\text{push}}{\rightarrow} \mathsf{IFDCache}_{\mathsf{C}_a}$		store $(\mathsf{S}_c, h_{\mathsf{S}_c,\mathsf{C}_a}, T_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c})$ into $\mathcal{PDB}$
$\mathsf{S}_y \overset{\text{pop}}{\leftarrow} \mathsf{ASQueue}_{\mathsf{C}_a}[1]$		
$\mathsf{C}_a\_\mathsf{S}_y.\mathit{Avaliable} := 1$		
$(\mathsf{S}_y, pk^{ots}_{\mathsf{S}_y,\mathsf{C}_a}, h_{\mathsf{S}_y,\mathsf{C}_a}, \sigma_{\mathsf{S}_y,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_y,\mathsf{C}_a}, \mathsf{SCaches}_{\mathsf{S}_y,\mathsf{C}_a}) \overset{\text{pop}}{\leftarrow} \mathsf{IFDCache}_{\mathsf{C}_a}$		
$pk^{ots}_{\mathsf{S}_y,\mathsf{C}_a} := pk^{ots,nx}_{\mathsf{S}_y,\mathsf{C}_a}$		
Delete $(h_{\mathsf{S}_y,\mathsf{C}_a}, \sigma_{\mathsf{S}_y,\mathsf{C}_a}, \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a})$		
$\mathsf{ClearSC}(\mathsf{S}_y, \mathsf{SCache}_{\mathsf{C}_a})$		
Randomly select S_x, s.t. $\mathsf{C}_a\_\mathsf{S}_x.\mathit{Available} = 1$		
$\mathsf{S}_x \overset{\text{push}}{\rightarrow} \mathsf{ASQueue}_{\mathsf{C}_a}[1]$		

Figure 2: A Two-party MA Protocol from OTS.

C_a would also push the tuple $(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCaches}_{\mathsf{S}_c,\mathsf{C}_a})$ into $\mathsf{IFDCache}_{\mathsf{C}_a}$, which reflects the usage of S_c's secret key. The contents of $\mathsf{IFDCache}_{\mathsf{C}_a}$ will be checked by other authentication servers later. Meanwhile, C_a would pop up the first authentication server S_y and delete all its related data in $\mathsf{IFDCache}_{\mathsf{C}_a}$ and $\mathsf{SCache}_{\mathsf{C}_a}$ since its cached data must have been verified by $z-1$ authentication servers (if no one has reported the identify fraud event about it). Eventually, C_a randomly selects an authentication server S_x whose available status $\mathsf{C}_a\_\mathsf{S}_x.\mathit{Available} = 1$.

On the server side, S_c is supposed to faithfully store its protocol transcript $T_{\mathsf{S}_c,\mathsf{C}_a}$ into $\mathcal{PDB}$ together with a signature σ_{S_c} that is generated based on its long-term secret key, i.e., the tuple $(\mathsf{S}_c, T_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c})$ is stored into $\mathcal{PDB}$. Therefore, any other parties (e.g., the next authentication server selected for authenticating C_a) can check the honesty of S_c (e.g., as done by Algorithm 2 which is discussed later).

Weak Client Identity Fraud Detection. If the client C_a's secret keys (including OTS secret key and pre-shared key $K_{\mathsf{S}_c,\mathsf{C}_a}$) are compromised by an attacker $\mathcal{A}$, then $\mathcal{A}$ can impersonate C_a. If C_a is impersonated but not controlled by $\mathcal{A}$, then the most recent OTS public recorded in $\mathcal{PDB}$ must be distinct to the one kept by C_a, which means that the real C_a cannot be authenticated by an authentication server anymore. In particular, if an uncorrupted authentication server received a valid A_2 but the underlying signature σ_{C_a} (authenticated by A_2) is invalid, then S_c would know that C_a must have been impersonated before. Similar detection approaches can also be applied to the authentication server. However, the above IFD approach can only resit outsider attackers who do not control the device of a victim (either client or server). An attacker who controls the victim's device would not leave out any proof for IFD (i.e., inconsistent OTS public key), because all OTS keys would be updated and stored in the device faithfully by the attacker.

4.2 Delegated Mutual Authentication with Strong Server Identity Fraud Detection

In this section, we focus on constructing a MA-IFD protocol 4MA by generically leveraging 2MA, which can provide strong server identity fraud detection.

Design Rationale. In 4MA, the authentication between two clients C_a and C_b are delegated by authentication servers S_c and S_d respectively, so that the two-party mutual authentication protocol instances $\mathsf{2MA}(\mathsf{C}_a, \mathsf{S}_c, \mathsf{AI}_{\mathsf{C}_a}, \mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a})$, $\mathsf{2MA}(\mathsf{S}_c, \mathsf{S}_d, \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})$ and $\mathsf{2MA}(\mathsf{C}_b, \mathsf{S}_d, \mathsf{AI}_{\mathsf{C}_b}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{C}_b})$ are executed. Meanwhile, the 2MA instance between client and authentication server is run based on OTS, but the 2MA instance between two authentication servers is run based on regular multiple-time signature scheme (without SCache). Recall that, in 2MA, each party can specify auxiliary input AI to run the protocol. Therefore, we can specifically define the contents of auxiliary inputs to glue up the 2MA instances to eventually yield 4MA. Namely, we leverage on AI to convey the intended partner's identity and nonces generated in 2MA instances so that each party can uniquely have a partner session at each intended communication partner. Furthermore, we will add a few protocol steps to achieve identity fraud detection capability.

Protocol Description. The setup and the post-acceptance procedures are identical to that of 2MA. The protocol execution is shown by Figure 3, and the protocol messages are listed in Table 1. In the protocol execution of 4MA, we only slightly modify the timing of the protocol messages of 2MA instances and the assertion variables as follows:

- After first two protocol steps in $\mathsf{2MA}(\mathsf{S}_c, \mathsf{S}_d, \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})$, two authentication servers suspend to wait for the one-time signatures and the relevant message authentication codes from the clients respectively (before sending m_4). If a server received either an invalid signature or invalid message authentication code from the client, then it rejects the session $\mathsf{2MA}(\mathsf{S}_c, \mathsf{S}_d, \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})$.
- If S_x for $x \in \{c, d\}$, the corresponding party rejects the session $\mathsf{2MA}(\mathsf{S}_c, \mathsf{S}_d, \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})$, then S_x sets the authentication assertion as $\mathsf{ast}_{\mathsf{S}_x,\mathsf{C}_y} :=$ No-Client-Partner, where C_y for $y \in \{a, b\}$ is the client that is directly delegated by S_x; Otherwise, S_x sets $\mathsf{ast}_{\mathsf{S}_x,\mathsf{C}_y} :=$ Client-Partner-Auth.
- If S_c happens to be equivalent to S_d, then the session $\mathsf{2MA}(\mathsf{S}_c, \mathsf{S}_d, \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}, \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})$ does not need to execute.

Strong Server Identity Fraud Detection. Our goal is to enable a client to provide a 'proof' regarding any identity fraud event that a compromised authentication server, which can be controlled by attackers (e.g., malicious insiders), impersonated as the honest client to communicate with another honest client. Recall that in the post-acceptance phase, each authentication server should sign and insert their protocol transcript into the public database $\mathcal{PDB}$ so that all other authentication servers can verify the protocol executions later. Note that from the protocol transcript, any party can 'replay' the corresponding 2MA instance. If the most recent protocol transcript in $\mathcal{PDB}$ involving C_a and S_c is valid then the up-to-date OTS public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ is recorded and valid as well. We stress that the identity fraud cache $\mathsf{IFDCache}_{\mathsf{C}_a}$ can provide a 'proof' if C_a is deceived by an authentication server before. Generally speaking, an honest authentication server can identify an identity fraud event from $\mathsf{IFDCache}_{\mathsf{C}_a}$ by checking whether the relevant transcript has been correctly recorded in $\mathcal{PDB}$ and the sessions implied by the corresponding transcript have been faithfully executed (e.g., each session has a unique partner session). We stress that the current OTS public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ of S_c stored at either the client C_a must be checked by other $z-1$ authentication servers in ASQueue before it can be used to verify the next signature (i.e., after a successful OTS public key update in the post-acceptance phase of 2MA). Among those $z-1$ authentication servers, we assume that at least one of them is compromised, so that the IFD works if and only if the adversaries corrupted at most $z-2$ servers. By applying our IFD procedure, the OTS public key $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$ in a tuple $(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a})$ from $\mathsf{IFDCache}_{\mathsf{C}_a}$ should exist in $\mathcal{PDB}$ as well, otherwise S_c must have behaved dishonestly. Similarly, the hash value of the protocol transcript, which involves the usage of $pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}$, should be recorded in $\mathcal{PDB}$ if S_c is honest. However, for a malicious S_c which tries to impersonate C_a, it cannot write a valid protocol transcript involving C_a since without knowing C_a's secret key S_c is unable to generate a valid (unforgeable) signature for executing the protocol. This is also why we need the client to provide a proof.

To quickly identify a partner session, we let GetSID be a function which takes as input a transcript $T_{\mathsf{S}_c,\mathsf{C}_a}$, and outputs a session identifier $\mathsf{sid} = \mathsf{C}_a||\mathsf{C}_b||\mathsf{S}_c||\mathsf{S}_d||N_{\mathsf{C}_a}||N_{\mathsf{C}_b}||N_{\mathsf{S}_c,\mathsf{C}_a}||N_{\mathsf{S}_c,\mathsf{S}_d}||N_{\mathsf{S}_d,\mathsf{S}_b}$, where the identities are sored in lexicographic order. If two sessions recorded by transcripts $T_{\mathsf{S}_c,\mathsf{C}_a}$ and $T_{\mathsf{S}_d,\mathsf{C}_b}$ (respectively) are partnered then it must have that $\mathsf{GetSID}(T_{\mathsf{S}_c,\mathsf{C}_a}) = \mathsf{GetSID}(T_{\mathsf{S}_d,\mathsf{C}_b})$.

More specifically, to achieve identity fraud detection, when S_e receives a valid signature from C_a, it runs the Algorithm 2 which takes as input $\mathsf{IFDCache}_{\mathsf{C}_a}$, and output either a tuple $(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a})$ related to an identity fraud event or 0 meaning that no identity fraud event is found. If an identity fraud event is found, then the deceived client C_c and the cheater S_d should stop working before they are fixed. Here, we send the IFDCache along with the authentication (for simplicity), that may eat some bandwidth for online authentication. But we stress that IFDCache can be sent to the next intended authentication server S_x right after the last session is terminated and before the current (newly updated) OTS key of the client is used. In the meantime, the client should use the pre-shared key $K_{\mathsf{S}_x,\mathsf{C}_a}$ to protect the integrity of $\mathsf{IFDCache}_{\mathsf{C}_a}$ (based on a message authentication code scheme) when transmitting it.[4] In this scenario, the client can even trigger the identify fraud detection immediately after a session is done.

5 SECURITY ANALYSIS

5.1 Security of 2MA

Theorem 1. *We assume that $\ell_r \leq \ell_s$, the one-time signature* OTS *is SEUF-CMA secure, the hash function* CRH *is collision-resistant, the pseudo-random function family* PRF *is secure, and the size of* ASQueue *has a size z such that $3 \leq z \leq v$. Then the proposed two-party mutual authentication protocol* 2MA *is secure with* $\mathsf{Adv}_{\mathsf{2MA},\mathcal{A}}(z-2, \kappa, \mathsf{MA}) \leq \frac{(3u\cdot v\cdot d)^2}{2^{\ell_s}} + \mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) + u\cdot v\cdot \mathsf{Adv}^{\mathsf{IND\text{-}CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, 3u\cdot v\cdot d + v\cdot d) + 2u\cdot v\cdot d\cdot \mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{OTS},\mathcal{F}}(\kappa, 1)$.

We prove this theorem via a sequence of games following [35]. In the proof, we first change the games to exclude the collision

[4]We stress that only the use of the pre-shared key (without the OTS signature) is enough to protect $\mathsf{IFDCache}_{\mathsf{C}_a}$ since if S_x is honest (not compromised) then the pre-shared key is uncompromised either. For a compromised S_x, the integrity of $\mathsf{IFDCache}_{\mathsf{C}_a}$ is useless.

$$\begin{tikzcd}
\mathsf{C}_a & & \mathsf{S}_c & & \mathsf{S}_d & & \mathsf{C}_b \\
T_{\mathsf{C}_a} \arrow[rr, leftrightarrow, "2\mathsf{MA}(\mathsf{C}_a{,}\mathsf{S}_c{,}\mathsf{AI}_{\mathsf{C}_a}{,}\mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a})"] & & T_{\mathsf{S}_c,\mathsf{C}_a},\ T_{\mathsf{S}_c,\mathsf{S}_d} \arrow[rr, leftrightarrow, "2\mathsf{MA}(\mathsf{S}_c{,}\mathsf{S}_d{,}\mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}{,}\mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c})"] & & T_{\mathsf{S}_d,\mathsf{S}_c},\ T_{\mathsf{S}_d,\mathsf{C}_b} \arrow[rr, leftrightarrow, "2\mathsf{MA}(\mathsf{C}_b{,}\mathsf{S}_d{,}\mathsf{AI}_{\mathsf{C}_b}{,}\mathsf{AI}_{\mathsf{S}_d,\mathsf{C}_b})"] & & T_{\mathsf{C}_b}
\end{tikzcd}$$

Figure 3: A MA-IFD Protocol 4MA **from** 2MA.

Table 1: Message Transcripts and Auxiliary Inputs.

Variables	Contents
T_{C_a}	$\mathsf{C}_a \| \mathsf{S}_c \| N_{\mathsf{C}_a} \| N_{\mathsf{S}_c,\mathsf{C}_a} \| pk^{ots}_{\mathsf{C}_a} \| \mathsf{SCache}_{\mathsf{C}_a}$ $\| \mathsf{IFDCache}_{\mathsf{C}_a} \| pk^{ots,nx}_{\mathsf{C}_a} \| \mathsf{AI}_{\mathsf{C}_a} \| \mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a}$
$T_{\mathsf{S}_c,\mathsf{C}_a}$	$T_{\mathsf{C}_a} \| \sigma_{\mathsf{C}_a} \| pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a} \| \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a} \| pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}$
$T_{\mathsf{S}_c,\mathsf{S}_d}$	$\mathsf{S}_c \| \mathsf{S}_d \| N_{\mathsf{S}_c,\mathsf{S}_d} \| N_{\mathsf{S}_d,\mathsf{S}_c} \| \mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d} \| \mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c}$
$T_{\mathsf{S}_d,\mathsf{S}_c}$	$T_{\mathsf{S}_c,\mathsf{S}_d} \| \sigma_{\mathsf{S}_c,\mathsf{S}_d}$
T_{C_b}	$\mathsf{C}_b \| \mathsf{S}_d \| N_{\mathsf{C}_b} \| N_{\mathsf{S}_d,\mathsf{C}_b} \| pk^{ots}_{\mathsf{C}_b} \| \mathsf{SCache}_{\mathsf{C}_b}$ $\| \mathsf{IFDCache}_{\mathsf{C}_b} \| pk^{ots,nx}_{\mathsf{C}_b} \| \mathsf{AI}_{\mathsf{C}_b} \| \mathsf{AI}_{\mathsf{S}_d,\mathsf{C}_b}$
$T_{\mathsf{S}_d,\mathsf{C}_b}$	$T_{\mathsf{C}_b} \| \sigma_{\mathsf{C}_b} \| pk^{ots}_{\mathsf{S}_d,\mathsf{C}_b} \| \mathsf{SCache}_{\mathsf{S}_d,\mathsf{C}_b} \| pk^{ots,nx}_{\mathsf{S}_d,\mathsf{C}_b}$
$\mathsf{AI}_{\mathsf{C}_a}$	C_b
$\mathsf{AI}_{\mathsf{S}_c,\mathsf{C}_a}$	$N_{\mathsf{C}_b} \| \mathsf{S}_d \| N_{\mathsf{S}_d,\mathsf{S}_c} \| N_{\mathsf{S}_d,\mathsf{C}_b}$
$\mathsf{AI}_{\mathsf{S}_c,\mathsf{S}_d}$	$\mathsf{C}_a \| N_{\mathsf{C}_a} \| N_{\mathsf{S}_c,\mathsf{C}_a}$
$\mathsf{AI}_{\mathsf{S}_d,\mathsf{S}_c}$	$\mathsf{C}_b \| N_{\mathsf{C}_b} \| N_{\mathsf{S}_d,\mathsf{C}_b}$
$\mathsf{AI}_{\mathsf{C}_b}$	C_a
$\mathsf{AI}_{\mathsf{S}_d,\mathsf{C}_b}$	$N_{\mathsf{C}_a} \| \mathsf{S}_c \| N_{\mathsf{S}_c,\mathsf{S}_d} \| N_{\mathsf{S}_c,\mathsf{C}_a}$

Algorithm 2: IFDetection: Identity Fraud Detection

```
Input: $\mathsf{IFDCache}_{\mathsf{C}_a}$
Output: 0 or $(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a})$
$res := \emptyset$
while $\mathsf{IFDCache}_{\mathsf{C}_a} \neq \emptyset$ do
    res:=$(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a}, \mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a},$
    $\mathsf{ast}_{\mathsf{S}_c,\mathsf{C}_a}) \overset{\text{pop}}{\leftarrow} \mathsf{IFDCache}_{\mathsf{C}_a}$
    if $\mathsf{SChainVfy}(\mathsf{S}_c, pk^{ots}_{\mathsf{S}_c,\mathsf{C}_a}, \sigma_{\mathsf{S}_c,\mathsf{C}_a}, h_{\mathsf{S}_c,\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{S}_c,\mathsf{C}_a},$
    $\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}) = 1$ and $h_{\mathsf{S}_c,\mathsf{C}_a} \notin \mathcal{PDB}$ then
        goto IFfound
    if $h_{\mathsf{S}_c,\mathsf{C}_a} \in \mathcal{PDB}$ then
        Get $T_{\mathsf{S}_c,\mathsf{C}_a} \leftarrow \mathcal{PDB}$ in terms of $h_{\mathsf{S}_c,\mathsf{C}_a}$
        $\mathrm{sid}^* = \mathsf{GetSID}(T_{\mathsf{S}_c,\mathsf{C}_a})$
        Parse $(\mathsf{S}_d, \mathsf{C}_b) \in \mathrm{sid}^*$
        Retrive $T_{\mathsf{S}_d,\mathsf{C}_b} \in \mathcal{PDB}$ s.t. $\mathsf{GetSID}(T_{\mathsf{S}_c,\mathsf{C}_a}) = \mathrm{sid}^*$
        if $T_{\mathsf{S}_d,\mathsf{C}_b}$ is not found then
            goto IFfound
        else
            Parse $pk^{ots}_{\mathsf{C}_b}, \sigma_{\mathsf{C}_b}, h_{\mathsf{C}_b}, pk^{ots,nx}_{\mathsf{C}_b}, \mathsf{SCache}_{\mathsf{C}_b} \leftarrow T_{\mathsf{S}_d,\mathsf{C}_b}$
            if $\nexists T_{\mathsf{S}_e,\mathsf{C}_b} \in \mathcal{PDB}$ s.t. $\mathsf{S}_e \neq \mathsf{S}_d$ and $pk^{ots}_{\mathsf{C}_b} \in T_{\mathsf{S}_e,\mathsf{C}_b}$ then
                goto IFfound
            if $\mathsf{SChainVfy}(\mathsf{S}_c, pk^{ots}_{\mathsf{C}_a}, \sigma_{\mathsf{C}_b}, h_{\mathsf{C}_a}, pk^{ots,nx}_{\mathsf{C}_a},$
            $\mathsf{SCache}_{\mathsf{C}_a}) = 0$ then
                goto IFfound
return 0
IFfound: return $res$
```

possibilities regarding either the nonce or the randomness used by the OTS key generation. Meanwhile, the security of CRH ensures that the signed hash value is collision-free with non-negligible probability. Eventually, we show that the adversary cannot forge the authentication messages generated by PRF and OTS. The full proof of this theorem will be given in the full version of this paper.

5.2 Security of 4MA

THEOREM 2. *We assume that $\ell_r \leq \ell_s$, both signature schemes* OTS *and* SIG *are SEUF-CMA secure, the hash function* CRH *is collision-resistant, and the size of* ASQueue *has a size z such that $3 \leq z \leq v$. Then the proposed MA-IFD protocol* 4MA *is secure with* $\mathsf{Adv}_{4\mathsf{MA},\mathcal{A}}(z-2, \kappa, \mathsf{MA\text{-}IFD}) \leq \frac{(3u\cdot v\cdot d)^2}{2^{\ell_s-1}} + 2\cdot\mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) + 2u\cdot v\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, 3u\cdot v\cdot d + v\cdot d) + 4u\cdot v\cdot d\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{OTS},\mathcal{F}}(\kappa, 1) + 2v\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, 2u\cdot v\cdot d)$.

We divide adversaries into two categories to analyze the authentication and key exchange respectively:

(1) *Authentication-adversary* can succeed in breaking the MA property, i.e., condition $C1$ holds in the security experiment;
(2) *IFD-adversary* can fail the identity fraud detection, i.e., condition $C2$ holds in the security experiment.

To prove Theorem 2, we present two lemmas. Each analyzes one of the security properties of the proposed protocol. Specifically, Lemma 1 bounds the success probability ϵ_{auth} of authentication-adversaries, and Lemma 2 bounds the success probability ϵ_{ifd} of IFD-adversaries. Then we have $\mathsf{Adv}_{4\mathsf{MA},\mathcal{A}}(z-2, \kappa, \mathsf{MA\text{-}IFD}) \leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{ifd}}$. In the following, we just briefly introduce the idea of the proof.

LEMMA 1. *For any PPT adversary $\mathcal{A}$, the probability that there exists an oracle $\Pi^{s^*}_{\mathsf{C}^*_i}$ accepts with satisfying condition $C1$ is at most* $\epsilon_{\mathsf{auth}} \leq \frac{(3u\cdot v\cdot d)^2}{2^{\ell_s}} + \mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) + u\cdot v\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, 3u\cdot v\cdot d + v\cdot d) + 2u\cdot v\cdot d\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{OTS},\mathcal{F}}(\kappa, 1) + v\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, 2u\cdot v\cdot d)$.

The proof is similar to that of Theorem 1. Here we just present the proof idea for simplicity. The security reduction concerning randomness collision is identical to that in the proof of Theorem 1. The main difference is that there are two types of signature schemes.

LEMMA 2. *For any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ can output an oracle (C^*_i, s^*) satisfying condition $C2$ is at most* $\epsilon_{\mathsf{ifd}} \leq \frac{(3u\cdot v\cdot d)^2}{2^{\ell_s}} + \mathsf{Adv}^{\mathsf{CRH}}_{\mathsf{CRH},\mathcal{H}}(\kappa) + u\cdot v\cdot\mathsf{Adv}^{\mathsf{IND\text{-}CMA}}_{\mathsf{PRF},\mathcal{B}}(\kappa, 3u\cdot v\cdot d + v\cdot d) + 2u\cdot v\cdot d\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{OTS},\mathcal{F}}(\kappa, 1) + v\cdot\mathsf{Adv}^{\mathsf{seuf\text{-}cma}}_{\mathsf{SIG},\mathcal{F}}(\kappa, 2u\cdot v\cdot d)$.

Supposed that $\mathcal{A}$ corrupted a direct authentication server $\mathsf{S}_c \in \mathsf{ASQueue}_{\mathsf{C}_i}$ and deceived the client C_i in a session $\Pi^t_{\mathsf{S}_c}$ with transcript $T^t_{\mathsf{S}_c,\mathsf{C}^*_i}$. Note that each direct authentication server in $\mathsf{ASQueue}_{\mathsf{C}_i}$ has to execute one complete mutual authentication procedure, and before S_c is popped out, its protocol transcript must be verified by at least $z-1$ authentication servers among which at least 1 authentication server S^*_e is honest and uncorrupted by assumption. According to the identity fraud detection algorithm (i.e., Algorithm 2), S^*_e will check: i) Whether $T^t_{\mathsf{S}_c,\mathsf{C}^*_i} \in \mathcal{PDB}$; ii) Whether the session of C^*_i recorded by $T^t_{\mathsf{S}_c,\mathsf{C}^*_i}$ has a valid partner session at every partner of C^*_i in that session. If one of the checks fails, then

S_e^* would report the identity fraud event and block C_i^* and S_c. To make S_e^* accept, the adversary should successfully impersonate the uncorrupted party $\mathcal{ID}_j \in T^t_{\mathsf{S}_c,\mathsf{C}_i^*}$ in order that it can record the $T^t_{\mathsf{S}_c,\mathsf{C}_i^*}$ without being detected. Hence, the identity fraud detection security property is implied by the mutual authentication security property.

The result of Theorem 2 implies the soundness of the identify fraud detection since an honest authentication server S_c can always identify that C_i^* is deceived within the protocol execution between them according to condition $C2$ in Definition 4.

6 PERFORMANCE

General Analysis. Here we assume that the 2MA protocol instances between a client C_a and an authentication server S_c are run over a network with a good quality of service, e.g., industrial network or intranet. In such a network, man-in-the-middle adversaries cannot continuously interfere with the communication between C_a and S_c without getting caught. Before running the 2MA protocol, two parties can diagnose the network via a few hello messages (e.g., via ping). Therefore, the OTS public key of S_c can be normally updated promptly at C_a. As a result, C_a does not need to verify a very long signature-chain in $\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}$. However, the 2MA protocol instances between two servers can run over any network (including the Internet) since long-term keys are used for authentication. And there is no need to cache the OTS update procedure.

The size of SCache determines the performance of the 2MA instances (involving OTS) since we may need to verify the signature-chain cached in it to verify the current signature. However, a network adversary which drops or modifies the network packets may lead to Denial-of-Service (DoS) attacks. Note that if there are such DoS adversaries, no authentication protocol can work at all. Fortunately, the network adversaries can be identified by a party, when it, for example, continuously received a few invalid message authentication codes (MAC) by our design. If the MAC checks fail, then the OTS keys cached in SCache would not grow. In this case, the party can initiate the DoS attacks detection to eliminate it before running the 4MA. To mitigate the DoS attacks, both client and authentication server can also have multiple OTS pubic key pairs as backup. Once a pubic key fails to update in $\xi = |\mathsf{SCache}| = 5$ attempts, then it could trigger a DoS or identity fraud attack alarm, and switch to use the backup OTS keys for recovery. And it is very easy to commit new backup OTS keys when the DoS attacks are removed. In practice, it is, therefore, reasonable to assume that the upper-bound of authentication server's cache $|\mathsf{SCache}_{\mathsf{S}_c,\mathsf{C}_a}|$ is $\xi = 5$, and the upper-bound of client's cache $|\mathsf{SCache}_{\mathsf{C}_a}|$ is $\xi(z-1)$ since it needs to keep the all OTS update procedures with $(z-1)$ unchecked servers. Furthermore, the minimum size of ASQueue is $z = 3$, in which case 4MA can resist 1 compromised server. To resist more compromised servers, it is needed to enlarge the number of z, i.e., deploying more authentication servers.

Further Discussion. Since the DoS tolerant parameter ξ is small (e.g., $1 \le \xi \le 5$ by assumption), to obtain an optimized performance for our protocol, it is sufficient to use a small z (i.e., the number of the server which are chosen by a client for both authentication and identity fraud detection). One could set the parameter z such

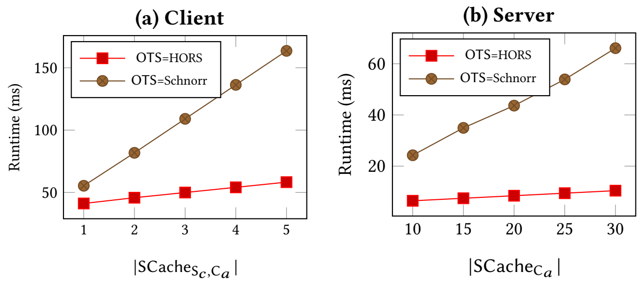

Figure 4: Runtimes in Milliseconds (ms).

that $3 \le z \le 6$. For a cyber-physical system with n authentication servers such that $n > 6$, each client can randomly choose z servers out of n servers to run the protocol.

On the other hand, it is also able to reduce the communication overhead of C_a in an online authentication session by sending the $\mathsf{IFDCache}_{\mathsf{C}_a}$ to the next intended authentication server S_c in the queue $\mathsf{ASQueue}_{\mathsf{C}_a}$ during the idle time of C_a (before doing the online authentication with S_c). Meanwhile, the sent $\mathsf{IFDCache}_{\mathsf{C}_a}$ can be protected by a message authentication code generated based on the pre-shared key $K_{\mathsf{S}_c,\mathsf{C}_a}$. This optimization would roughly reduce the communication cost by a factor of z for online authentication.

Implementation. We implement our system into the firmware layer of an OpenPLC [5] running on a Raspberry Pi 3 to simulate a client, and a laptop with Intel Core i7-8750H is used to simulate the server in our system. OpenPLC implements the firmware layer of a real PLC, which allows users to implement control logic on top of it. Most importantly, OpenPLC is a fully open-source implementation of a PLC, so we are able to integrate our authentication methods into its firmware. Here we first consider an instantiation instantiate the OTS scheme with the one called HORS [31] To implement the OTS, we consider two concrete instantiations. We first instantiate the OTS scheme by the one called HORS [31] which has an optimized computational cost. For simplicity, we realized both the hash function and the one-way function (required by HORS) with SHA-256. In the implementation of HORS, the signing key consists of a set of random values $\{s_i\}_{i\in[t]}$ where $|s_i| = 80$ and $t = 256$ (which equals to the output length of the SHA-256). Each signature of HORS is a subset of the singing key $\{s_{i_1}, s_{i_2}, \dots, s_{i_k}\}$ where each index i_j (for $1 \le j \le k$) is a $\log_2^t$ bits substring of the hash value of a message (see more details in [31]) and $k = 32$ in our setting. To obtain better memory and communication costs, we also choose to use Schnorr signature [33] to instantiate both the OTS and the long-term key-based signature (required by 4MA). We implemented the Schnorr signature based on the MNT224 ecliptic curve. The PRF is realized by HMAC [22]. We implemented our protocol based on Miracl library [28]. The runtime of 4MA is shown in Figure 4 based on HORS and Schnorr, respectively.

To calculate the communication and the memory overheads, the lengths of identity, timestamp, and random nonce are considered as 32bits, 64bits, and 256bits, respectively. Since the memory and communication cost is close, we roughly depict them together for simplicity. And we fix $\xi = 5$ in this benchmark but change z from 3 to 8. We roughly show the memory and the communication costs in Figure 5.

Protocol	Authentication Factors	Security Properties					Communication Pass/KiloBytes	Computation Cost
		S-Auth	M-Auth	IA-SKS	IFD	SH		
He et al. [17]	PW+LSK	√	×	√	×	×	2/700	21H+4MUL
Challa et al. [9]	Bio+PW+LPK	√	×	√	×	×	2/420	1Fe+14Mul+12h
Yang et al. [37]	PW+LPK	√	√	√	×	×	3/944	59h
4MA (HORS)	LSK+OTS	√	√	×	√	√	$4/(13.4 + 10.7z\xi + 5.4z)$	$(\xi(k + 33z - 33) + t + 266)$H +3MUL
4MA (Schnorr)	LSK+OTS	√	√	×	√	√	$4/(1.7 + 1.4z\xi + 0.7z)$	$(10 + z\xi)$H $+(2z\xi + 7)$MUL

Table 2: Comparison

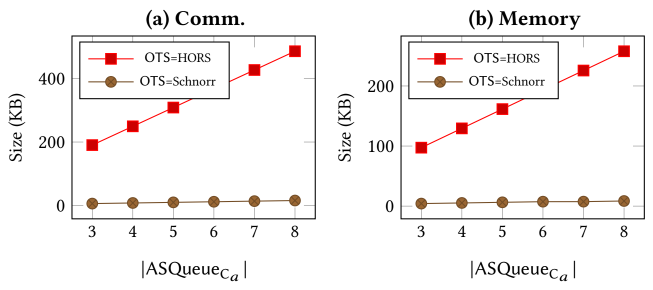

Figure 5: Communication and Memory in KiloByte (KB).

Comparison. Here we briefly compare our proposed schemes with some existing lightweight delegated authentication protocols, including He et al. [17], Challa et al. [9] and Yang et al.[37]. For comparison, We consider the following perspectives: (i) authentication factors, (ii) main security properties, (iii) number of communication passes, and (iv) computation cost. Furthermore, we let 'FE' denote a fuzzy extractor operation to obtain a secret from biometrics. We let 'S-Auth' denote single-side explicit authentication, 'M-Auth' denote mutual authentication, 'IA-SKS' denotes implicit authentication with session key security, 'IFD' denotes the server's identity fraud detection, and 'BS' denote the backward security. Our protocol can provide backward security since the OTS can evolve to a new one if the compromised key is not used by the adversary.

To calculate the communication overhead, the lengths of identity, timestamp, random nonce, hash value and ECC group value are considered as 32bits, 64bits, 256bits, 256bits and 224bits, respectively. To compare the computation cost, we let 'H' be hash function operation and 'MUL' be an ECC multiplication. We denote 'Bio' to be the biometric authentication factor, 'PW' to be password, 'LSK' denote long-term symmetric key, and 'LPK' denote the long-term public key. The computation and communication costs of 4MA are then calculated based on HORS OTS [31]. As there are many different types of entities in these protocols, we only show the computation cost of all entities for readability and simplicity.

The detailed comparison is shown in Table 2: it shows that our protocol 4MA are less efficient than the previous delegated authentication protocols. But the overall computation cost (as shown in Figure 4) of 4MA is still practical for constrained devices. Although the third message sent in our protocol includes a large cache IFDCache (which dominates the communication cost), it can be transmitted via a high-speed local network (e.g., 1000Mb Ethernet) in cyber-physical systems.

Summary. The computational cost of 4MA is with ξ (i.e., the DoS tolerant parameter). The client's performance is still practical for our recommended parameters. The communication and the memory costs are linear with both parameters z (i.e., the number of authentication servers used to do IFD) and ξ. For $3 \leq z \leq 8$ and $1 \leq \xi \leq 5$, those costs are acceptable for modern PLCs, which usually have 1MB to 4MB user memory [6, 36]. To sum up, 4MA instantiating with HORS is much more efficient than 4MA instantiating with Schnorr. But the latter one has much fewer communication and memory overheads.

Although the third message sent in our protocol includes a large cache IFDCache (which dominates both the communication and the memory costs), it can be transmitted via a high-speed local network (e.g., 1000Mb Ethernet) in cyber-physical systems. As our protocol 4MA is generic, one can instantiate it with different concrete algorithms (regarding those building blocks) to obtain either better performance or more security properties (e.g., post-quantum security).

7 RELATED WORK

Nali and Oorschot [30] previously studied identity fraud detection by using a single sequentially evolved one-time symmetric authentication key, i.e., a new authentication key is generated solely based on the last key. To deal with the synchronization issue, Yang and Guo proposed a dynamic authentication credential (DAC) framework (hereafter referred to as the YG scheme). However, the YG scheme requires a PKI-based cryptographic building block to generate the ephemeral secret seed for updating DAC, so it is likely to be computationally expensive and is therefore unfit for power-constrained devices. Besides, the YG scheme is designed for two-party secure communication with any authentication delegator, so it cannot be applied to the three-party case.

In [37], Yang *et al.* proposed a lightweight delegated authenticated key agreement (AKA) protocol for wireless sensor networks. It is based on a new DAC framework, which is extended from the YG scheme. One of the features of using DAC in this scheme is to detect previous impersonation attacks on users/sensors by comparing their current DAC with the corresponding ones stored at the gateway node. However, the symmetric DAC cannot resist the server breach.

Recently, Milner *et al.* [27] developed the foundations and proposed several constructions for security protocols that can automatically detect, if a secret (such as a key or password) has been misused. For key misuse detection, their constructions mainly on an evolving public/secret key scenario that puts an emphasis on a two-party authentication case. However, when considering a specific lightweight public key cryptographic scenario, e.g., one-time signature, new network threats would be sprung up (as we mentioned in Section 1). Moreover, unlike ours, Milner et al.'s work did not consider the key synchronization problem during the key evolving procedure. Nevertheless, it is non-trivial to extend their construction idea to the lightweight delegated authentication that may involve symmetric key (for efficiency reasons) and more protocol participants.

Another important research direction on the mitigation of server breach is the threshold authentication. Agrawal et al. [2] introduced a generic password-based threshold token-based authentication framework (called it as PASTA) which allows n servers to serve as identity provider. In PASTA, any $t \leq n$ servers can collaboratively verify passwords and generate tokens, while no $t-1$ servers can forge a valid token and launch online dictionary attacks. PASTA leverages on threshold oblivious pseudo-random function (TOPRF) [15] as its main building block. However, like other threshold cryptosystems (e.g., [8, 12, 13, 32]), the concrete instances of TOPRF still requires public key cryptographic operations, which are not suitable for resource-constrained devices in CPS.

We stress that all the works mentioned above do not consider the identity fraud issues caused by the compromised server in lightweight delegated authentication protocols. In this work, we focus on the formulation of a security model and new constructions for lightweight delegated authentication with identity fraud detection.

8 CONCLUSION

We have proposed the first lightweight key management framework and a delegated authentication protocol with identity fraud detection (IFD). We have shown that the proposed protocol can be applied to cyber-physical systems since the scheme is based on lightweight cryptographic primitives. We have also presented the formal security result of the proposed scheme under a new security model which formulates mutual authentication and server's identity fraud detection simultaneously. We leave the construction of an efficient lightweight authentication protocol with more security features such as session key security as our future work. For example, one could use our authentication protocol as a compiler to build authenticated key exchange protocols by following [25].

ACKNOWLEDGMENTS

Jianting Ning is supported by the Natural Science Foundation of China under Grant No.61872051 and 61972094. This work is also supported by the SUTD start-up research under Grant No.: SRG-ISTD-2017-124.

REFERENCES

[1] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. 2005. *Password-Based Authenticated Key Exchange in the Three-Party Setting*. Springer Berlin Heidelberg, 65–84.

[2] Shashank Agrawal, Peihan Miao, Payman Mohassel, and Pratyay Mukherjee. 2018. PASTA: PASsword-based Threshold Authentication. In *CCS*. ACM, 2042–2059.

[3] Chuadhry Mujeeb Ahmed, Carlos Murguia, and Justin Ruths. 2017. Model-based Attack Detection Scheme for Smart Water Distribution Networks. In *AsiaCCS*. ACM, 101–113.

[4] Rifaqat Ali, Arup Kumar Pal, Saru Kumari, Marimuthu Karuppiah, and Mauro Conti. 2017. A secure user authentication and key-agreement scheme using wireless sensor networks for agriculture monitoring. *Future Generation Computer Systems* (2017).

[5] Thiago Rodrigues Alves. [n.d.]. OpenPLC: Getting Started on Raspberry Pi. https://goo.gl/CgFXWz [Accessed Jan., 2018].

[6] Rockwell Automation. [n.d.]. ComplactLogic System Selection Guide. https://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1769-sg001_-en-p.pdf [Accessed Aug., 2019].

[7] Mihir Bellare and Phillip Rogaway. 1995. Provably secure session key distribution: the three party case. In *STOC*. 57–66.

[8] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. 2018. Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. In *CRYPTO (1)*. Springer, 565–596.

[9] Sravani Challa, Mohammad Wazid, Ashok Kumar Das, Neeraj Kumar, Goutham Reddy Alavalapati, Eun-Jun Yoon, and Kee-Young Yoo. 2017. Secure Signature-Based Authenticated Key Establishment Scheme for Future IoT Applications. *IEEE Access* 5 (2017), 3028–3043.

[10] Chin-Chen Chang, Wei-Yuan Hsueh, and Ting-Fang Cheng. 2016. A Dynamic User Authentication and Key Agreement Scheme for Heterogeneous Wireless Sensor Networks. *Wireless Personal Communications* 89, 2 (2016), 447–465.

[11] Liqun Chen, Hoon Wei Lim, and Guomin Yang. 2014. Cross-Domain Password-Based Authenticated Key Exchange Revisited. *ACM Trans. Inf. Syst. Secur.* 16, 4 (2014), 15:1–15:32.

[12] Ivan Damgård and Maciej Koprowski. 2001. Practical Threshold RSA Signatures without a Trusted Dealer. In *EUROCRYPT*. Springer, 152–165.

[13] Yvo Desmedt and Yair Frankel. 1989. Threshold Cryptosystems. In *CRYPTO*. Springer, 307–315.

[14] Nicolas Falliere, Liam O Murchu, and Eric Chien. 2011. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response* 5, 6 (2011), 29.

[15] Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. 2005. Keyword Search and Oblivious Pseudorandom Functions. In *TCC*. Springer, 303–324.

[16] Oded Goldreich. 2006. *Foundations of Cryptography: Volume 1*. Cambridge University Press, New York, NY, USA.

[17] Jun He, Zheng Yang, Jianxun Zhang, Wanping Liu, and Chao Liu. 2018. On the security of a provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. *Int. J. Distrib. Sen. Netw.* 14, 1 (Jan 2018), 1–8.

[18] Andreas Hülsing, Christoph Busold, and Johannes A. Buchmann. 2012. Forward Secure Signatures on Smart Cards. In *SAC*. Springer, 66–80.

[19] Qi Jiang, Sherali Zeadally, Jianfeng Ma, and Debiao He. 2017. Lightweight Three-Factor Authentication and Key Agreement Protocol for Internet-Integrated Wireless Sensor Networks. *IEEE Access* 5 (2017), 3376–3392.

[20] Chenglu Jin, Xiaolin Xu, Wayne P. Burleson, Ulrich Rührmair, and Marten van Dijk. 2015. PLayPUF: Programmable Logically Erasable PUFs for Forward and Backward Secure Key Management. *IACR ePrint* 2015 (2015), 1052.

[21] Eduard Kovacs. [n.d.]. Attackers Alter Water Treatment Systems in Utility Hack: Report. https://www.securityweek.com/attackers-alter-water-treatment-systems-utility-hack-report.

[22] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. 1997. HMAC: Keyed-Hashing for Message Authentication. *RFC* (1997), 1–11.

[23] Xiong Li, Jianwei Niu, Md. Zakirul Alam Bhuiyan, Fan Wu, Marimuthu Karuppiah, and Saru Kumari. 2018. A Robust ECC-Based Provable Secure Authentication Protocol With Privacy Preserving for Industrial Internet of Things. *IEEE Trans. Industrial Informatics* 14, 8 (2018), 3599–3609.

[24] Xiong Li, Jieyao Peng, Jianwei Niu, Fan Wu, Junguo Liao, and Kim-Kwang Raymond Choo. 2018. A Robust and Energy Efficient Authentication Protocol for Industrial Internet of Things. *IEEE IoT Journal* 5, 3 (2018), 1606–1615.

[25] Yong Li, Sven Schäge, Zheng Yang, Christoph Bader, and Jörg Schwenk. 2014. New Modular Compilers for Authenticated Key Exchange. In *ACNS*. Springer, 1–18.

[26] Yong Li, Sven Schäge, Zheng Yang, Florian Kohlar, and Jörg Schwenk. 2014. On the Security of the Pre-shared Key Ciphersuites of TLS. In *PKC*. 669–684.

[27] Kevin Milner, Cas Cremers, Jiangshan Yu, and Mark Ryan. 2017. Automatically Detecting the Misuse of Secrets: Foundations, Design Principles, and Applications. In *CSF*. IEEE Computer Society, 203–216.

[28] Miracl 2018. MIRACL Cryptographic Library. https://bit.ly/2MltKVG

[29] Carlos Murguia and Justin Ruths. 2016. Characterization of a CUSUM model-based sensor attack detector. In *CDC*. IEEE, 1303–1309.

[30] D. Nali and Paul C. van Oorschot. 2008. CROO: A Universal Infrastructure and Protocol to Detect Identity Fraud. In *ESORICS*. 130–145.

[31] Leonid Reyzin and Natan Reyzin. 2002. Better than BiBa: Short One-Time Signatures with Fast Signing and Verifying. In *ACISP*. Springer, 144–153.

[32] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. 1994. How to share a function securely. In *STOC*. ACM, 522–533.

[33] Claus-Peter Schnorr. 1989. Efficient Identification and Signatures for Smart Cards. In *CRYPTO*. Springer, 239–252.

[34] Information security — Key management — Part 7: Cross-domain password-based authenticated key exchange. 2020. ISO/IEC DIS 11770-7.

[35] Victor Shoup. 2004. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332. http://eprint.iacr.org/.

[36] SIEMENS. [n.d.]. S7-1500/S7-1500F Technical Data. https://w3.siemens.com/mcms/programmable-logic-controller/en/advanced-controller/s7-1500/cpu/Documents/s7-1500_techn_data_cpu_en.pdf [Accessed Aug., 2019].

[37] Zheng Yang, Junyu Lai, Yingbing Sun, and Jianying Zhou. 2019. A Novel Authenticated Key Agreement Protocol With Dynamic Credential for WSNs. *ACM Trans. Sen. Netw.* 15, 2, Article 22 (Feb. 2019), 27 pages.

[38] Liehuang Zhu, Cong Guo, Zijian Zhang, Wei Fu, and Rixin Xu. 2017. A Novel Contributory Cross-Domain Group Password-Based Authenticated Key Exchange Protocol with Adaptive Security. In *DSC*. IEEE Computer Society, 213–222.