# HMACCE: Establishing Authenticated and Confidential Channel from Historical Data for Industrial Internet of Things

Chenglu Jin, Zheng Yang, Tao Xiang, Sridhar Adepu and Jianying Zhou

**Abstract**—Industrial Internet of Things (IIoT) is a new paradigm for building intelligent industrial control systems, and how to establish a secure channel in IIoT for machine-to-machine (M2M) communication is a critical problem because the devices in IIoT suffer from various attacks and may leak confidential information. Traditional authenticated and confidential channel establishment (ACCE) protocols neither apply for resource-constrained IIoT devices nor satisfy leakage resilience. In this paper, we introduce a new security notion: historical data based multi-factor ACCE (HMACCE) to address this issue and propose two HMACCE protocols. Our HMACCE protocols use three authentication factors, i.e., a symmetric secret key, historical data, and a set of secret tags associated with the historical data, to establish a secure communication channel between the client and the server. The key idea is to use the secret key managed by an IIoT edge device to quickly verify the relationship between the historical data and its associated tags stored on the server. Our HMACCE has the following remarkable features. First, it is lightweight and tailored for resource-constrained IIoT devices. Second, it is *bounded historical tag leakage resilience*, which means that if a small portion of the secret tags is leaked to an adversary, it will not affect its security with an overwhelming probability. Moreover, as a security enhancement service, our HMACCE can be easily integrated with legacy IIoT devices by running simple authenticated key exchange protocols.

**Index Terms**—Industrial IoT, Historical Data, Authentication, Authenticated Confidential Channel Establishment, Multi-Factor Authentication, Security Enhancement Service.

## 1 Introduction

Industrial Internet of Things (IIoT) is a variant of Internet-of-Things (IoT), which enables internet connectivity and communication between internet-enabled devices and systems. That is, IIoT mainly aims to enable intelligence in traditional Cyber-Physical Systems (CPSs), like water treatment systems and power grids. These systems are critical for the daily life of millions of people. However, the security of this kind of system is always an afterthought, which opens a tremendous attacking surface on CPSs for malicious adversaries [1]. Even worse, many legacy devices with very limited or no security protection are still in use. Since they have been running for decades, it becomes a non-trivial task to upgrade or replace them. Therefore, security enhancements of legacy devices are highly demanded in practice now. As the first step towards a secure system, we need to protect the communication between the devices in the field and the servers/control centers because most of the devices are required to report their status and data acquired in the field to the server, and they accept commands from the server. In the context of CPS, this kind of server is usually called supervisory control and data acquisition (SCADA) system.

In the existing literature, many end-to-end encryption and message authentication methods were suggested between controllers and SCADA system [2], [3], [4], but none of them answered the question about how to establish such a secure communication channel. Of course, one can simply use single factor authenticated key exchange (AKE) protocols [5], [6] or authenticated and confidential channel establishment (ACCE) protocols [7], [8], but *can we enhance their security by introducing additional authentication factors?* [1] Because it is a machine-to-machine (M2M) authentication, the existing two/multi-factor authenticated key exchange (AKE) protocols [9], [10], [11], [12], [13], which usually use passwords or fingerprints as the second factor, do not apply here. Multi-factor M2M AKE might be instantiated from the generic framework [14] by Fleischhacker et al., which allows one to build a protocol by securely mixing multiple types and quantities of authentication factors such as low-entropy (one-time) passwords/PINs, high-entropy private/public keys and biometric factors. However, their framework does not cover the authentication factors that are lightweight

*Chenglu Jin and Zheng Yang contribute equally and share the first authorship. They are sorted alphabetically in the author list. The Southwest University is the first affiliation to finish this work.*

- *C. Jin is with CWI Amsterdam, Science Park 123 1098 XG Amsterdam, Netherlands. E-mail: chenglu.jin@cwi.nl*
- *Z.Yang is with Southwest University, No.2 Tiansheng Road Beibei District,Chongqing 400715,P.R.China; Email: youngzheng@swu.edu.cn*
- *Tao Xiang is with Chongqing University, No.174 Shazhengjie, Shapingba, Chongqing, 400044, China. E-mail: txiang@cqu.edu.cn. Tao Xiang is the corresponding author.*
- *S. Adepu is with the University of Bristol, Beacon House, Queens Road, Bristol, BS8 1QU, UK. E-mail:sridhar.adepu@bristol.ac.uk*
- *J. Zhou is with the iTrust, Singapore University of Technology and Design, 8 Somapah Rd, Singapore, 487372. E-mail:jianying_zhou@sutd.edu.sg*

[1] Note that an ACCE protocol can be built from an AKE protocol and an authenticated encryption scheme, so one can consider AKE as one of the most important building blocks of ACCE. However, the security notion of ACCE does not directly imply that of AKE. We refer readers to Section 2 and [7] for more detail about ACCE.

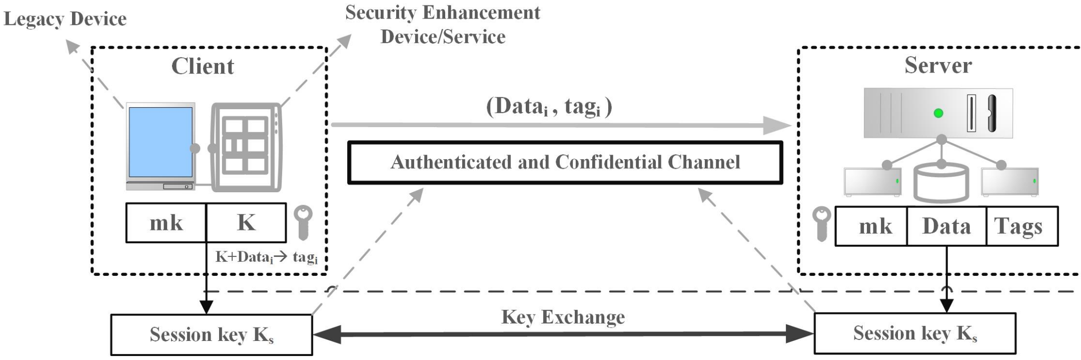


Fig. 1: Overview of Our HMACCE Protocol.

while being able to satisfy partial leakage resilience. In general, the servers in geographically concentrated cyber-physical systems, like manufacturing systems and nuclear plants (as in the instance of Stuxnet [1]), are more vulnerable than the individual devices/controllers, because they are more likely to be accessible via the Internet by remote attackers. Hence, we want to find a new authentication factor on the server side with a stronger security level than a conventional secret key stored even on the same machine.

Recall that CPS devices keep sending data to the SCADA system for monitoring. Actually, for data analysis, the historical data in most of the SCADA systems is collected and stored in a dedicated process historian instead of their main servers [15]. This directly implies that the historical data has a different security level from secret keys. Moreover, a secret key, usually hundreds of bits, can be leaked very fast in a security breach, but a large database on the same server will clearly at least slow down the secret leakage by a few orders of magnitude and consequently implies a different security level. Therefore, a secret key, a database of historical data stored in a historian, and a database of data associated tags stored on a SCADA server are the perfect authentication factors with three different security levels, such that compromising one factor does not lead to a corruption of another authentication factor in practice. As another fact, the historical data and its tag (discussed later) are growing all the time, so a piece of historical data leaked in the past may not be valid as an authentication factor soon after. This makes a successful impersonation even harder.

**Existing Historical Data based Authentication Protocols.** The concept of using historical data as an authentication factor was introduced in [16] and further developed in [17] at ESORICS'16. The early scheme [16] uses the historic data straightforwardly as a symmetric key shared between the client and the server. This imposes a non-trivial storage overhead to the client, which is sometimes infeasible for a resource-constrained CPS device. Recently Chan et al. [17] introduced a scalable historical data based two-factor authentication scheme (which will be referred to as the CWZT scheme). Namely, the first authentication factor is a long-term symmetric key, and the second authentication factor is a dynamically growing set of secret tags associated with historical data. The CWZT protocol is wisely derived from the proof of retrievability (PoR) protocol [18], in which the server authenticates itself to the client by proving that it possesses all historical data sent by the client. As one of their major contributions, the CWZT protocol only requires the client to store a small constant-sized secret (e.g., $512$ bits), which well fits CPS devices. Chan et al. also introduced historical tag leakage resilience in a bounded-storage model [19], [20] as its security feature so that partial historical tag leakage does not affect much of its security.

**Vulnerabilities of the CWZT Protocols.** (1) According to our analysis in Section 4, the CWZT protocol is vulnerable to a tag stealing attack. In short, we show that an adversary can steal *all* the historical tags through legitimate interactions with the server, given only *one piece of historical tag* (associated with one data piece) that is somehow leaked. Note that the *partial historical tag leakage* is allowed in the adversarial model of the CWZT protocol and was claimed as one of the major contributions in [17]. (2) In [17], the authors suggested using the first authentication factor to protect the transmission of the second authentication factor (tags). This completely deviates from the motivation of having two authentication factors. Thus how to secure the transmission of data and tags from the client to the server based on a leakage resilient multi-factor session key establishment procedure is still an open problem.

**Our Contributions.** Since we need a secure channel to transmit the authentication factors, we formalize our constructions by a new security notion, historical data based multi-factor ACCE (HMACCE). Due to the vulnerabilities and limitations of the existing authentication protocols mentioned above, we cannot simply extend the existing authentication protocols to an authenticated and confidential channel establishment (ACCE) protocol. We have to reconsider the fundamental authentication problem based on historical data and redesign a new ACCE protocol from scratch. More specifically, we made four significant contributions as follows:

1) We analyze the state-of-the-art historical data based authentication protocol (the CWZT protocol [17] proposed at ESORICS'16) and propose a tag-stealing attack that breaks the security claim of the CWZT protocol via legitimate interactions.
2) To build a solid theoretical foundation of our proposed HMACCE protocols, we are the first to formally define two indistinguishability-based security models for HMACCE, and later we analyze our proposed protocols in these security models.

3) As one of the main contributions of this paper, we introduce two HMACCE protocols $\Pi_{\mathsf{woFS}}$ (without forward secrecy) and $\Pi_{\mathsf{FS}}$ (with forward secrecy) and proved their security in the random oracle model.
4) To show the impact of our protocols in the real world, we demonstrate how our protocols can be deployed in the field to enhance the security services of legacy devices. Also, we implemented $\Pi_{\mathsf{woFS}}$ and $\Pi_{\mathsf{FS}}$, and evaluated their performance experimentally.

**Technical Overview.** An overview of our first HMACCE protocol $\Pi_{\mathsf{woFS}}$ is presented in Figure 1. The client device and the server share a master key ($mk$) as their first authentication factor. When the client sends data to the server, it generates a secret tag associated with the data using a tag generation key $K$. The server stores all tuples $\{(Data_i, tag_i)\}$ separately as its second and third authentication factors, respectively, while the client only needs to store $K$ as its second authentication factor. The client only has two authentication factors due to its limited storage space, i.e., not storing any historical data. In our HMACCE protocols, both parties can use their authentication credentials to run the key exchange procedure to generate a session key for a secure channel that protects the underlying data and tag transmission.

In addition, $\Pi_{\mathsf{woFS}}$ has a remarkable security feature called historical tag leakage resilience, such that a small portion of tag leakage will not affect the security of $\Pi_{\mathsf{woFS}}$ much. Notice that although this feature was first introduced in [17], they failed to achieve it due to the tag stealing attack we will introduce in Section 4. Also, because of the clear separation of the two authentication factors in $\Pi_{\mathsf{woFS}}$, $\Pi_{\mathsf{woFS}}$ can be easily used to enhance the security of legacy CPS devices. An additional device with the second factor can be attached to a legacy device (with the first factor embedded), intercept its traffic, and complete most of the computation in $\Pi_{\mathsf{woFS}}$.

One limitation of $\Pi_{\mathsf{woFS}}$ is that it can only defend against *static* bounded-leakage regarding the historical tags, and it does not provide perfect forward secrecy. In a static bounded-leakage model, the adversary can only learn a fraction of the secret tags *at the beginning of the security game*. Nevertheless, the static bounded-leakage resilience is still valuable and useful for HMACCE in practice since the leaked tags will be outdated quickly when the historical data is growing. Theoretically, an attacker may try to adaptively attack many sessions as formulated in the seminal work about entity authentication model [21]. To achieve this adaptive bounded-leakage resilience and perfect forward secrecy, we design the second HMACCE protocol $\Pi_{\mathsf{FS}}$. In $\Pi_{\mathsf{FS}}$, we use the first protocol $\Pi_{\mathsf{woFS}}$ as a compiler to transform any passively secure two-message key exchange (TKE) protocols to be actively secure HMACCE protocols. Because the session key does not depend on the authentication keys (unlike $\Pi_{\mathsf{woFS}}$), $\Pi_{\mathsf{FS}}$ can resist adaptive bounded-leakage, i.e., the adversary can get access to a bounded number of valid historical tags at any time in the security experiment.

## 2 Related Work

**Lightweight AKE Protocols.** Due to the limitations of power constrained devices, e.g., sensor networks or IoT devices, researchers have been dedicated to developing lightweight multi-factor AKE protocols in conjunction with specific communication models or application scenarios. For example, the lightweight multi-factor AKE protocols proposed in [9], [13], [22] are designed for wireless sensor networks (WSNs), and there are many protocols [11], [23] for Internet-of-Things (IoT). In [24], Chattaraj et al. proposed an AKE protocol for cloud computing services. For different application scenarios and computation power of players, different authentication factors might be involved. The commonly used authentication factors are long-term symmetric keys and users' passwords. To enhance its security, a protocol might also incorporate biometric factors [9], [11] with more entropy than a password into authentication. However, none of the above lightweight AKE protocols cover the leakage resilient property in our proposals. Recently, Haase and Labrique [25] proposed a verifier-based password-authenticated key-exchange (V-PAKE) called AuCPace, which can tolerate public-key-infrastructure (PKI) failures while using in IIoT applications. However, this scheme is not a pure M2M AKE scheme since it involves human authentication. Moreover, this scheme does not support the leakage resilience and legacy compliance properties which are addressed in our scheme. In short, none of the above lightweight AKE protocols can realize the leakage resilience property.

**Comparison between ACCE and AKE.** Unlike AKE protocols satisfying session key indistinguishability property, a secure ACCE (Authenticated and Confidential Channel Establishment) protocol guarantees indistinguishability between different messages transmitted in the channel [7], [8]. Informally speaking, an ACCE channel can guarantee that data transmitted over this channel is confidential (indistinguishable), and the channel also preserves the authenticity and integrity of the messages. Therefore, a sequence of messages read from this channel corresponds exactly to the sequence of messages sent by the honest and legitimate sender, and these messages are not known to attackers. Also, ACCE and AKE are highly related since ACCE protocols can be constructed by a stateful length-hiding authenticated encryption and an authenticated key exchange protocol.

**Cryptographic Primitives for Perfect Forward Secrecy (PFS).** Considering the importance of PFS, many AKE schemes are proposed with PFS based on Diffie-Hellman key exchange (DHKE), e.g., [9], [11], [26], [27], [13]. Fortunately, some results (e.g., [28], [29]) have shown that DHKE protocols are feasible to be realized with the elliptic curves cryptography (ECC) optimized for embedded systems. We also instantiate our protocol $\Pi_{\mathsf{FS}}$ with ECC based DHKE protocol for comparison.

**Generic AKE Compilers.** A research line related to our second protocol $\Pi_{\mathsf{FS}}$ is the AKE compiler that securely combines authentication protocols (AP) with passively secure key exchange protocols (KE) in a modular and generic manner, e.g., [30], [31], [14], [32]. However, no existing AKE compilers leverage historical data based authentication protocol as a building block. Our protocol $\Pi_{\mathsf{FS}}$ presents a new way to realize ACCE compilers.

# 3 Preliminaries

**General Notations.** Let $\kappa \in \mathbb{N}$ denote the security parameter and $1^\kappa$ be a string of $\kappa$ ones. We let $[n] = \{1, \ldots, n\} \subset \mathbb{N}$ denote the set of integers between 1 and $n$. We write $a \overset{\$}{\leftarrow} S$ to denote the operation sampling a uniformly random element from a set $S$. We let $\|$ denote the concatenation (operation) of two strings.

**Random Oracles.** Bellare and Rogaway [33] first used the random oracle as a tool to prove the security of cryptographic schemes. In this paper, we assume that the hash function $h(\cdot)$ is modeled as a random oracle. A random oracle is stateful. Namely, on input of a value $m \in \{0,1\}^*$, the random oracle query $h(m)$ proceeds as follows: (i) With respect to the first query on $m$, the oracle returns a true random value $r_m$ from the output space, and records the tuple $(m, r_m)$ into its query list $\mathsf{HL}$; (ii) If $m \in \mathsf{HL}$, then the oracle returns its associated random value $r_m$ recorded in $\mathsf{HL}$. As in [34], we use a uniformly random salt $\chi \overset{\$}{\leftarrow} \mathcal{X}$ as input of $h$ to sample a random oracle $h(\chi, \cdot)$, where $\mathcal{X}$ is the salt space. When the salt is clear in the context, we may write $h(\cdot)$ instead of $h(\chi, \cdot)$ for simplicity. The random salt can be used to prevent vulnerabilities introduced in [34].

**Stateful Length-Hiding Authenticated Encryption.** We review the stateful length hiding authenticated encryption (SLHAE) security that is originally defined by Paterson *et al.* [35] and also used by Jager *et al.* [7], [8] for their analysis of TLS 1.2.

Here a *stateful symmetric encryption scheme* consists of two algorithms $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$. Algorithm $(C, st'_e) \overset{\$}{\leftarrow} \mathsf{StE.Enc}(k, \ell_c, H, m, st_e)$ takes as input a secret key $k$ from the key space $\mathcal{K}_{\mathsf{SLHAE}}$ (i.e., $k \in \mathcal{K}_{\mathsf{SLHAE}}$), an output ciphertext length $\ell_c \in \mathbb{N}$, some header data $H \in \{0,1\}^*$, a plaintext $m$ from the message space $\mathcal{M}_{\mathsf{SLHAE}}$ (i.e., $m \in \mathcal{M}_{\mathsf{SLHAE}}$), and the current state $st_e \in \{0,1\}^*$, and outputs either a ciphertext $C \in \{0,1\}^{\ell_c}$ and an updated state $st'_e$ or an error symbol $\perp$ if for instance the output length $\ell_c$ is not valid for the message $m$.

Algorithm $(m', st'_d) = \mathsf{StE.Dec}(k, H, C, st_d)$ takes as input a key $k$, header data $H$, a ciphertext $C$, and the current state $st_d \in \{0,1\}^*$, and returns an updated state $st'_d$ and a value $m'$ which is either the message encrypted in $C$, or a distinguished error symbol $\perp$ indicating that $C$ is not a valid ciphertext. Both encryption state $st_e$ and decryption state $st_d$ are initialized to an empty string $\emptyset$. Algorithm $\mathsf{StE.Enc}$ may be probabilistic, while $\mathsf{StE.Dec}$ is always deterministic. One could refer to [35] for more details.

The security of a SLHAE is formally defined in Appendix A. Informally, SLHAE should satisfy both the confidentiality and integrity requirements of a communication channel.

**Passively Secure Two-message Key Exchange.** We consider a two-message key exchange ($\mathsf{TKE}$) protocol in which the session key is established within only two protocol passes. In each protocol pass, a single message is sent by a party. We further assume that each protocol player does not hold any long-term secret key for simplicity. Specifically, a general $\mathsf{TKE}$ protocol may consist of three polynomial time algorithms ($\mathsf{TKE.Setup}, \mathsf{TKE.MSG}, \mathsf{TKE.SKG}$) which are defined as follows:

- $pms \leftarrow \mathsf{TKE.Setup}(1^\kappa)$: On input $1^\kappa$, the setup algorithm outputs $pms$, a set of system parameters. We assume the other algorithms may implicitly use $pms$.
- $m_{\mathsf{id}_1} \overset{\$}{\leftarrow} \mathsf{TKE.MSG}(\mathsf{id}_1, r_{\mathsf{id}_1}, m_{\mathsf{id}_2})$: The message generation algorithm takes as input a party's identity $\mathsf{id}_1$, a randomness $r_{\mathsf{id}_1} \overset{\$}{\leftarrow} \mathcal{R}_{\mathsf{TKE}}$ and a message $m_{\mathsf{id}_2} \in \mathcal{M}_{\mathsf{TKE}}$ received from party $\mathsf{id}_2$, and outputs a message $m_{\mathsf{id}_1} \in \mathcal{M}_{\mathsf{TKE}}$ to be sent, where $\mathcal{R}_{\mathsf{TKE}}$ is the randomness space and $\mathcal{M}_{\mathsf{TKE}}$ is the message space. Note that if $\mathsf{id}_1$ is the sender then $m_{\mathsf{id}_2} = \emptyset$.
- $K \leftarrow \mathsf{TKE.SKG}(\mathsf{id}_1, r_{\mathsf{id}_1}, \mathsf{id}_2, m_{\mathsf{id}_2})$: The session key generation algorithm takes as input the participants' identities $\mathsf{id}_1$ and $\mathsf{id}_2$, the randomness $r_{\mathsf{id}_1}$ and the received message $m_{\mathsf{id}_2}$ from party $\mathsf{id}_2$, and outputs a session key $K \in \mathcal{K}_{\mathsf{TKE}}$, where $\mathcal{K}_{\mathsf{TKE}}$ is the session key space.

A $\mathsf{TKE}$ is secure under passive attacks, meaning an adversary cannot distinguish an established secret key from a random string just by passively observing the communication between the two parties. A formal definition of the correctness and security of $\mathsf{TKE}$ can be found in Appendix A.

# 4 Cryptanalysis of the CWZT Scheme

In this section, we revisit the security property of CWZT scheme [17, §5.1] regarding the resilience to the leakage of historical tags. We will introduce an attack to subvert the leakage resilience of CWZT scheme. Note that leakage resilience is an intrinsic property that distinguishes historical data relevant authentication factors from other symmetric key based authentication factors.

## 4.1 Protocol Review

We first briefly review the CWZT scheme. Let $\mathbb{Z}_p$ be an abelian group with prime order $p$ that has $\kappa$ bits. The CWZT protocol makes use of two pseudorandom functions $f : \{0,1\}^\kappa \times \{0,1\}^* \rightarrow \mathbb{Z}_p$ and $E : \{0,1\}^\kappa \times \{0,1\}^\kappa \rightarrow \{0,1\}^\kappa$, and a cryptographic hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$. The protocol running between a verifier $\mathsf{id_C}$ and a prover $\mathsf{id_S}$ is shown in Figure 2.

## 4.2 A Tag Stealing Attack

Here we introduce an attack where an attacker $\mathcal{A}$ who knows one secret tuple $(h(d_j), t_j)$ is able to steal all the other historical tags, i.e., $\{(h(d_i), t_i)\}_{i \in [L], i \neq j}$. In our attack, we exploit the fact that there is no authentication to the verifier. This fact enables an attacker masquerading the verifier $\mathsf{id_C}$ to choose two malicious selection sets $\mathsf{I}_1$ and $\mathsf{I}_2$, which only differ in one index that is associated with the target token, which we want to steal. In a nutshell, we need two assumptions that (i) $\mathcal{A}$ has corrupted the first authentication key $sk^1_{\mathsf{id_C},\mathsf{id_S}} = sk^1_{\mathsf{id_S},\mathsf{id_C}}$, and (ii) $\mathcal{A}$ learns one secret tuple $(h(d_j), t_j)$ with an arbitrary index $j$. Note that this is allowed by the CWZT scheme [17].

In the following, we show how the attacker $\mathcal{A}$ steals the $i^*$-th token (for $i^* \in [L]$ and $i^* \neq j$) holding by prover $\mathsf{id_S}$.

- $\mathcal{A}$ somehow corrupts $sk^1_{\mathsf{id_C},\mathsf{id_S}}$ and $(d_j, t_j)$.
- $\mathcal{A}$ masquerades as the verifier $\mathsf{id_C}$ to choose a randomness $r$ and a selection set $\mathsf{I}_1$, such that $i^* \notin \mathsf{I}_1$ and $j \in \mathsf{I}_1$.
- $\mathcal{A}$ sends $(\mathsf{I}_1, r)$ to $\mathsf{id_S}$ in a session, and receives the authentication messages $(X_1, Y_1)$.

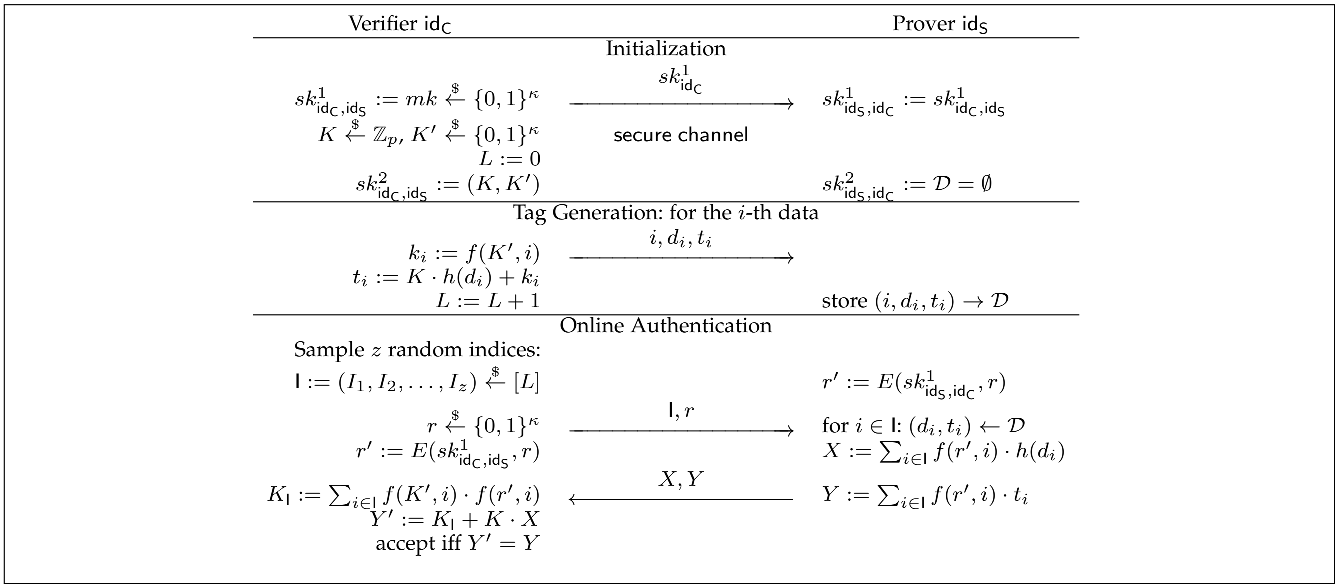

Verifier $\mathsf{id_C}$
Prover $\mathsf{id_S}$
Initialization
$sk^1_{\mathsf{id_C},\mathsf{id_S}} := mk \xleftarrow{\$} \{0,1\}^\kappa$
$sk^1_{\mathsf{id_C}}$
$sk^1_{\mathsf{id_S},\mathsf{id_C}} := sk^1_{\mathsf{id_C},\mathsf{id_S}}$
$K \xleftarrow{\$} \mathbb{Z}_p$, $K' \xleftarrow{\$} \{0,1\}^\kappa$
secure channel
$L := 0$
$sk^2_{\mathsf{id_C},\mathsf{id_S}} := (K, K')$
$sk^2_{\mathsf{id_S},\mathsf{id_C}} := \mathcal{D} = \emptyset$
Tag Generation: for the $i$-th data
$k_i := f(K', i)$
$i, d_i, t_i$
$t_i := K \cdot h(d_i) + k_i$
$L := L + 1$
store $(i, d_i, t_i) \to \mathcal{D}$
Online Authentication
Sample $z$ random indices:
$\mathsf{I} := (I_1, I_2, \ldots, I_z) \xleftarrow{\$} [L]$
$r' := E(sk^1_{\mathsf{id_S},\mathsf{id_C}}, r)$
$r \xleftarrow{\$} \{0,1\}^\kappa$
$\mathsf{I}, r$
for $i \in \mathsf{I}$: $(d_i, t_i) \leftarrow \mathcal{D}$
$r' := E(sk^1_{\mathsf{id_C},\mathsf{id_S}}, r)$
$X := \sum_{i\in\mathsf{I}} f(r', i) \cdot h(d_i)$
$K_{\mathsf{I}} := \sum_{i\in\mathsf{I}} f(K', i) \cdot f(r', i)$
$X, Y$
$Y := \sum_{i\in\mathsf{I}} f(r', i) \cdot t_i$
$Y' := K_{\mathsf{I}} + K \cdot X$
accept iff $Y' = Y$

Fig. 2: The CWZT Protocol [17].

$$h(d_{i^*}) = \frac{X_2 - X_1 + f(r', j) \cdot h(d_j)}{f(r', i^*)} = \frac{\left(\sum_{i\in \mathsf{I}_1 \setminus j} f(r', i) \cdot h(d_i) + f(r', i^*) \cdot h(d_{i^*})\right) - \sum_{i\in \mathsf{I}_1 \setminus j} f(r', i) \cdot h(d_i)}{f(r', i^*)}. \tag{1}$$

$$t_{i^*} = \frac{Y_2 - Y_1 + f(r', j) \cdot t_j}{f(r', i^*)} = \frac{\left(\sum_{i\in \mathsf{I}_1 \setminus j} f(r', i) \cdot t_i + f(r', i^*) \cdot t_{i^*}\right) - \sum_{i\in \mathsf{I}_1 \setminus j} f(r', i) \cdot t_i}{f(r', i^*)}. \tag{2}$$

- In another session, $\mathcal{A}$ chooses a selection set $\mathsf{I}_2$ by replacing the index $j$ with $i^*$, and sends $(\mathsf{I}_2, r)$ to $\mathsf{id_S}$ in another session, and receives the authentication messages $(X_2, Y_2)$.
- $\mathcal{A}$ computes $r' := E(sk^1_{\mathsf{id_C},\mathsf{id_S}}, r)$, $f(r', j)$, and $f(r', i^*)$.
- Then $\mathcal{A}$ can obtain $h(d_{i^*})$ and $t_{i^*}$ by Equation 1 and Equation 2, respectively.

By repeating the above attack steps, the attacker can obtain other authentication tokens as he/she wishes.

**Attack Discussion.** Note that the computation on the authentication proof $Y$ is a linear combination of the secrets derived from those authentication factors (i.e., ephemeral key $f(r', i)$ generated based on the symmetric key $sk^1_{\mathsf{id_S},\mathsf{id_C}}$ and historical tags $t_i$ ). However, the ephemeral keys derived by the first authentication factor $sk^1_{\mathsf{id_S},\mathsf{id_C}}$ cannot provide any protection for the historical tags in the computation of $Y$ since $sk^1_{\mathsf{id_S},\mathsf{id_C}}$ might be corrupted. Hence, the security of those authentication factors should be considered *independently* in the protocol design. Since the verifier (i.e., the client $\mathsf{id_C}$) cannot be explicitly authenticated (within two passes), the selection set $\mathsf{I}$ can be malicious, which implies that the authentication proof $Y$ is generated maliciously as well. Hence, the selection set should be determined by both parties instead. Based on the above observations, we will show how to avoid this problem in our HMACCE constructions.

## 5 HMACCE Security Model

In this section, we define new indistinguishability-based security models for historical data based multi-factor authenticated and confidential channel establishment protocols (HMACCE), and we will formulate the security goals that our upcoming HMACCE protocols can achieve.

**Threat Model.** Our security model follows the security models of ACCE in literature [7], [8], [36], [37]. In contrast to previous models, we particularly formulate the authentication factors related to historical data and the security property regarding leakage resilience. Namely, we allow the adversary (1) to have full knowledge of all public parameters in the protocol and have full control over the network, so he/she can send, eavesdrop, intercept, and alter messages in the network. (2) The adversary is also allowed to corrupt authentication factors used in the protocol, including keys and historical data. (3) The adversary can request to reveal the derived secret key of any party. (4) The adversary is allowed to get access to the randomness used by any party. (5) The adversary can also register additional malicious clients and authentication keys. (6) The adversary has access to an encryption oracle and a decryption oracle used in the protocol.

**Execution Environment.** Here, we consider an environment where two honest parties exist, i.e., an honest client $\mathsf{id_C}^*$ and an honest server $\mathsf{id_S}^*$. In the following, we let $\mathsf{ID}$ be a general identity to denote one of the honest parties in $\{\mathsf{id_C}^*, \mathsf{id_S}^*\}$.[2] However, we would allow an adversary to register new malicious clients. The client $\mathsf{id_C}$ and the server $\mathsf{id_S}$ would share a long-term symmetric authenti-

[2] Here, we only consider two honest parties for simplicity. Multiple honest parties' security can be asymptotically derived from the two-party case.

cation key $sk^1_{\mathsf{id_C},\mathsf{id_S}}$ as the first authentication factor. The second authentication key of a client is denoted by $sk^2_{\mathsf{id_C},\mathsf{id_S}}$ (which is used to verify the authentication message from $\mathsf{id_S}$). Besides the first symmetric authentication factor shared with the client, the server $\mathsf{id_S}$ would store distinct authentication factors, i.e., historical data $\mathcal{D}_1$ and the corresponding secret historical tags $\mathcal{D}_2$, where each piece of historical data is associated with a secret historical tag. We denote them by $sk^2_{\mathsf{id_S},\mathsf{id_C}} = \mathcal{D}_1$ and $sk^3_{\mathsf{id_S},\mathsf{id_C}} = \mathcal{D}_2$ such that $sk^\alpha_{\mathsf{id_S},\mathsf{id_C}} = (sk^\alpha_{\mathsf{id_S},\mathsf{id_C}}(1), sk^\alpha_{\mathsf{id_S},\mathsf{id_C}}(2), \ldots, sk^\alpha_{\mathsf{id_S},\mathsf{id_C}}(L))$ for $\alpha \in \{2,3\}$ that comprises of the sub-authentication keys denoted by $sk^\alpha_{\mathsf{id_S},\mathsf{id_C}}(i)$ for $i \in [L]$, where $L \in \mathbb{N}$ is the number of the stored historical data. Moreover, each party also maintains states $\{cst^i\}$ denoting the $i$-th authentication factor corruption status $cst^i \in \{\mathsf{exposed}, \mathsf{fresh}\}$ for $i \in \{1,2,3\}$. For example, if $sk^2_{\mathsf{id_S},\mathsf{id_C}}$ is corrupted, the party $\mathsf{id_S}$ must have $cst^i_{\mathsf{id_S},\mathsf{id_C}} = \mathsf{exposed}$. We assume the authentication factors of a party are stored independently so that the corruption of a factor does not affect the others.

To emulate the protocol executions, we assume that each party $\mathsf{ID}$ can carry out at most $\rho \in \mathbb{N}$ sessions that are modeled by a set of oracles $\{\pi^u_{\mathsf{ID}} : i \in [\ell], u \in [\rho]\}$. All oracles can have access to the authentication keys of their owner. Moreover, we assume each oracle $\pi^u_{\mathsf{ID}}$ maintains a list of independent internal state variables: (i) $\Phi^u_{\mathsf{ID}}$ – session decision $\Phi^u_{\mathsf{ID}} \in \{\mathtt{accept}, \mathtt{reject}\}$; (ii) $\mathsf{pid}^u_{\mathsf{ID}}$ – the identity of the intended communication partner; (iii) $K^u_{\mathsf{ID}}$ – session key of $\pi^u_{\mathsf{ID}}$; (iv) $T^u_{\mathsf{ID}}$ – protocol messages orderly sent and received by $\pi^u_{\mathsf{ID}}$; (v) $b^u_{\mathsf{ID}} \in \{0,1\}$ – a bit sampled by the challenger for each oracle $\pi^u_{\mathsf{ID}}$ at the beginning of the game and used in the security game of SLHAE.

We assume that the session key $K^u_{\mathsf{ID}}$ will be assigned with a non-empty value if and only if $\Phi^u_{\mathsf{ID}} = \mathtt{accept}$, which means that the session key is accepted by the corresponding party. [3]

**Adversarial Model.** To model the power of an active adversary $\mathcal{A}$, we realize $\mathcal{A}$ as a probabilistic polynomial time (PPT) algorithm that can ask the following queries:

- $\mathsf{Send}(\mathsf{ID}, u, m)$[4]: The adversary can send any message $m$ to the oracle $\pi^u_{\mathsf{ID}}$ via this query. Oracle $\pi^u_{\mathsf{ID}}$ will respond to the next protocol message $m^*$ (if any) to be sent according to the protocol specification and its internal states. An oracle of the honest client $\mathsf{id_C}^*$ is initiated via sending the oracle the first message $m = \top$ consisting of a special initialization symbol $\top$. The oracle variables will be updated accordingly (following the protocol specification) after each Send query.
- $\mathsf{RevealKey}(\mathsf{ID}, u)$: The oracle $\pi^u_{\mathsf{ID}}$ responds with the value of $K^u_{\mathsf{ID}}$.
- $\mathsf{Corrupt1}(\mathsf{ID}_1, \mathsf{ID}_2)$: For honest parties $(\mathsf{ID}_1, \mathsf{ID}_2) \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}$, this query returns the first authentication key $sk^1_{\mathsf{ID}_1,\mathsf{ID}_2}$ of an honest party $\mathsf{ID}_1$, and sets $cst^1_{\mathsf{ID}_1,\mathsf{ID}_2} = cst^1_{\mathsf{ID}_2,\mathsf{ID}_1} := \mathsf{exposed}$.
- $\mathsf{Corrupt2}(\mathsf{ID}_1, \mathsf{ID}_2)$: For honest parties $(\mathsf{ID}_1, \mathsf{ID}_2) \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}$, this query returns the second authentication key $sk^2_{\mathsf{ID}_1,\mathsf{ID}_2}$ of an honest party $\mathsf{ID}_1$, and sets $cst^2_{\mathsf{ID}_1,\mathsf{ID}_2} := \mathsf{exposed}$.
- $\mathsf{Corrupt3}$: This query returns the third authentication key $sk^3_{\mathsf{id_S}^*,\mathsf{id_C}^*}$, and sets $cst^3_{\mathsf{id_S}^*,\mathsf{id_C}^*} := \mathsf{exposed}$.
- $\mathsf{RevealR}(\mathsf{ID}, u)$[5]: This query returns the randomness (e.g., the ephemeral Diffie-Hellman key) generated by $\pi^u_{\mathsf{ID}}$.
- $\mathsf{HTLeak}(i)$: This query returns the $i$-th sub-key $sk^3_{\mathsf{id_S}^*,\mathsf{id_C}^*}(i)$.
- $\mathsf{RegClient}(\mathsf{id_C}_i, sk^1_{\mathsf{id_C}_i,\mathsf{id_S}^*}, sk^2_{\mathsf{id_C}_i,\mathsf{id_S}^*}, sk^2_{\mathsf{id_S}^*,\mathsf{id_C}_i}, sk^3_{\mathsf{id_S}^*,\mathsf{id_C}_i})$: This query allows the adversary to register malicious clients and authentication keys. If $\mathsf{id_C}_i$ exists, then the old keys will be replaced with the input ones.
- $\mathsf{Encrypt}(\mathsf{ID}, u, m_0, m_1, \ell_c, H)$: This query takes as input two messages $m_0$ and $m_1$, length parameter $\ell_c$, and header data $H$. If $\Phi^u_{\mathsf{ID}} \neq \mathtt{accept}$ then $\pi^u_{\mathsf{ID}}$ returns $\bot$. Otherwise, it proceeds as the ENC oracle query in Figure 5, depending on the session key $K^u_{\mathsf{ID}}$, the random bit $b^u_{\mathsf{ID}}$, and the other internal states of $\pi^u_{\mathsf{ID}}$ (e.g., $cnt_e$ and $st_e$).
- $\mathsf{Decrypt}(\mathsf{ID}, u, C, H)$: This query takes as input a ciphertext $C$ and header data $H$. If $\pi^u_{\mathsf{ID}}$ has $\Phi^u_{\mathsf{ID}} \neq \mathtt{accept}$ then $\pi^u_{\mathsf{ID}}$ returns $\bot$. Otherwise, it proceeds as the DEC oracle in Figure 5, depending on the session key $K^u_{\mathsf{ID}}$, the random bit $b^u_{\mathsf{ID}}$, and the other internal states of $\pi^u_{\mathsf{ID}}$ (e.g., $cnt_d$, $st_d$, and phase).

**Secure HMACCE Protocols.** We first review a notion called *matching conversations* [21] to formulate the relation between two sessions. We will use a variant that is refined in [7].

*Matching Conversations.* An oracle $\pi^u_{\mathsf{ID}}$ is said to have a matching conversation to an oracle $\pi^v_{\mathsf{pid}^u_{\mathsf{ID}}}$, if either (i) $\pi^u_{\mathsf{ID}}$ has sent all protocol messages, and $T^v_{\mathsf{pid}^u_{\mathsf{ID}}}$ is a prefix of $T^u_{\mathsf{ID}}$, or (ii) $\pi^v_{\mathsf{pid}^u_{\mathsf{ID}}}$ has sent all protocol messages, and $T^u_{\mathsf{ID}}$ is a prefix of $T^v_{\mathsf{pid}^u_{\mathsf{ID}}}$. We also call $\pi^v_{\mathsf{pid}^u_{\mathsf{ID}}}$ meeting all above conditions to be the partner oracle of $\pi^u_{\mathsf{ID}}$.

*Correctness.* We say an HMACCE protocol $\Pi$ is correct, if two accepted oracles $\pi^u_{\mathsf{id_C}^*}$ and $\pi^v_{\mathsf{id_S}^*}$ have matching conversations, then both oracles should generate the same session key.

We will use a variable $\mathsf{MN} \in \{\mathsf{FS}, \mathsf{woFS}\}$ to denote the HMACCE security either with PFS (Perfect Forward Secrecy) or without PFS (woFS). In the following, we present a unified security experiment with/without FS based on MN. For an HMACCE protocol without PFS, we only define static historical tag leakage. However, for an HMACCE protocol with PFS, we define adaptive historical tag leakage.

*HMACCE Security Experiment* ($\Pi$, MN)*:* A challenger $\mathcal{C}$ will play a game with an adversary $\mathcal{A}$ based on a target HMACCE protocol $\Pi$ and the security variable MN. In the initialization phase of the game, $\mathcal{C}$ first implements a collection of oracles $\{\pi^u_{\mathsf{ID}} : \mathsf{ID} \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}, u \in [\rho]\}$ and randomly samples $\{b^u_{\mathsf{ID}} : \mathsf{ID} \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}, u \in [\rho]\}$ for the honest client $\mathsf{id_C}^*$ and the honest server $\mathsf{id_S}^*$, respectively. All authentication keys are generated according to the protocol specifications. $\mathcal{C}$ gives the adversary $\mathcal{A}$ all identities

[3] Note that, throughout the paper, the superscript $u$ of an oracle or a state of an oracle is the index of the oracle, while the other superscripts are 1, 2 or 3 (e.g., $sk^1_{\mathsf{id_C},\mathsf{id_S}}$, $sk^2_{\mathsf{id_C},\mathsf{id_S}}$, and $sk^3_{\mathsf{id_C},\mathsf{id_S}}$) denoting which authentication factor it is referring to. The subscript always represents the ID of a user.

[4] The Send query allows all interactions between the adversary and any parties. It essentially covers the adversarial capabilities in the traditional Dolev-Yao model [37].

[5] The RevealR query models the session-state reveal capability introduced in the CK model [36]. That is, we specifically define the state that can be leaked, to be the randomness sampled during the protocol execution. All other intermediate secret values for session key generation should be well protected without being compromised.

as input. There are two phases in the game, and in each phase, distinct queries can be asked. In the first phase, $\mathcal{A}$ is allowed to ask queries to HTLeak to model static historical tag leakage. $\mathcal{A}$ can send $\mathcal{C}$ a symbol $\vdash$ to switch to the next phase. In the second phase, $\mathcal{A}$ can ask a polynomial number of queries to Send, Corrupt1, Corrupt2, Corrupt3, RevealKey, RevealR, RegClient, Encrypt, and Decrypt. If $\mathsf{MN} = \mathsf{woFS}$, the HTLeak query is not allowed in the second phase. However, if $\mathsf{MN} = \mathsf{FS}$, the adversary can query HTLeak in this phase to model adaptive leakage. Eventually, $\mathcal{A}$ may terminate and output a tuple $(\mathsf{ID}, u, b')$ as its guess for the bit $b^u_{\mathsf{ID}}$ of $\pi^u_{\mathsf{ID}}$, where the oracle $\pi^u_{\mathsf{ID}}$ will be called as the *test oracle*.

The difference between static and adaptive historical tag leakage is whether the HTLeak query is allowed in the second phase of the above security experiment. We give a formulation of *full corruption* (of a party) as follows so that partial corruption is its complement.

*Full Corruption.* We define the full corruption of a party $\mathsf{ID} \in \{\mathsf{id_C}, \mathsf{id_S}\}$ via a function FullC which takes as input two identities $(\mathsf{id_C}, \mathsf{id_S})$ and the number $q_l$ of HTLeak query that is allowed, and outputs 1 to denote full corruption of ID and 0 otherwise. $\mathsf{FullC}(\mathsf{ID}, \mathsf{id_C}, \mathsf{id_S}, q_l) = 1$ if one of the following conditions holds:

1) $\mathsf{id_C}$ was taken as input to any RegClient query;
2) $cst^1_{\mathsf{id_C},\mathsf{id_S}} = cst^2_{\mathsf{id_C},\mathsf{id_S}} = \mathsf{exposed}$;
3) $cst^1_{\mathsf{id_S},\mathsf{id_C}} = cst^2_{\mathsf{id_S},\mathsf{id_C}} = cst^3_{\mathsf{id_S},\mathsf{id_C}} = \mathsf{exposed}$;
4) $cst^1_{\mathsf{id_S},\mathsf{id_C}} = cst^2_{\mathsf{id_S},\mathsf{id_C}} = \mathsf{exposed}$ and $\mathcal{A}$ asked more than $q_l$ HTLeak queries;
5) $\mathsf{ID} = \mathsf{id_C}$ and $cst^1_{\mathsf{id_S},\mathsf{id_C}} = cst^3_{\mathsf{id_S},\mathsf{id_C}} = \mathsf{exposed}$;
6) $\mathsf{ID} = \mathsf{id_C}$, $cst^1_{\mathsf{id_C},\mathsf{id_S}} = \mathsf{exposed}$ and $\mathcal{A}$ asked more than $q_l$ HTLeak queries.

The last two conditions are added because $\mathsf{id_C}$ has one less authentication factor than $\mathsf{id_S}$. Basically, we shall model the authentication for a specific party $\mathsf{ID} \in \{\mathsf{id_C}, \mathsf{id_S}\}$ when $\mathsf{FullC}(\mathsf{ID}, \mathsf{id_C}, \mathsf{id_S}, q_l) = 0$.

In the following security definition, we let $\mathsf{ID}^*$ denote the party submitted to the Encrypt query for the test and let $\widetilde{\mathsf{ID}^*}$ denote the identity that is required to provide explicit authentication.

***Definition 1 (HMACCE Security).*** We say a PPT adversary $\mathcal{A}$ $(t, \epsilon, q_l, \mathsf{MN})$-breaks an HMACCE protocol $\Pi$ in the security experiment with MN, if $\mathcal{A}$ runs in time $t$, and one of the following conditions is satisfied:

- **Authentication**: When $\mathcal{A}$ terminates, then with probability $\epsilon$ there exists an oracle $\pi^u_{\widetilde{\mathsf{ID}^*}}$ such that
  - $\mathsf{FullC}(\widetilde{\mathsf{ID}^*}, \widetilde{\mathsf{ID}^*}, \mathsf{pid}^u_{\widetilde{\mathsf{ID}^*}}, q_l) = 0$ when $\pi^u_{\widetilde{\mathsf{ID}^*}}$ accepts, and
  - $\pi^u_{\widetilde{\mathsf{ID}^*}}$ has no unique partner oracle at the party $\mathsf{pid}^u_{\widetilde{\mathsf{ID}^*}}$.

  We say that $\pi^u_{\widetilde{\mathsf{ID}^*}}$ accepts *maliciously* if it accepts satisfying the above conditions.
- **Channel Confidentiality** : When $\mathcal{A}$ terminates and outputs a tuple $(\mathsf{ID}^*, u, b')$, and
  - $\mathcal{A}$ asked an $\mathsf{Encrypt}(\mathsf{ID}^*, u, \cdot, \cdot, \cdot, \cdot)$ query without failure, and
  - if $\mathsf{MN} = \mathsf{woFS}$ then $\mathsf{FullC}(\mathsf{ID}^*, \mathsf{ID}^*, \mathsf{pid}^u_{\mathsf{ID}^*}, q_l) = 0$ and $\mathsf{FullC}(\mathsf{pid}^u_{\mathsf{ID}^*}, \mathsf{ID}^*, \mathsf{pid}^u_{\mathsf{ID}^*}, q_l) = 0$, and
  - if $\mathsf{MN} = \mathsf{FS}$ then $\mathsf{FullC}(\mathsf{ID}^*, \mathsf{ID}^*, \mathsf{pid}^u_{\mathsf{ID}^*}, q_l) = 0$ and $\mathsf{FullC}(\mathsf{pid}^u_{\mathsf{ID}^*}, \mathsf{ID}^*, \mathsf{pid}^u_{\mathsf{ID}^*}, q_l) = 0$ when $\pi^u_{\mathsf{ID}^*}$ accepts, and
  - $\mathcal{A}$ neither asked $\mathsf{RevealKey}(\mathsf{ID}^*, u)$ nor $\mathsf{RevealR}(\mathsf{ID}^*, u)$, and
  - if $\pi^v_{\mathsf{pid}^u_{\mathsf{ID}^*}}$ is a partner oracle of the test oracle $\pi^u_{\mathsf{ID}^*}$, $\mathcal{A}$ queried neither $\mathsf{RevealKey}(\mathsf{pid}^u_{\mathsf{ID}^*}, v)$ nor $\mathsf{RevealR}(\mathsf{pid}^u_{\mathsf{ID}^*}, v)$,

  and then the probability $b'$ equals to the bit $b^u_{\mathsf{ID}}$ satisfies $|\Pr[b' = b^u_{\mathsf{ID}^*}] - 1/2| \geq \epsilon$. We say that $\mathcal{A}$ *answers the encryption-challenge correctly* if $b' = b^u_{\mathsf{ID}^*}$ and all above conditions are met.

We say that an HMACCE protocol is $(t, \epsilon, q_l, \mathsf{MN})$-secure, if there exists no PPT adversary that $(t, \epsilon, q_l, \mathsf{MN})$-breaks it.

***Remark 1.*** Here, we define a model with static credentials for a fixed period of time. However, our model can be extended to analyze multiple instances of time periods (e.g., using a sliding window approach as in our construction) if each time window has independent static credentials. In a protocol design, honest parties can switch to the next time window when the authentication credentials for the current time window have been used for a predefined number of authentication. It is further elaborated in our constructions in Section 6. Since the authentication credentials used in $\mathrm{TW}_i$ (Time Window $i$) are generated within $\mathrm{TW}_{i-1}$, the leakage of authentication credentials (including data and tags) occurs in $\mathrm{TW}_{i-1}$. Nevertheless, we stress that such leakage is already covered by our defined queries, i.e., the exposed session key in $\mathrm{TW}_{i-1}$ (e.g., caused by RevealKey query) may be the 'reason' of the HTLeak query in $\mathrm{TW}_i$. Note that, in our formalization of HTLeak and Corrupt queries, we do not constrain what caused the leakage or corruption. In other words, such leakage or corruption can be caused by attacks (e.g., session key compromise) launched in $\mathrm{TW}_{i-1}$. Hence, we can focus on the security formalism of a one-time window. If the secure channel established in a one-time window is secure, then the authentication credentials transmitted within this time window are secure as well, which lays the foundation of the security of the next time window. Meanwhile, the synchronization of time windows can refer to other time-based cryptographic primitives, such as time-based one-time passwords [38], [4], [39].

## 6 An Efficient HMACCE Protocol

In this section, we develop an efficient HMACCE Protocol in the random oracle model denoted by $\Pi_{\mathsf{woFS}}$. The main construction idea of $\Pi_{\mathsf{woFS}}$ is to directly use authentication factors to derive a session key.

**Protocol Description.** Let $\mathbb{Z}_p$ be a cyclic group with a prime order $p$ that has a bit-length $\ell_p$, and $\mathbb{Z}_p^* = \mathbb{Z}_p/\{0\}$. In our protocol, we need a cryptographic hash function $h : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and a stateful length hiding authenticated encryption $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$. We assume that the server chooses a uniform salt $\chi_{\mathsf{id_S}}$ for each client to randomize the hash function, which is implicitly used as input of $h$. Let $\ell_r$ be a bit-length defining a randomness space. In our protocols, the historical data is considered as one of the authentication factors, so we assume it to

| client $\mathsf{id_C}$ | | server $\mathsf{id_S}$ |
|---:|:---:|:---|
| | Initialization | |
| $sk^1_{\mathsf{id_C},\mathsf{id_S}} := mk \overset{\$}{\leftarrow} \{0,1\}^\kappa$ | $\xrightarrow{sk^1_{\mathsf{id_C}}}$ | $sk^1_{\mathsf{id_S},\mathsf{id_C}} := sk^1_{\mathsf{id_C},\mathsf{id_S}}$ |
| $sk^2_{\mathsf{id_C},\mathsf{id_S}} := K \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ | secure channel | $sk^2_{\mathsf{id_S},\mathsf{id_C}} := \mathcal{D}_1 = \{(i, d_i, h(d_i \Vert i))\}_{i\in[L]}$ |
| $cnt := 0$ | | $sk^3_{\mathsf{id_S},\mathsf{id_C}} := \mathcal{D}_2 = \{(i, t_i)\}_{i\in[L]}$ |
| $st_e := \emptyset$ | | $st_d := \emptyset$ |
| | Online Authentication and Key Exchange | |
| Sample $z$ distinct random indices: | | Sample $z$ distinct random indices: |
| $\mathsf{I}_C := (i_1, i_2, \ldots, i_z) \overset{\$}{\leftarrow} \mathsf{SI}$ | | $\mathsf{I}_S \overset{\$}{\leftarrow} \mathsf{SI} \backslash \mathsf{I}_C$ |
| $r_1 \overset{\$}{\leftarrow} \{0,1\}^{\ell_r}$ | $\xrightarrow{\mathsf{I}_C, r_1}$ | $r_2 \overset{\$}{\leftarrow} \{0,1\}^{\ell_r}$ |
| $\mathsf{I} := \mathsf{I}_C \cup \mathsf{I}_S$ | $\xleftarrow{\mathsf{I}_S, r_2, X, M}$ | $\mathsf{I} := \mathsf{I}_C \cup \mathsf{I}_S$ |
| $\mathsf{sid} := \mathsf{id_C} \Vert r_1 \Vert \mathsf{id_S} \Vert X \Vert r_2 \Vert \mathsf{I}$ | | for $i \in \mathsf{I}$: $(h(d_i \Vert i), t_i) \leftarrow \mathcal{D}_1 \& \mathcal{D}_2$ |
| $K_\mathsf{I} := \sum_{i\in\mathsf{I}} h(K \Vert i)$ | | $X := \sum_{i\in\mathsf{I}} h(d_i \Vert i)$ |
| $Y' := K_\mathsf{I} + K \cdot X$ | | $Y := \sum_{i\in\mathsf{I}} t_i$ |
| $M' := h(mk \Vert Y' \Vert \mathsf{sid} \Vert \text{‘Auth’})$ | | $\mathsf{sid} := \mathsf{id_C} \Vert r_1 \Vert \mathsf{id_S} \Vert X \Vert r_2 \Vert \mathsf{I}$ |
| reject if $M \neq M'$ | | $M := h(mk \Vert Y \Vert \mathsf{sid} \Vert \text{‘Auth’})$ |
| accept $K_s := h(mk \Vert Y' \Vert \mathsf{sid} \Vert \text{‘SeK’})$ | | $K_s := h(mk \Vert Y \Vert \mathsf{sid} \Vert \text{‘SeK’})$ |
| | Tag Generation: for the $i$-th data | |
| $k_i := h(K \Vert i)$ | $\xrightarrow{C_i}$ | $(m_i, st'_d) := \mathsf{StE.Dec}(K_s, H, C_i, st_d)$ |
| $t_i := K \cdot h(d_i \Vert i) + k_i$ | | accept if $m_i \neq \perp$ |
| $m_i := i \Vert d_i \Vert t_i$ | | $i \Vert d_i \Vert t_i \leftarrow m_i$ |
| $(C_i, st'_e) \overset{\$}{\leftarrow} \mathsf{StE.Enc}(K_s, \ell_c, H, m_i, st_e)$ | | store $(i, d_i, h(d_i \Vert i)) \rightarrow \mathcal{D}_1$ |
| $cnt := cnt + 1$ | | store $(i, t_i) \rightarrow \mathcal{D}_2$ |

Fig. 3: An Efficient HMACCE Protocol $\Pi_{\mathsf{woFS}}$.

be unpredictable and have some min-entropy[6]. As stated in [41], any unpredictable string (regardless of its min-entropy) with bit-length that is larger than $\ell_p$, in the random oracle model, can be used to extract an unpredictable $\ell_p$-bit uniform random string in $\mathbb{Z}_p$. To avoid the leakage of historical data and tags being over the security threshold, we adopt a sliding window alike approach. We let $\mathsf{SI}$ be a set with size $L$, which stores the indices of historical data and tags that will be used for authentication and key exchange. We assume that the indices in $\mathsf{SI}$ can be used at most $\phi$ times, so once they have been used $\phi$ times, we will refresh $\mathsf{SI}$ with the following $L$ unused historical data and tags from $(\mathcal{D}_1, \mathcal{D}_2)$.

The protocol $\Pi_{\mathsf{woFS}}$ running between a client $\mathsf{id_C}$ and a server $\mathsf{id_S}$ is shown in Figure 3, which consists of three phases described below.

- **Initialization**. In this phase, the client $\mathsf{id_C}$ and the server $\mathsf{id_S}$ first randomly generate a symmetric authentication key $sk^1_{\mathsf{id_C},\mathsf{id_S}} = sk^1_{\mathsf{id_S},\mathsf{id_C}} := mk \overset{\$}{\leftarrow} \{0,1\}^\kappa$ which is used as the first authentication factor. The second authentication factor of $\mathsf{id_C}$ is randomly chosen as $sk^2_{\mathsf{id_C},\mathsf{id_S}} = K \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, whereas the second and third authentication factors of $\mathsf{id_S}$ are initialized with random sets $(sk^2_{\mathsf{id_S},\mathsf{id_C}}, sk^3_{\mathsf{id_S},\mathsf{id_C}}) = (\mathcal{D}_1, \mathcal{D}_2) = (\{(i, d_i, h(d_i||i))\}_{i\in[L]}, \{(i, t_i)\}_{i\in[L]})$. That is, to bootstrap the protocol, we require the client to randomly generate $L$ data and compute the corresponding tags in $\mathcal{D}_2$ with the method described in the tag generation phase using $K$. Also, since we assume no adversary exists during the initialization phase, the transmission of the $L$ dummy data and tags does not require encryption. In practice, one can encrypt the data and tag transmission using the first authentication factor $mk$ or the previous secure session key when applicable.
- **Tag Generation**. When the client $\mathsf{id_C}$ sends a data piece $d_i$ to the server $\mathsf{id_S}$, $\mathsf{id_C}$ would compute an authentication tag $t_i$ based on $sk^2_{\mathsf{id_C},\mathsf{id_S}} = K$. Each tag is generated as $t_i := K \cdot h(d_i||i) + k_i \pmod p$, where $k_i := h(K||i)$. After the tag is generated, $\mathsf{id_C}$ would locally increase the tag counter $cnt$ by one, encrypt the message $m_i = i||d_i||t_i$ to yield the ciphertext $(C_i, st'_e) \overset{\$}{\leftarrow} \mathsf{StE.Enc}(K_s, \ell_c, H, m_i, st_e)$, and send $C_i$ to $\mathsf{id_S}$. Then $\mathsf{id_S}$ decrypts $C_i$ to get the plaintext $(m_i, st'_d) = \mathsf{StE.Dec}(K_s, H, C_i, st_d)$, and privately stores the tuple $(i, d_i, h(d_i||i)) \rightarrow \mathcal{D}_1$ and $(i, t_i) \rightarrow \mathcal{D}_2$, i.e., $sk^2_{\mathsf{id_S},\mathsf{id_C}}(i) = (i, d_i, h(d_i||i))$ and $sk^3_{\mathsf{id_S},\mathsf{id_C}}(i) = (i, t_i)$. In the first communication after a new channel is established, the server will verify whether the ciphertext sent by the client can be successfully decrypted. If yes, the server accepts the client as an authenticated entity. Note that a session key may be used to transmit multiple data and corresponding tags. We assume the client should generate at least $L$ data and tag pairs in one window.
- **Authentication and Confidential Channel Establishment Phase**. The client $\mathsf{id_C}$ and the server $\mathsf{id_S}$ would interactively run the authenticated and confidential channel establishment protocol online to generate a session key $K_s$ as shown in Figure 3. One of the key procedures of confidential channel establishment is the *key exchange* phase which is used to generate a session key shared by two communication partners. The established session key will be used to protect the underlying data and tag transmission using $\mathsf{StE}$. During this phase, both parties would first exchange two random nonces $r_1, r_2 \overset{\$}{\leftarrow} \{0,1\}^{\ell_r}$, and two random index selection sets $(\mathsf{I}_C, \mathsf{I}_S)$ with $z$ dis-

[6] As a validation of this assumption, we evaluated the min-entropy of sensor measurements in real industrial control systems based on one dataset of the operations of a real-world water treatment system [40]. The min-entropy of the sensor data in each stage is between 4.52 and 7.80 when the system is running.

tinct random indices in each set, where $\mathsf{I}_C \overset{\$}{\leftarrow} \mathsf{SI}$ and $\mathsf{I}_S \overset{\$}{\leftarrow} \mathsf{SI} \backslash \mathsf{I}_C$. Let $\mathsf{I} = \mathsf{I}_C \cup \mathsf{I}_S$. Next, $\mathsf{id}_\mathsf{S}$ makes use of its historical data (indexed by $\mathsf{I}$) to compute a message $X := \sum_{i \in \mathsf{I}} h(d_i||i) \pmod p$, and an intermediate secret $Y := \sum_{i \in \mathsf{I}} t_i \pmod p$. In our scheme, the hash values of data are not secrets. Next, $Y$ is used as a secret seed to generate the authentication message $M := h(mk||Y||\mathsf{sid}||\text{'Auth'})$ and the final session key $K_s := h(mk||Y'||\mathsf{sid}||\text{'SeK'})$, where $\mathsf{sid}$ is the session identifier concatenating the protocol messages and identities of participants. The messages $(X, M)$ are sent to $\mathsf{id}_\mathsf{C}$ for authentication. To verify $M$, $\mathsf{id}_\mathsf{C}$ computes $K_\mathsf{I} := \sum_{i \in \mathsf{I}} h(K||i) \pmod p$, $Y' := K_\mathsf{I} + K \cdot X \pmod p$, and $M' := h(mk||Y'||\mathsf{sid}||\text{'Auth'})$. If $M' \neq M$, then $\mathsf{id}_\mathsf{C}$ rejects the session. Otherwise, it generates the session key as $\mathsf{id}_\mathsf{S}$. We assume that two parties synchronize a variable $\xi$ which stores the times of the selection set $\mathsf{SI}$ that has been used. If $\xi = \phi$, then all indices in $\mathsf{SI}$ plus $L$, i.e., sliding to the next time window. Moreover, the entity authentication of the client is verified when the server receives a valid ciphertext.

**Construction Discussions.** To improve the CWZT protocol, we modify and add several critical steps to fix the vulnerabilities of the CWZT protocol and achieve the HMACCE functionality. Below, we highlight our main differences with the state-of-the-art CWZT protocol [17].

- *Security improvement for authentication.* In Section 4, we have shown an attack to subvert the leakage resilient security property of the CWZT scheme, that an attacker who corrupts the first authentication factor and one piece of data and its tag can then steal all other secret tags. To circumvent this attack, the server in $\Pi_\mathsf{woFS}$ contributes a random set $\mathsf{I}_S$, such that the subset of selected historical data is determined by both parties (see Figure 3), instead of only relying on the client.
- *New confidential channel establishment feature.* Unlike the CWZT protocol, our protocol realizes the full-fledged authenticated and confidential channel establishment (achieving both authentication and channel confidentiality security goals). Our protocol enables both parties to establish an authenticated and confidential channel for securely transmitting the new authentication factors (i.e., data and its associated tags) so that the historical data based authentication and key exchange make sense.
- *Other security considerations*. We consider data and its tag as distinct authentication factors because they are stored separately. The adversary who only corrupts the tags or the data cannot actively impersonate the server to the client. For instance, if the adversary does not know the data, then it is unable to generate a valid $X$ to make the client accept $M$. Moreover, unlike the CWZT protocol, each party should contribute a nonce $r_i$ (for $i \in \{1, 2\}$) so that the session identifier is unique in each session to resist *replay attacks*.
- *Performance improvement.* Unlike the CWZT protocol, our protocol does not derive many session-specific ephemeral keys from the first authentication factor to protect $Y$. Since $Y$ is protected by a hash function in our scheme, we could simplify its computation to achieve better performance. As a result, we roughly save 3× hash operations compared to the CWZT protocol, although we provide an additional key exchange functionality.

**Limitations.** One of the limitations of $\Pi_\mathsf{woFS}$ is that it cannot provide forward secrecy when all secrets used to compute a session key $K_s$ of a player are compromised. If the client is not fully corrupted, then along with the growth of the second authentication factor, the newly generated session key, depending on the selection set (which is chosen from an increasingly larger range), can still be secure. As we will show in the security proof that the probability regarding the event: all indices of a selection set chosen in a session are compromised by the adversary before, is negligible with a proper choice of $z$ (e.g., $z = 163$ for 128 bits security). Thus, the attacker needs to either keep stealing the second and third authentication factors or compromise the client's device, which might be located in a more physically secure place in CPSs.

Another limitation of $\Pi_\mathsf{woFS}$ is that it can only satisfy static historical tag leakage. When the $\mathsf{HTLeak}$ query can be asked adaptively, the adversary will be able to ask $\mathsf{HTLeak}$ queries with indices that appeared in the $\mathsf{Encrypt}$ query to break the confidential channel security. In addition, if the adversary obtained more than $q_l$ tags, then the key exchange security is jeopardized since the session key is derived from those tags. This limitation of $\Pi_\mathsf{woFS}$ is caused by the side-effect of using the secret tags for both authentication and key exchange features.

***Theorem 1.*** Suppose that the hash function $h$ is indistinguishable from a random oracle in time $t_h$ and with at most $q_h$ queries, the stateful length hiding authenticated encryption $\mathsf{StE}$ is $(t_\mathsf{SLHAE}, \epsilon_\mathsf{SLHAE})$-secure, and each data piece is unpredictable. Then $\Pi_\mathsf{woFS}$ is $(t', \epsilon_{\Pi_\mathsf{woFS}}, q_l, \mathsf{woFS})$-secure with $t' = (t_h + t_\mathsf{SLHAE}) \approx t$, $\phi \leq q_l$, and $\epsilon_{\Pi_\mathsf{woFS}} \leq \frac{\rho^2}{2^{\ell_r - 1}} + 14\rho \cdot (\frac{q_l}{L-z})^z + \frac{(14\rho+22+6L)\cdot q_h}{2^{\ell_p}} + (2\rho^2 + 6\rho) \cdot \epsilon_\mathsf{SLHAE}$.

**Security Analysis.** We divide adversaries into two categories to analyze the authentication and key exchange respectively:
(i) *Authentication-adversary* can succeed in making an oracle accept maliciously; (ii) *Encryption-adversary* is able to answer the encryption-challenge correctly.

To prove Theorem 1, we present two lemmas. Each analyzes one of the security properties of the proposed protocol. Specifically, Lemma 1 bounds the success probability $\epsilon_\mathsf{auth}$ of authentication-adversaries, and Lemma 2 bounds the advantage $\epsilon_\mathsf{enc}$ of encryption-adversaries. Then we have $\epsilon_{\Pi_\mathsf{woFS}} \leq \epsilon_\mathsf{auth} + \epsilon_\mathsf{enc}$.

The full proof of Theorem 1 is given in [42]. In the following, we just present the outline of the proof.

***Lemma 1.*** For any adversary $\mathcal{A}$ running in time $t' \approx t$, the probability that there exists an oracle $\Pi^u_{\mathsf{id}_\mathsf{C}^*}$ that accepts maliciously is at most $\epsilon_\mathsf{auth} \leq \frac{\rho^2}{2^{\ell_r}} + 6\rho \cdot (\frac{q_l}{L})^z + \frac{(6\rho+9+3L)\cdot q_h}{2^{\ell_p}} + 6\rho \cdot \epsilon_\mathsf{SLHAE}$.

The proof of this lemma has four main steps. First, we exclude the collision among the random nonces, which occurs with negligible probability $\frac{\rho^2}{2^{\ell_r}}$ due to the birthday paradox. Let $\mathsf{S}$ be the set of indices submitted to the $\mathsf{HTLeak}$ query. Then, in a second step, when the third authentication factor is not corrupted (which occurs with probability $1/3$

since there are 3 authentication factors), then the probability that an oracle $\pi^u_{\mathsf{id_C}^*}$ accepts maliciously and sends out a selection set $\mathsf{I}^*_C$ such that $\mathsf{I}^*_C \subseteq \mathsf{S}$ is about $1-(\Pr[\overline{\mathsf{I}^*_C \subseteq \mathsf{S}}])^\rho = 1-(1-(\frac{q_l}{L})^z)^\rho < \rho\cdot(\frac{q_l}{L})^z$. This implies that at least one factor of a party, which is not fully corrupted, is not known by the adversary. Hence, the adversary is only able to obtain the session key by its random guesses. In the last step, we can exclude the attacks of adversaries by injecting ciphertexts that are not sent by a fresh oracle due to the security of SLHAE.

***Lemma* 2.** For any adversary $\mathcal{A}$ running in time $t' \approx t$, the advantage of an adversary $\mathcal{A}$ which answers encryption-challenge better than a random guess is at most $\epsilon_{\mathsf{enc}} \leq \frac{\rho^2}{2^{\ell_r}} + 8\rho \cdot (\frac{q_l}{L-z})^z + \frac{(8\rho+13+3L)\cdot q_h}{2^{\ell_p}} + 2\rho^2 \cdot \epsilon_{\mathsf{SLHAE}}$.

The proof of this lemma mainly relies on authentication security and compromised secret tags. The key issue here is whether the adversary knows all secret tags used to compute the session key of the test oracle. Note that the adversary can only manipulate the selection set of the client $\mathsf{I}_C$, which is not authenticated. Hence, the probability that the selection set $\mathsf{I}^*_S$ used by the test oracle such that $\mathsf{I}^*_S \subseteq \mathsf{S}$ is about $\rho \cdot (\frac{q_l}{L-z})^z$ which can be negligible with proper parameters. Because of the security of $\mathsf{StE}$, the adversary is not able to distinguish which message ($m_0$ or $m_1$) is encrypted.

**Discussion.** Note that our formal model and security analysis cover the Ephemeral Secret Leakage (ESL) attacks due to the formalization of the $\mathsf{RevealR}$ query. An attacker is allowed to get all randomness used by a party in a session. Our security proof shows that our protocol is secure in our security model.

**Resilience against Quantum Attacks.** Industrial control system devices usually have a long lifespan, so the threats of quantum computers become a practical concern for $\Pi_{\mathsf{woFS}}$ if deployed. As we showed in Theorem 1, the security of $\Pi_{\mathsf{woFS}}$ can be reduced to the security of the underlying hash function and stateful length hiding authenticated encryption. These two functions can be instantiated by SHA-3/SHA-2 and AES, respectively, as used in the selected algorithms in the NIST PQC competition [43]. We know that idealized hash function and stateful length hiding authenticated encryption are mainly subject to the attack of Grover's algorithm [44], which likely requires the output/key size to be doubled for achieving the same security level under quantum attacks as the security level used to hold for classical adversaries. Moreover, since $\Pi_{\mathsf{woFS}}$ is a generic construction based on hash functions and stateful length hiding authenticated encryption, one can always drop in post-quantum secure algorithms to instantiate the building blocks if new breakthroughs in quantum cryptanalysis on concrete algorithms are made.

## 7 An HMACCE Protocol with Stronger Security

In this section, we propose an HMACCE protocol called $\Pi_{\mathsf{FS}}$, which overcomes the limitations of $\Pi_{\mathsf{woFS}}$. The idea of the construction of this protocol is to use the authentication procedure as a compiler to transform a general passively secure two-message key exchange protocol to achieve HMACCE security. To realize our idea, we need to add one more authentication message to achieve mutual explicit authentication for both parties before the session key is used. Compared with $\Pi_{\mathsf{woFS}}$, the protocol $\Pi_{\mathsf{FS}}$ can achieve not only PFS but also the resilience of adaptive historical tag leakage. Also, $\Pi_{\mathsf{FS}}$ can still guarantee authentication and key exchange security when *all* tags are corrupted, but the historical data is not corrupted. It is because the session key in $\Pi_{\mathsf{FS}}$ no longer depends on the tags.

**Protocol Description.** In this protocol, one more primitive is needed, i.e., a $\mathsf{TKE}$ protocol with parameters $pms \leftarrow \mathsf{TKE.Setup}(1^\kappa)$. We assume that the randomness space of $\mathsf{TKE}$ is $\mathcal{R}_{\mathsf{TKE}} = \{0,1\}^{\ell_r}$. We depict the protocol $\Pi_{\mathsf{FS}}$ in Figure 4.

***Theorem* 2.** Suppose that the hash function $h$ is indistinguishable from a random oracle in time $t_h$ and with at most $q_h$ queries, and each data piece is unpredictable, the stateful length hiding authenticated encryption $\mathsf{StE}$ is $(t_{\mathsf{SLHAE}}, \epsilon_{\mathsf{SLHAE}})$-secure, and the two-message key exchange protocol $\mathsf{TKE}$ is $(t_{\mathsf{TKE}}, \epsilon_{\mathsf{TKE}})$-passively-secure. Then $\Pi_{\mathsf{FS}}$ is $(t', \epsilon_{\Pi_{\mathsf{FS}}}, q_l, \mathsf{FS})$-secure with $t' = (t_{\mathsf{SLHAE}} + t_{\mathsf{TKE}} + t_h) \approx t$, $\phi \leq q_l$, and $\epsilon_{\Pi_{\mathsf{FS}}} \leq \frac{\rho^2}{2^{\ell_r - 1}} + 18\rho\cdot(\frac{q_l}{L-z})^z + \frac{(18+6L+18\rho)\cdot q_h}{2^{\ell_p}} + (\rho^2+4\rho)\cdot\epsilon_{\mathsf{TKE}} + (\rho^2+12\rho)\cdot\epsilon_{\mathsf{SLHAE}}$.

Similarly, we prove Theorem 2 via the following two lemmas. Lemma 3 bounds the success probability $\epsilon_{\mathsf{auth}}$ of authentication-adversaries, and Lemma 4 bounds the advantage $\epsilon_{\mathsf{enc}}$ of encryption-adversaries. Then we have $\epsilon_{\Pi_{\mathsf{FS}}} \leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{enc}}$.

In the following, we just present the outline of the proof.

***Lemma* 3.** For any adversary $\mathcal{A}$ running in time $t' \approx t$, the probability that there exists an oracle $\Pi^u_{\mathsf{ID}^*}$ that accepts maliciously is at most $\frac{\rho^2}{2^{\ell_r}} + 2\rho\cdot\epsilon_{\mathsf{TKE}} + 9\rho\cdot(\frac{q_l}{L-z})^z + \frac{(9+3L+9\rho)\cdot q_h}{2^{\ell_p}} + 6\rho\cdot\epsilon_{\mathsf{SLHAE}}$.

***Lemma* 4.** For any adversary $\mathcal{A}$ running in time $t' \approx t$, the advantage of an adversary $\mathcal{A}$ which answers encryption-challenge correctly better than a random guess is at most $\frac{\rho^2}{2^{\ell_r}} + 9\rho\cdot(\frac{q_l}{L-z})^z + \frac{(9+3L+9\rho)\cdot q_h}{2^{\ell_p}} + (\rho^2+2\rho)\cdot\epsilon_{\mathsf{TKE}} + (\rho^2+6\rho)\cdot\epsilon_{\mathsf{SLHAE}}$.

Basically, the proof of this theorem can be extended from the proof of Theorem 1. We outline our proof idea as follows. The authentication messages $M_1$ and $M_2$ are computed in a similar way as $M$ in $\Pi_{\mathsf{woFS}}$, therefore we can reduce the authentication security regarding $M_1$ and $M_2$ in a similar way as the proof of Theorem 1 when the tags leakage is below a threshold. The advantage of an adversary $\mathcal{A}$ breaking the authentication of $\Pi_{\mathsf{FS}}$ is twice of breaking the authentication of $\Pi_{\mathsf{woFS}}$. Also, the random values $r_1$ and $r_2$ in $\Pi_{\mathsf{woFS}}$ are replaced with $m_1$ and $m_2$ in $\Pi_{\mathsf{FS}}$, because of the security of the $\mathsf{TKE}$ protocol [31, Lemma1].

Moreover, if there is no adversary that can break the authentication property of $\Pi_{\mathsf{FS}}$, then there would be only a passive adversary between the test oracle and its partner oracle (which must exist due to the explicit authentication messages $M_1$ and $M_2$). This fact enables us to reduce the channel confidentiality of $\Pi_{\mathsf{FS}}$ to the security of $\mathsf{TKE}$. We present the formal security reduction in [42].

| client $\mathsf{id_C}$ | | server $\mathsf{id_S}$ |
|---|---|---|
| The **Initialization** and **Tag Generation** phases are identical to those of $\Pi_{\mathsf{woFS}}$ | | |
| Online Authentication and Key Exchange | | |
| Sample $z$ distinct random indices: | | Sample $z$ distinct random indices: |
| $\mathsf{I}_C := (i_1, i_2, \ldots, i_z) \xleftarrow{\$} \mathsf{SI}$ | | $\mathsf{I}_S :\xleftarrow{\$} \mathsf{SI} \backslash \mathsf{I}_C$ |
| $\tilde{r_1} \xleftarrow{\$} \{0,1\}^{\ell_r}$ | | $\tilde{r_2} \xleftarrow{\$} \{0,1\}^{\ell_r}$ |
| $m_1 \leftarrow \mathsf{TKE.MSG}(\mathsf{id_C}, \tilde{r}_1, \emptyset)$ | $\xrightarrow{\mathsf{I}_C, m_1}$ | $m_2 \leftarrow \mathsf{TKE.MSG}(\mathsf{id_S}, \tilde{r}_2, m_1)$ |
| $\mathsf{I} = \mathsf{I}_C \cup \mathsf{I}_S$ | $\xleftarrow{\mathsf{I}_S, X, m_2, M_2}$ | $\mathsf{I} = \mathsf{I}_C \cup \mathsf{I}_S$ |
| $\mathsf{sid} := \mathsf{id_C}\|\|\mathsf{id_S}\|\|X\|\|m_1\|\|m_2\|\|\mathsf{I}$ | | for $i \in \mathsf{I}$: $(h(d_i\|\|i), t_i) \leftarrow \mathcal{D}_1 \& \mathcal{D}_2$ |
| $K_{\mathsf{I}} := \sum_{i \in \mathsf{I}} h(K\|\|i)$ | | $X := \sum_{i \in \mathsf{I}} h(d_i\|\|i)$ |
| $Y' := K_{\mathsf{I}} + K \cdot X$ | | $Y := \sum_{i \in \mathsf{I}} t_i$ |
| $M_2' := h(mk\|\|Y'\|\|\mathsf{sid}\|\|\text{‘Auth’}\|\|2)$ | | $\mathsf{sid} := \mathsf{id_C}\|\|\mathsf{id_S}\|\|X\|\|m_1\|\|m_2\|\|\mathsf{I}$ |
| reject if $M_2 \neq M_2'$ | | $M_2 := h(mk\|\|Y\|\|\mathsf{sid}\|\|\text{‘Auth’}\|\|2)$ |
| $K^{ke} \leftarrow \mathsf{TKE.SKG}(\mathsf{id_C}, \tilde{r}_1, \mathsf{id_S}, m_2)$ | | $M_1' := h(mk\|\|Y\|\|\mathsf{sid}\|\|\text{‘Auth’}\|\|1)$ |
| $M_1 := h(mk\|\|Y'\|\|\mathsf{sid}\|\|\text{‘Auth’}\|\|1)$ | $\xrightarrow{M_1}$ | reject if $M_1 \neq M_1'$ |
| | | $K^{ke} \leftarrow \mathsf{TKE.SKG}(\mathsf{id_S}, \tilde{r}_2, \mathsf{id_C}, m_1)$ |
| accept $K_s := h(K^{ke}\|\|\mathsf{sid})$ | | accept $K_s := h(K^{ke}\|\|\mathsf{sid})$ |

Fig. 4: An HMACCE Protocol $\Pi_{\mathsf{FS}}$ with Perfect Forward Secrecy.

**Resilience against Quantum Attacks.** Similar to $\Pi_{\mathsf{woFS}}$, the security of $\Pi_{\mathsf{FS}}$ can be reduced to the underlying hash function, stateful length hiding authenticated encryption, and two-message key exchange protocol (TKE). Unlike the two symmetric key cryptographic primitives, some TKE protocols do suffer dramatically from the attacks of quantum computers. For example, elliptic curve based TKE can be broken by Shor's algorithms [45]. However, post-quantum secure TKE also exists (e.g., based on ideal lattices [46]), and it can be instantiated in a post-quantum secure $\Pi_{\mathsf{FS}}$.

## 8 Security Enhancement Service for Legacy Devices

In this section, we show an important practical aspect of our proposed protocols, i.e., they can strengthen the security of existing legacy devices as an independent security enhancement service (i.e., without modifying the original legacy devices).

Here we consider a legacy device that has a symmetric key $mk$ shared with the server (i.e., the first authentication factor in our scheme). In case the legacy devices do not have an ACCE built in, it becomes trivial for us to enhance their security. We can simply add a new device like what the authors did in [2] to intercept the traffic of legacy devices and run the complete HMACCE protocols with the server. In this way, almost any ACCE/AKE protocols can be legacy-compliant. However, the practical difficulty is how to be compatible with legacy devices that run common ACCE protocols. A common (toy) ACCE solution deployed on a legacy device can be that two parties generate the session key (or the authentication message) in a form $K_s := h(mk, r_C\|\|r_S\|\|aux)$, where $r_C$ and $r_S$ are nonces selected by the client and the server respectively, and $aux$ may contain other protocol messages if any (e.g., Diffie-Hellman public keys). Our HMACCE protocols can be simply adapted to enhance the security of such a legacy device with the above toy ACCE without modifying its original operations. To deploy our protocol, a separate secure device, storing the tag key $K$ of the client, is directly and securely connected to the legacy device (e.g., via local LAN cables). After the new device executes our HMACCE protocol steps except for the session key generation, it only needs to send the secret hash value $H(Y\|\|sid\|\|\text{‘SeK’})$ to the legacy device as the $r_S$ in the toy ACCE scheme. The server can compute the same session key in exactly the same way. Meanwhile, we can choose to drop the explicit authentication message $M$ in our protocol depending on whether the legacy protocol has explicit message authentication steps[7]. In the last step of $\Pi_{\mathsf{FS}}$, the secret key is derived as $K_s := h(K^{ke}\|\|\mathsf{sid})$. If we want to adapt this protocol to legacy devices, the secret key will be $K_s := h(mk\|\|K^{ke}\|\|\mathsf{sid})$, and the proxy device can help compute $h(K^{ke}\|\|\mathsf{sid})$ and send the result to the legacy device as $r_S$.

To apply the above security enhancement in practice, we only need to check whether the legacy device runs an ACCE protocol in the above form of the toy example. One famous protocol instance meeting our requirement is the Transport Layer Security (TLS) Protocol with pre-shared key cipher-suits [47], [8], which are proposed for power-constrained devices (such as EMV card). For example, our first protocol $\Pi_{\mathsf{woFS}}$ can be used to enhance the security of TLS_PSK, and the second protocol $\Pi_{\mathsf{FS}}$ is suitable for TLS_DHE_PSK, where TLS_PSK uses only symmetric key (PSK) for authentication, and TLS_DHE_PSK uses a Diffie-Hellman exchange authenticated with a pre-shared key. Besides, the TLS protocols have explicit authentication steps.

## 9 Comparison

Here we briefly compare our proposed schemes with recent lightweight authenticated key exchange protocols, i.e., He et al. [13], Challa et al. [11], AuCPace [25], and Yang et al. [49]. Although some of these protocols are designed for a three-party scenario, two-party AKE procedures are also involved. We compare these four protocols from the following perspectives: (i) authentication factors, (ii) main

[7] The CWZT scheme is not legacy-compliant since the computation of $Y$ needs two authentication factors, so it should be deployed in one device where both authentication factors are stored together.

| Protocol | Authentication Factors | Security Properties | | | | Comm. Passes | Comm. Bits | Computation Cost |
|---|---|---|---|---|---|---|---|---|
| | | M-Auth | IA-SKS | B-Leak | PFS | | | |
| He et al.[13] | PW+LSK | × | √ | × | √ | 2 | 5888 | 21H+4MUL |
| Challa et al.[11] | Bio+PW+LPK | × | √ | × | × | 2 | 2528 | 1FE+14MUL+12H |
| AuCPace [25] | PW | √ | √ | × | √ | 8 | 2176 | 14H +7MUL |
| Yang et al.[48] | PW +LSK | √ | √ | × | √ | 3 | 5376 | 27H |
| $\Pi_{\mathsf{woFS}}$ | LSK+HT+HD | √ | √ | √<br>(static) | × | 2 | 3992/5656* | 326H+ 1AED |
| $\Pi_{\mathsf{FS}}$ | LSK+HT+HD | √ | √ | √<br>(adaptive) | √ | 3 | 4748/6493* | 328H+4MUL+1AED |

TABLE 1: Comparison. * denotes the numbers of transmitted bits for 80/128 bits security levels, respectively.

security properties, (iii) number of communication passes, (iv) bits transmitted in the protocol, and (iv) computation cost. To compare the computational cost, we instantiate our protocol $\Pi_{\mathsf{FS}}^{RO}$ with the elliptic curve cryptography (ECC) based Diffie-Hellman key exchange protocol (as the other compared protocols). To show the number of bits transmitted in the authentication phase of the protocols, we use the parameters in Section 10. Moreover, we let 'FE' denote a fuzzy extractor operation to obtain a secret from biometrics. Let 'M-Auth' denote mutual authentication, 'B-Leak' denote bounded leakage, and 'IA-SKS' denote implicit authentication with session key security. To compare the computation cost, let 'H' denote a hash operation, 'MUL' denote an ECC multiplication, and 'AED' denote the SLHAE encryption and decryption. Let 'Bio' denote the biometric authentication factor, 'PW' denote the password, 'HD' denote the historical data, 'HT' denote historic tags, 'LSK' denote the long-term symmetric key, and 'LPK' denote the long-term public key. We summarize the comparison in Table 1.

Although our protocols are less efficient than He et al. and Yang et al. protocols, we provide one more security property, i.e., bounded-leakage resilience. Since a hash operation is not expensive, the overall performance of $\Pi_{\mathsf{woFS}}$ is still practical (as shown in Table 2) for constrained devices. Note that the size of the transmitted data is dominated by the bits representing $\mathsf{I}_C$ and $\mathsf{I}_S$. In case the bandwidth is highly constrained, one can possibly use a pseudorandom number generator to generate $\mathsf{I}_C$ and $\mathsf{I}_S$ on both sides, then only two seeds of the pseudorandom number generator need to be transmitted, instead of two sets of indices.

## 10 Experimental Results

**Implementation Parameters.** We consider the upper-bound of the sessions of each party to be $\rho = O(2^{30})$ in practice, $\frac{q_l}{L-z} \approx 1/2$, $q_h = 2^{30}$ and $L = 2^{15}$. In the following, we list the parameters used in our implementation of $\Pi_{\mathsf{woFS}}$ and $\Pi_{\mathsf{FS}}$ based on the corresponding security levels: (i) for the security level $\kappa = 80$, we use $\ell_r = 142$, $z = 114$, and $\ell_p = 144$; (ii) for the security level $\kappa = 128$, we use $\ell_r = 190$, $z = 163$, and $\ell_p = 193$.

**Experiments Setup.** We used one PC (with Intel Core i7-8565U processor) as a server, and a Raspberry Pi 3 plus (with a Quad Core A53 (ARMv8) 1.4GHz CPU and 1GB RAM) is taken as the proxy device on the client side. We used an 8-bit Arduino as the legacy device. Our implementation is based on MIRACL cryptographic library [50], where the hash function used is SHA256, and the $\mathsf{TKE}$ protocol used in our second protocol is the Diffie-Hellman key exchange protocol based on the 224-bit/256-bit standard elliptic curve (over $GF(p)$) provided by MIRACL. The stateful length-hiding authenticated encryption is instantiated by 256-bit AES-GCM-SIV. The computation on the legacy devices is based on Arduino Cryptography Library [51].[8]

**Performance Evaluation.** We first measured the tag generation time on the client. It takes 6.21 $ms$ to generate a tag and encrypt the data and tag, assuming the data size is 1KB. On the server side, due to the high efficiency of AES-NI, it only takes 0.85 $\mu s$ to decrypt one package sent from the client. Also, we measured the time consumed by the authentication procedure and the key generation procedure separately on the server, the client (proxy device), and the legacy device. Since the computation on legacy devices is just 2 or 3 hash functions, if all the computation on the client side is performed by the proxy device, we observe no noticeable difference in performance. The performance in Table 2 is reported in milliseconds. 'Auth' denotes the performance of all other steps in authentication, and 'KE' denotes the time for the ephemeral key and the session key generations. The performance bottleneck is clearly on the client side because it is a resource-constrained embedded system device, and it needs $2z$ hash operations for one authentication. However, even for 128 bits security, the latency on the client side in $\Pi_{\mathsf{woFS}}$ totals only 22.149 $ms$, which is efficient enough to be deployed in real-world applications.

## 11 Conclusions and Open Problems

In this paper, we have shown two ways to build multi-factor ACCE protocols based on historical data in the random oracle model. The proposed protocols are efficient enough for resource-constrained devices in CPS or IoT. In particular, the first protocol only requires a few hash operations on the client. One open problem worth solving in the future is how to construct an HMACCE protocol in the standard model. Its challenge is to generate a pseudorandom seed from the authentication tags. As another interesting future work, one can consider building threshold HMACCE (e.g., following the idea in [53]) to further enhance the security against the compromise of authentication credentials.

## References

[1] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[8] It might be interesting to evaluate our protocols (as future work) on other CPS devices with special hardware architecture (e.g., programmable logic controller [52]).

| | $\Pi_{\mathsf{woFS}}$ | | | $\Pi_{\mathsf{FS}}$ | | |
|---|---|---|---|---|---|---|
| | Server | Client | Legacy | Server | Client | Legacy |
| Auth | 0.167/0.241 | 14.102/20.043 | 2.106 | 2.483/4.824 | 63.894/106.474 | 4.210 |
| KE | 0.035/0.041 | 0.297/0.328 | 2.106 | 2.282/4.373 | 51.192/87.379 | 2.106 |

TABLE 2: The performance of the proposed HMACCE protocols for (80-bit security/128-bit security), measured in $ms$. The latency of the legacy device remain the same because it only computes 2 or 3 SHA256 evaluations in both cases.

[2] J. H. Castellanos, D. Antonioli, N. O. Tippenhauer, and M. Ochoa, "Legacy-compliant data authentication for industrial control system traffic," in *ACNS*. Springer, 2017, pp. 665–685.

[3] C. Jin, S. Valizadeh, and M. van Dijk, "Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems," in *ICPS*. IEEE, 2018, pp. 824–829.

[4] C. Jin, Z. Yang, M. van Dijk, and J. Zhou, "Proof of aliveness," in *Proceedings of the 35th Annual Computer Security Applications Conference*. ACM, 2019, pp. 1–16.

[5] K. A. R. Craig Trivelpiece and R. Campero, "Machine-to-machine and machine to cloud end-to-end authentication and security," October 2016, uS Patent 15/091,634.

[6] A. Esfahani, G. Mantas, R. Matischek, F. B. Saghezchi, J. Rodriguez, A. Bicaku, S. Maksuti, M. G. Tauber, C. Schmittner, and J. Bastos, "A lightweight authentication mechanism for M2M communications in industrial iot environment," *IEEE IoT-J*, vol. 6, no. 1, pp. 288–296, 2019.

[7] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "On the security of TLS-DHE in the standard model," in *CRYPTO*, ser. LNCS, vol. 7417. Springer, 2012, pp. 273–293.

[8] Y. Li, S. Schäge, Z. Yang, F. Kohlar, and J. Schwenk, "On the security of the pre-shared key ciphersuites of TLS," in *PKC*, ser. LNCS, vol. 8383. Springer, 2014, pp. 669–684.

[9] A. K. Das, S. Kumari, V. Odelu, X. Li, F. Wu, and X. Huang, "Provably secure user authentication and key agreement scheme for wireless sensor networks," *Security and Communication Networks*, vol. 9, no. 16, pp. 3670–3687, 2016.

[10] R. Zhang, Y. Xiao, S. Sun, and H. Ma, "Efficient multi-factor authenticated key exchange scheme for mobile communications," *IEEE Trans. Dependable Secur. Comput.*, vol. 16, no. 4, pp. 625–634, 2019.

[11] S. Challa, M. Wazid, A. K. Das, N. Kumar, G. R. Alavalapati, E. Yoon, and K. Yoo, "Secure signature-based authenticated key establishment scheme for future iot applications," *IEEE Access*, vol. 5, pp. 3028–3043, 2017.

[12] S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena, "Two-factor authentication with end-to-end password security," in *PKC*, ser. LNCS, vol. 10770. Springer, 2018, pp. 431–461.

[13] J. He, Z. Yang, J. Zhang, W. Liu, and C. Liu, "On the security of a provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks," *I. J. of Distributed Sensor Networks*, vol. 14, no. 1, pp. 1–11, 2018.

[14] N. Fleischhacker, M. Manulis, and A. Azodi, "A modular framework for multi-factor authentication and key exchange," in *SSR*, 2014, pp. 190–214.

[15] SIEMENS, "Simatic process historian." [Online]. Available: https://sie.ag/2RUJZOU

[16] A. C. Chan, "Efficient defence against misbehaving TCP receiver dos attacks," *Computer Networks*, vol. 55, no. 17, pp. 3904–3914, 2011.

[17] A. C. Chan, J. W. Wong, J. Zhou, and J. C. M. Teo, "Scalable two-factor authentication using historical data," in *ESORICS*, ser. LNCS, vol. 9878. Springer, 2016, pp. 91–110.

[18] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.

[19] Y. Aumann, Y. Z. Ding, and M. O. Rabin, "Everlasting security in the bounded storage model," *IEEE Trans. Info. Theory*, vol. 48, no. 6, pp. 1668–1680, 2002.

[20] S. Dziembowski, "Intrusion-resilience via the bounded-storage model," in *TCC*, ser. LNCS, vol. 3876. Springer, 2006, pp. 207–224.

[21] M. Bellare and P. Rogaway, "Entity authentication and key distribution," in *CRYPTO*, ser. LNCS, vol. 773. Springer, 1993, pp. 232–249.

[22] J. Zhang, H. Zhong, J. Cui, Y. Xu, and L. Liu, "SMAKA: secure many-to-many authentication and key agreement scheme for vehicular networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 1810–1824, 2021.

[23] M. Wazid, A. K. Das, V. Odelu, N. Kumar, M. Conti, and M. Jo, "Design of secure user authenticated key management protocol for generic iot networks," *IEEE IoT-J*, vol. 5, no. 1, pp. 269–282, 2018.

[24] D. Chattaraj, M. Sarma, and A. K. Das, "A new two-server authentication and key agreement protocol for accessing secure cloud services," *Computer Networks*, vol. 131, pp. 144–164, 2018.

[25] B. Haase and B. Labrique, "Aucpace: Efficient verifier-based PAKE protocol tailored for the iiot," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2019, no. 2, pp. 1–48, 2019.

[26] Z. Yang, J. Lai, C. Liu, W. Liu, and S. Li, "Simpler generic constructions for strongly secure one-round key exchange from weaker assumptions," *Comput. J.*, vol. 60, no. 8, pp. 1145–1160, 2017.

[27] Z. Yang and J. Lai, "New constructions for (multiparty) one-round key exchange with strong security," *SCIENCE CHINA Information Sciences*, vol. 61, no. 5, pp. 059 102:1–059 102:3, 2018.

[28] X. Jia, D. He, L. Li, and K. R. Choo, "Signature-based three-factor authenticated key exchange for internet of things applications," *Multimed. Tools. Appl.*, vol. 77, no. 14, pp. 18 355–18 382, 2018.

[29] Y. Chen, L. López-Santidrián, J. Martínez, and P. Castillejo, "A lightweight privacy protection user authentication and key agreement scheme tailored for the internet of things environment: Lightpriauth," *J. Sensors*, vol. 2018, p. 7574238, 2018.

[30] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, "Generic compilers for authenticated key exchange," in *ASIACRYPT*, ser. LNCS, vol. 6477. Springer, 2010, pp. 232–249.

[31] Y. Li, S. Schäge, Z. Yang, C. Bader, and J. Schwenk, "New modular compilers for authenticated key exchange," in *ACNS*, ser. LNCS, vol. 8479. Springer, 2014, pp. 1–18.

[32] Z. Yang, C. Liu, W. Liu, S. Luo, H. Long, and S. Li, "A lightweight generic compiler for authenticated key exchange from non-interactive key exchange with auxiliary input," *I. J. Network Security*, vol. 18, no. 6, pp. 1109–1121, 2016.

[33] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *ACM CCS*. ACM, 1993, pp. 62–73.

[34] Y. Dodis, S. Guo, and J. Katz, "Fixing cracks in the concrete: Random oracles with auxiliary input, revisited," in *EUROCRYPT*, ser. LNCS, vol. 10211, 2017, pp. 473–495.

[35] K. G. Paterson, T. Ristenpart, and T. Shrimpton, "Tag size does matter: Attacks and proofs for the TLS record protocol," in *ASIACRYPT*, ser. LNCS, vol. 7073. Springer, 2011, pp. 372–389.

[36] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2001, pp. 453–474.

[37] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[38] D. Kogan, N. Manohar, and D. Boneh, "T/key: Second-factor authentication from secure hash chains," in *CCS*. ACM, 2017, pp. 983–999.

[39] Z. Yang, C. Jin, J. Ning, Z. Li, A. Dinh, and J. Zhou, "Group time-based one-time passwords and its application to efficient privacy-preserving proof of location," in *ACSAC*. ACM, 2021, pp. 497–512.

[40] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *International Conference on Critical Information Infrastructures Security*. Springer, 2016, pp. 88–99.

[41] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *CRYPTO*, ser. LNCS, vol. 10991. Springer, 2018, pp. 757–788.

[42] C. Jin, Z. Yang, S. Adepu, and J. Zhou, "HMAKE: legacy-compliant multi-factor authenticated key exchange from historical data," *IACR Cryptol. ePrint Arch.*, p. 450, 2019.

[43] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs+ signature framework," in *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, 2019, pp. 2129–2146.

[44] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 212–219.

[45] J. Proos and C. Zalka, "Shor's discrete logarithm quantum algorithm for elliptic curves," *arXiv preprint quant-ph/0301141*, 2003.

[46] J. Zhang, Z. Zhang, J. Ding, M. Snook, and Ö. Dagdelen, "Authenticated key exchange from ideal lattices," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 719–751.

[47] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," Internet Engineering Task Force (IETF), Tech. Rep., 2008.

[48] C. Wang, D. Wang, Y. Tu, G. Xu, and H. Wang, "Understanding node capture attacks in user authentication schemes for wireless sensor networks," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[49] Z. Yang, J. He, Y. Tian, and J. Zhou, "Faster authenticated key agreement with perfect forward secrecy for industrial internet-of-things," *IEEE Trans. Ind. Informatics*, vol. 16, no. 10, pp. 6584–6596, 2020.

[50] "Miracl cryptographic library," 2018. [Online]. Available: https://bit.ly/2MltKVG

[51] "Arduino cryptography library," 2017. [Online]. Available: https://github.com/rweather/arduinolibs

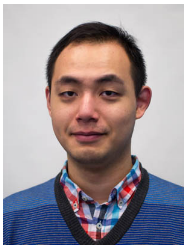
[52] Z. Yang, Z. Bao, C. Jin, Z. Liu, and J. Zhou, "Plcrypto: A symmetric cryptographic library for programmable logic controllers," *IACR Trans. Symmetric Cryptol.*, vol. 2021, no. 3, pp. 170–217, 2021.

[53] W. Li, H. Cheng, P. Wang, and K. Liang, "Practical threshold multi-factor authentication," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3573–3588, 2021.

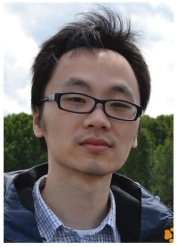
[54] K. G. Paterson, T. Ristenpart, and T. Shrimpton, "Tag size does matter: Attacks and proofs for the TLS record protocol," in *ASIACRYPT*, ser. LNCS, vol. 7073. Springer, 2011, pp. 372–389.

## Acknowledgment

This work is supported by supported by the National Key R&D Program of China under Grant No. 2022YFB3103500, the Natural Science Foundation of China under Grant Nos. (61872051, 62072062, and U20A20176), the Natural Science Foundation of Chongqing under Grant No. CSTB2022NSCQ-MSX0437, and the Fundamental Research Funds for the Central Universities under Grant No. SWU-KR22003. Chenglu Jin's work was done when he was with the University of Connecticut and partially supported by AFOSR MURI under award number FA9550-14-1-0351. We would like to thank SUTD iTrust for sharing the dataset of the SWaT testbed.



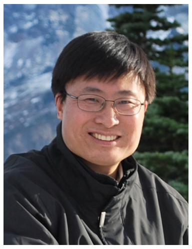
**Chenglu Jin** is a tenure-track researcher in the Computer Security Group at Centrum Wiskunde & Informatica (CWI Amsterdam), Netherlands. Prior to joining CWI Amsterdam, he was an assistant research professor at New York University. He got his Ph.D. degree from the University of Connecticut in 2019. He also holds a Master's degree and a Bachelor's degree from New York University and Xidian University, respectively. His research interests are hardware security, cyber-physical system security, and applied cryptography.



**Zheng Yang** received his Ph.D. degree from Horst Görtz Institute for IT Security, Ruhr-University Bochum, in 2013. He is currently a Professor with the College of Computer and Information Science, Southwest University, China. He was a post-doc researcher with the University of Helsinki, and the Singapore University of Technology and Design. His main research interests include information security, cryptography, and privacy.

**Tao Xiang** received the B.Eng., M.S., and Ph.D. degrees in computer science from Chongqing University, Chongqing, China, in 2003, 2005, and 2008, respectively. He is currently a Professor with the College of Computer Science, Chongqing University. He has published over 90 papers on international journals and conferences. He also served as a referee for numerous international journals and conferences. His research interests include multimedia security, cloud security, and privacy and cryptography.

**Sridhar Adepu** is a Lecturer with the University of Bristol. He got his a PhD degree from the Information Systems Technology and Design pillar at the Singapore University of Technology and Design (SUTD). His research focuses on security of cyber-physical systems. Prior to his PhD study, Sridhar was a research assistant at iTrust, SUTD. Sridhar holds a Masters degree in Computer Science from National Institute of Technology Rourkela, India.



**Jianying Zhou** is a full professor at Singapore University of Technology and Design (SUTD), and Co-Center Director for iTrust. Before joining SUTD, he was a principal scientist and the head of Infocomm Security Department at Institute for Infocomm Research, A*STAR. Prof. Zhou received PhD in Information Security from Royal Holloway, University of London. His research interests are in applied cryptography and network security, cyber-physical system security, mobile and wireless security. Prof. Zhou is a co-founder & steering committee co-chair of International Conference on Applied Cryptography and Network Security (ACNS). He is also steering committee chair of ACM AsiaCCS, and steering committee member of Asiacrypt and ISC.

```
$\mathsf{ENC}(m_0, m_1, \ell_c, H)$:
$cnt_e := cnt_e + 1$
$(C^{(0)}, st_e^{(0)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \ell_c, H, m_0, st_e)$
$(C^{(1)}, st_e^{(1)}) \xleftarrow{\$} \mathsf{StE.Enc}(k, \ell_c, H, m_1, st_e)$
If $C^{(0)} = \perp$ or $C^{(1)} = \perp$ then return $\perp$
$(C_u, H_u, st_e) := (C^{(b)}, H, st_e^{(b)})$
Return $C_u$
```

```
$\mathsf{DEC}(C, H)$:
$cnt_d := cnt_d + 1$
If $b = 0$, then return $\perp$
$(m, st_d) = \mathsf{StE.Dec}(k, H, C, st_d)$
If $cnt_d > cnt_e$ or $C \neq C_{cnt_d}$ or $H \neq H_{cnt_d}$, then phase := 1
If phase = 1 then return $m$
Return $\perp$
```

Fig. 5: ENC and DEC Oracles for the Stateful LHAE Security Experiment.

## APPENDIX A
## FORMAL SECURITY DEFINITIONS OF CRYPTOGRAPHIC BUILDING BLOCKS

### A.1 Stateful Length-Hiding Authenticated Encryption

The security of SLHAE [54] is formalized in the following security game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$. Figure 5 describes how the oracles ENC and DEC respond to $\mathcal{A}$'s queries. The values $cnt_e$, $cnt_d$, and phase are all initialized to 0 at the beginning of the security game.

- The challenger $\mathcal{C}$ selects $b \xleftarrow{\$} \{0, 1\}$, $k \xleftarrow{\$} \mathcal{K}_{\mathsf{SLHAE}}$, sets $st_e := \emptyset$, and $st_d := \emptyset$,
- $\mathcal{A}$ may adaptively query the encryption oracle ENC $q_e$ times and the decryption oracle DEC $q_d$ times.
- Finally, $\mathcal{A}$ outputs its guess $b' \in \{0, 1\}$.

***Definition 2.*** We say that the stateful symmetric encryption scheme $\mathsf{StE} = (\mathsf{StE.Enc}, \mathsf{StE.Dec})$ is $(t, \epsilon_{\mathsf{SLHAE}})$-*secure* , if any adversary running in time $t$ has an advantage of at most $\epsilon_{\mathsf{SLHAE}}$ to output $b'$ such that $b' = b$, i.e., $|\Pr[b' = b] - 1/2| \leq \epsilon_{\mathsf{SLHAE}}$, while the number of allowed queries $(q_e + q_d)$ is upper bounded by $t$.

### A.2 Passively Secure Two-message Key Exchange

We say that the TKE.SKG algorithm is correct, if for all random values $r_{\mathsf{id}_1}, r_{\mathsf{id}_2} \xleftarrow{\$} \mathcal{R}_{\mathsf{TKE}}$ and messages $m_{\mathsf{id}_1} \xleftarrow{\$} \mathsf{TKE.MSG}(\mathsf{id}_1, r_{\mathsf{id}_1}, \emptyset)$ and $m_{\mathsf{id}_2} \xleftarrow{\$} \mathsf{TKE.MSG}(\mathsf{id}_2, r_{\mathsf{id}_2}, m_{\mathsf{id}_1})$, it holds that $\mathsf{TKE.SKG}(\mathsf{id}_1, r_{\mathsf{id}_1}, \mathsf{id}_2, m_{\mathsf{id}_2}) = \mathsf{TKE.SKG}(\mathsf{id}_2, r_{\mathsf{id}_2}, \mathsf{id}_1, m_{\mathsf{id}_1})$.

We define a security experiment for passively secure TKE protocols as follows.

**Security Experiment**: The security experiment is carried out as a game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$ based on a protocol TKE. During the setup phase, $\mathcal{C}$ generates the parameters $pms \leftarrow \mathsf{TKE.Setup}(1^\kappa)$ and two identities $\{\mathsf{ID}_1, \mathsf{ID}_2\}$ of protocol participants. The adversary is given $pms$ and all identities as input. Next, $\mathcal{A}$ will interact with $\mathcal{C}$ via asking at most $d \in \mathbb{N}$ times $\mathsf{Execute}(\mathsf{id}_1, \mathsf{id}_2)$ query; for each Execute query, $\mathcal{C}$ runs a fresh protocol instance between $\mathsf{id}_1$ and $\mathsf{id}_2$, and returns the corresponding protocol messages' transcript $T$ and session key $K$ to $\mathcal{A}$. At some point, $\mathcal{A}$ submits a challenge request $\bowtie$. Upon receiving $\bowtie$, $\mathcal{C}$ runs a new protocol instance obtaining the transcript $T^*$ and the session key $K_1^*$, samples a random key $K_0^*$, and tosses a fair coin $b \in \{0, 1\}$. Then, $\mathcal{C}$ returns $(T^*, K_b^*)$ to $\mathcal{A}$. After the challenge query, $\mathcal{A}$ may continue making $\mathsf{Execute}(\mathsf{id}_1, \mathsf{id}_2)$ queries. Finally, $\mathcal{A}$ may terminate and output a bit $b'$.

***Definition 3.*** We say that a two-message key exchange protocol TKE is $(t, \epsilon_{\mathsf{TKE}})$-*passively-secure* if for all probabilistic polynomial time (PPT) adversaries running the above experiment in time $t$, it holds that $|\Pr[b = b'] - 1/2| \leq \epsilon_{\mathsf{TKE}}$.