HMAKE: Legacy-Compliant Multi-factor Authenticated Key Exchange from Historical Data

Chenglu Jin[1], Zheng Yang[2]*, Sridhar Adepu[2], and Jianying Zhou[2]

[1]University of Connecticut
chenglu.jin@uconnn.edu
[2]Singapore University of Technology and Design
{zheng_yang,adepu_sridhar,jianying_zhou}@sutd.edu.sg

Abstract. In this paper, we introduce two lightweight historical data based multi-factor authenticated key exchange (HMAKE) protocols in the random oracle model. Our HMAKE protocols use a symmetric secret key, as their first authentication factor, together with their second authentication factor, historical data exchanged between the two parties in the past, and the third authentication factor, a set of secret tags associated with the historical data, to establish a secure communication channel between the client and the server.
A remarkable security feature of HMAKE is bounded historical tag leakage resilience, which means that (informally speaking) if a small portion of the secret tags is leaked to an adversary, it will not affect the security of one HMAKE protocol with an overwhelming probability. Our first HMAKE protocol can provide *static* bounded leakage resilience, meaning that the secret tags are leaked at the beginning of the security game. To enhance its security, our second HMAKE protocol makes use of our first protocol as a compiler to transform any passively secure two-message key exchange protocol to an *actively* secure HMAKE protocol with perfect forward secrecy, and therefore it can be secure even if the historical tags are compromised *adaptively* by an attacker.
In addition to the strong security properties we achieved, our protocols can potentially have great impacts in practice: they are efficient in computation, and they are compatible with legacy devices in cyber-physical systems.

Keywords: Historical Data, Authentication, Authenticated Key Exchange, Security Model, Multi-Factor Authentication

1 Introduction

Cyber-Physical Systems (CPSs), like water treatment systems and nuclear plants, are critical for the daily life of millions of people. However, the security of this kind of systems is always an afterthought, which opens a tremendous attacking surface on CPSs for malicious adversaries [21]. 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

* Chenglu Jin and Zheng Yang contribute equally and share the first authorship. Zheng Yang is the corresponding author.

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 servers is usually called supervisory control and data acquisition (SCADA) systems.

In the existing literature, many end-to-end encryption and message authentication methods are suggested between controllers and SCADA system [5, 29], but none of them answered the question about how to establish such a secure communication channel. Of course, one can simply use a single factor authenticated key exchange protocol [13, 20], but can we enhance its security by introducing another authentication factor? Because it is a machine-to-machine (M2M) authentication, the existing two/multi-factor authenticated key exchange (AKE) protocols [14, 42, 6, 27, 24], 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 [22] 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 while being able to satisfy leakage resilience. We have to find another authentication factor on the server, and it should have a stronger security level from a conventional secret key stored on the same machine.

Recall that CPS devices keep sending data to the SCADA system for monitoring. Actually, for future data analysis, the historical data in most of the SCADA systems is stored in a dedicated process historian, instead of their main servers [36]. 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. As another fact, the historical data and its tag 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 an impersonation even harder.

Existing Historical Data based Authentication Protocols. The usage of historical data as an authentication factor in an authentication protocol was introduced in [8] and further developed in [9] at ESORICS'16. The early scheme [8] 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. At ESORICS'16, Chan et al. [9] introduced a scalable historical data based two-factor authentication scheme (which will be referred to as 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 [35], 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 secrets (e.g., 512 bits), which well fits CPS devices. Chan et al. also introduced historical tag leakage resilience in a bounded-storage model [1, 19] 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. Notice 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 [9]. (2) In [9], the authors suggested to use 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 is still an open problem.

Our Contributions. Due to the vulnerabilities and limitations of the existing protocols mentioned above, we cannot simply extend the existing authentication protocols to an AKE protocol. We have to reconsider the fundamental authentication problem based on historical data, and redesign a new AKE protocol from scratch. More specifically, we made four significant contributions as follows:

1. We analyze the stat-of-the-art historical data based authentication protocol (the CWZT protocol [9] proposed at ESORICS'16) and propose a tag-stealing attack which breaks the security claim of the CWZT protocol via legitimate interactions.
2. To build a solid theoretical foundation of our proposed HMAKE protocols, and to avoid repeating the mistakes in the previous designs, we are the first to formally define two indistinguishability-based security models for HMAKE, and later we analyze our proposed protocols in these security models.
3. As one of the main contributions of this paper, we introduce two HMAKE protocols Π_{woFS} (without forward secrecy) and Π_{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 to deploy our protocols in the field to enhance legacy devices. Also, we implemented Π_{woFS} and Π_{FS}, and evaluated their performance experimentally.

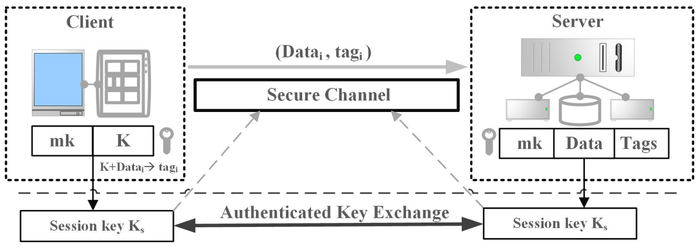

Fig. 1: Overview of Our HMAKE Protocol

Technical Overview. An overview of our first HMAKE protocol Π_{woFS} is presented in Fig. 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 would generate 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 the historical data. In our HMAKE protocols, both parties can use their authentication credentials to run the key exchange procedure to generate a session key to establish a secure channel, that is used to protect the underlying data and tag transmission.

In addition, Π_{woFS} still has a remarkable security feature called historical tag leakage resilience, such that a small portion of tag leakage will not affect the security of Π_{woFS} much. Notice that although this feature was first introduced in [9], they failed to achieve it due to the tag stealing attack we introduced. Also, because of the clear separation of the two authentication factors in Π_{woFS}, Π_{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 Π_{woFS}.

One limitation of Π_{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 HMAKE in practice since the leaked tags will be out-dated 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 [2]. To achieve this adaptive bounded-leakage resilience and perfect forward secrecy, we design the second HMAKE protocol Π_{FS}. In Π_{FS}, we use the first protocol Π_{woFS} as a compiler to transform any passively secure two-message key exchange (TKE) protocols to be an actively secure HMAKE protocol. Because the session key does not depend on the authentication keys (unlike Π_{woFS}), Π_{FS} can resist adaptive bounded-leakage, i.e., the adversary can get access to a bounded number of valid historical tags at any time of 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 develop lightweight multi-factor AKE protocols in conjunction with specific communication models or application scenarios. For example, the lightweight multi-factor AKE protocols proposed in [14, 24] are designed for wireless sensor networks (WSN), and there are many protocols [6, 37] for Internet-of-Things (IoT). Recently, Dua et al. [18] proposed a protocol to protect the communication of vehicles in smart cities. In [10], 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 would be the long-term symmetric key and users' memorable password. Most of the long-term symmetric key based lightweight AKE schemes, e.g.,[14, 6, 24, 40], require some tamper-proof devices (such as smart cards) to store the authentication key. To enhance its security, a protocol might also incorporate biometric factors [14, 6] into authentication that has more entropy than a password. However, none of the above lightweight AKE protocol covers the leakage resilient property as our proposals.

Cryptographic Primitives for PFS. Considering the importance of PFS, many AKE schemes are proposed with PFS based on Diffie-Hellman key exchange (DHKE), e.g., [14, 6, 39, 38, 24]. Fortunately, some results (e.g., [28, 12]) have shown that the DHKE protocol might be feasible to be realized with the elliptic curves cryptography (ECC) optimized for embedded systems. We also instantiate our protocol $\mathit{\Pi}_{\mathsf{FS}}$ with ECC based DHKE protocol for comparison.

Generic AKE Compilers. A research line related to our second protocol $\mathit{\Pi}_{\mathsf{FS}}$ is the AKE compiler that is to securely combine authentication protocols (AP) with passively secure key exchange protocols (KE) in a modular and generic manner, e.g., [25, 30, 22, 41]. However, no existing AKE compilers leverage historical data based authentication protocol as a building block. Our protocol $\mathit{\Pi}_{\mathsf{FS}}$ presents a new way to realize AKE compilers.

3 Preliminaries

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

Random Oracles. Bellare and Rogaway [3] 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 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 just returns a true random value r_m from the output space, and records the tuple (m, r_m) into its query list HL; (ii) If $m \in \mathsf{HL}$, then the oracle returns its associated random value r_m recorded in HL.

As in [16], we use a uniformly random salt $\chi \xleftarrow{\$} \mathcal{X}$ as input of h to sample a random oracle $h(\chi, \cdot)$, where $\mathcal{X}$ is a 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 [16].

Passively Secure Two-message Key Exchange. We consider a two-message key exchange (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 player of the protocol does not hold any long-term secret key for simplicity. Specifically, a general 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^κ, the setup algorithm outputs pms, a set of system parameters. We assume the other algorithms may implicitly use pms.
- $m_{\mathsf{id}_1} \xleftarrow{\$} \mathsf{TKE.MSG}(\mathsf{id}_1, r_{\mathsf{id}_1}, m_{\mathsf{id}_2})$: The message generation algorithm takes as input a party's identity id_1, a randomness $r_{\mathsf{id}_1} \xleftarrow{\$} \mathcal{R}_{\mathsf{TKE}}$ and a message $m_{\mathsf{id}_2} \in \mathcal{M}_{\mathsf{TKE}}$ received from party 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 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 an input the participants' identities id_1 and id_2, the randomness r_{id_1} and the received message m_{id_2} from party id_2, and outputs a session key $K \in \mathcal{K}_{\mathsf{TKE}}$, where $\mathcal{K}_{\mathsf{TKE}}$ is the session key space.

We say that the $\mathsf{TKE.SKG}$ algorithm is correct, if for all random values $r_{\mathsf{id}_1}, r_{\mathsf{id}_2} \overset{\$}{\leftarrow} \mathcal{R}_{\mathsf{TKE}}$ and all messages $m_{\mathsf{id}_1} \overset{\$}{\leftarrow} \mathsf{TKE.MSG}(\mathsf{id}_1, r_{\mathsf{id}_1}, \emptyset)$ and $m_{\mathsf{id}_2} \overset{\$}{\leftarrow} \mathsf{TKE.MSG}(\mathsf{id}_2, r_{\mathsf{id}_2}, m_{\mathsf{id}_1})$, 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 $\{\mathcal{ID}_1, \mathcal{ID}_2\}$ 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 $\mathsf{Execute}$ query, $\mathcal{C}$ runs a fresh protocol instance between id_1 and 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 $\ltimes$. Upon receiving $\ltimes$, $\mathcal{C}$ 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 1. *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*
$|\Pr[b = b'] - 1/2| \le \epsilon_{\mathsf{TKE}}$.

4 Cryptanalysis of the CWZT Scheme

In this section, we revisit the security property of CWZT scheme [9, §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 the 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 κ bits. The CWZT protocol makes use of two pseudorandom functions $f : \{0,1\}^\kappa \times \{0,1\}^* \to \mathbb{Z}_p$ and $E : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$, and a cryptographic hash function $h : \{0,1\}^* \to \mathbb{Z}_p$. The protocol running between a verifier id_C and a prover id_S is shown in Fig. 2.

$$\begin{array}{rcl}
\text{Verifier } \mathsf{id_C} & & \text{Prover } \mathsf{id_S} \\ \hline
\multicolumn{3}{c}{\text{Initialization}} \\
sk^1_{\mathsf{id_C},\mathsf{id_S}} = mk \xleftarrow{\$} \{0,1\}^\kappa & \xrightarrow{sk^1_{\mathsf{id_C}}} & sk^1_{\mathsf{id_S},\mathsf{id_C}} := sk^1_{\mathsf{id_C},\mathsf{id_S}} \\
K \xleftarrow{\$} \mathbb{Z}_p,\ K' \xleftarrow{\$} \{0,1\}^\kappa & \text{secure channel} & \\
L := 0 & & \\
sk^2_{\mathsf{id_C},\mathsf{id_S}} = (K, K') & & sk^2_{\mathsf{id_S},\mathsf{id_C}} = \mathcal{D} = \emptyset \\ \hline
\multicolumn{3}{c}{\text{Tag Generation: for the } i\text{-th data}} \\
k_i := f(K', i) & \xrightarrow{i, d_i, t_i} & \\
t_i := K \cdot h(d_i) + k_i & & \\
L := L + 1 & & \text{store } (i, d_i, t_i) \to \mathcal{D} \\ \hline
\multicolumn{3}{c}{\text{Online Authentication}} \\
\text{Sample } z \text{ random indices:} & & \\
\mathsf{I} = (I_1, I_2, \ldots, I_z) \xleftarrow{\$} [L] & & r' := E(sk^1_{\mathsf{id_S},\mathsf{id_C}}, r) \\
r \xleftarrow{\$} \{0,1\}^\kappa & \xrightarrow{\mathsf{I}, r} & \text{for } i \in \mathsf{I}\text{: } (d_i, t_i) \leftarrow \mathcal{D} \\
r' := E(sk^1_{\mathsf{id_C},\mathsf{id_S}}, r) & & X := \sum_{i \in \mathsf{I}} f(r', i) \cdot h(d_i) \\
K_{\mathsf{I}} := \sum_{i \in \mathsf{I}} f(K', i) \cdot f(r', i) & \xleftarrow{X, Y} & Y := \sum_{i \in \mathsf{I}} f(r', i) \cdot t_i \\
Y' := K_{\mathsf{I}} + K \cdot X & & \\
\text{accept iff } Y' = Y & &
\end{array}$$

Fig. 2: The CWZT Protocol [9].

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 I_1 and 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 arbitrary index j. Note that this is allowed by the CWZT scheme [9].

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 I_1, such that $i^* \notin \mathsf{I}_1$ and $j \in \mathsf{I}_1$.
- $\mathcal{A}$ sends (I_1, r) to $\mathsf{id_S}$ in a session, and receives the authentication messages (X_1, Y_1).
- In another session, $\mathcal{A}$ chooses a selection set I_2 by replacing the index j with i^*, and sends (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 it wishes.

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

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

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 I can be malicious which implies that the 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 HMAKE constructions.

5 HMAKE Security Model

In this section, we define new indistinguishability-based security models for historical data based multi-factor authenticated key exchange protocols (HMAKE). In these security models, we will formulate the security goals that our upcoming HMAKE protocols can achieve. The new models basically follow from the security models for AKE in literature, e.g., [2, 22, 31, 11]. In contrast to previous models, we particularly formulate the authentication factors related to historical data, and the security property regarding leakage resilience.

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 $\mathcal{ID}$ be a general identity to denote one of the honest parties in $\{\mathsf{id_C}^*, \mathsf{id_S}^*\}$.[1] 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 authentication 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

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

$\alpha \in \{2, 3\}$ 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 $\mathcal{ID}$ can carry out at most $\rho \in \mathbb{N}$ sessions that are modeled by a set of oracles $\{\pi^u_{\mathcal{ID}} : i \in [\ell], u \in [\rho]\}$. All oracles can have access to the authentication keys of its owner. Moreover, we assume each oracle $\pi^u_{\mathcal{ID}}$ maintains a list of independent internal state variables: (i) $\Phi^u_{\mathcal{ID}}$ – session decision $\Phi^u_{\mathcal{ID}} \in \{\mathtt{accept}, \mathtt{reject}\}$; (ii) $\mathsf{pid}^u_{\mathcal{ID}}$ – identity of the intended communication partner; (iii) $K^u_{\mathcal{ID}}$ – session key of $\pi^u_{\mathcal{ID}}$; (iv) $T^u_{\mathcal{ID}}$ – protocol messages orderly sent and received by $\pi^u_{\mathcal{ID}}$. We assume that the session key $K^u_{\mathcal{ID}}$ will be assigned with a non-empty value if and only if $\Phi^u_{\mathcal{ID}} = \mathtt{accept}$. [2]

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:

- $\mathsf{Send}(\mathcal{ID}, u, m)$: The adversary can send any message m to the oracle $\pi^u_{\mathcal{ID}}$ via this query. Oracle $\pi^u_{\mathcal{ID}}$ will respond 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.
- $\mathsf{RevealKey}(\mathcal{ID}, u)$: The oracle $\pi^u_{\mathcal{ID}}$ responds with the contents of $K^u_{\mathcal{ID}}$.
- $\mathsf{Corrupt1}(\mathcal{ID}_1, \mathcal{ID}_2)$: For honest parties $(\mathcal{ID}_1, \mathcal{ID}_2) \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}$, this query returns the first authentication key $sk^1_{\mathcal{ID}_1,\mathcal{ID}_2}$ of an honest party $\mathcal{ID}_1$, and sets $cst^1_{\mathcal{ID}_1,\mathcal{ID}_2} = cst^1_{\mathcal{ID}_2,\mathcal{ID}_1} := \mathsf{exposed}$.
- $\mathsf{Corrupt2}(\mathcal{ID}_1, \mathcal{ID}_2)$: For honest parties $(\mathcal{ID}_1, \mathcal{ID}_2) \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}$, this query returns the second authentication key $sk^2_{\mathcal{ID}_1,\mathcal{ID}_2}$ of an honest party $\mathcal{ID}_1$, and sets $cst^2_{\mathcal{ID}_1,\mathcal{ID}_2} := \mathsf{exposed}$.
- $\mathsf{Corrupt3}$: This query returns the third authentication key $sk^3_{\mathsf{id_S}^*,\mathsf{id_C}^*}$, and sets $cst^3_{\mathsf{id_S}^*,\mathsf{id_C}^*} := \mathsf{exposed}$.
- $\mathsf{RevealR}(\mathcal{ID}, u)$: This query returns the randomness generated by $\pi^u_{\mathcal{ID}}$.
- $\mathsf{HTLeak}(i)$: 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.
- $\mathsf{Test}(\mathcal{ID}, u)$: If the oracle $\pi^u_{\mathcal{ID}}$ has $\Phi^u_{\mathcal{ID}} \neq \mathtt{accept}$, then the oracle returns a failure symbol $\perp$. Otherwise, it flips a random bit b, samples a random key K_0, and sets $K_1 = K^u_{\mathcal{ID}}$. Finally, the key K_b is returned. We call the oracle $\pi^u_{\mathcal{ID}}$ in this query as a *test oracle*.

[2] 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.

Secure HMAKE Protocols. We first review a notion called *matching conversations* that was first introduced in [2] to formulate the relation between two sessions. We will use a variant that is refined in [26].

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

Correctness. We say a HMAKE protocol Π 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 the variable $\mathsf{MN} \in \{\mathsf{FS}, \mathsf{woFS}\}$ to denote the HMAKE 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 a HMAKE protocol without PFS, we only define static historical tag leakage, explicit authentication for the server, and the implicit authentication for the client. However, for a HMAKE protocol with PFS, we define mutual explicit authentication and adaptive historical tag leakage.

HMAKE Security Experiment (Π, MN): A challenger $\mathcal{C}$ will play a game with an adversary $\mathcal{A}$ based on a target HMAKE protocol Π and the security variable MN. In the initialization phase of the game, $\mathcal{C}$ first implements a collection of oracles $\{\pi^u_{\mathcal{ID}} : \mathcal{ID} \in \{\mathsf{id_C}^*, \mathsf{id_S}^*\}, u \in [\rho]\}$ 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 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, and RegClient. 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. During the second phase, $\mathcal{A}$ may issue a $\mathsf{Test}(\mathcal{ID}, u)$ query at most once. After the Test query, $\mathcal{A}$ can keep asking other queries as it wishes. Eventually, $\mathcal{A}$ may terminate and output a bit b' as its guess for b in the Test query.

The difference between static and adaptive historical tag leakage is whether 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 the partial corruption is its complement.

Full Corruption. We define the full corruption of a party $\mathcal{ID} \in \{\mathsf{id_C}, \mathsf{id_S}\}$ 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 $\mathcal{ID}$ and 0 otherwise. $\mathsf{FullC}(\mathcal{ID}, \mathsf{id_C}, \mathsf{id_S}, q_l) = 1$ 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 $\mathcal{A}$ queried more than q_l HTLeak queries;
5. $\mathcal{ID} = \mathsf{id_C}$ and $cst^1_{\mathsf{id_S},\mathsf{id_C}} = cst^3_{\mathsf{id_S},\mathsf{id_C}} = \mathsf{exposed}$;
6. $\mathcal{ID} = \mathsf{id_C}$, $cst^1_{\mathsf{id_C},\mathsf{id_S}} = \mathsf{exposed}$ and $\mathcal{A}$ queried more than q_l HTLeak queries.

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

In the following security definition, we let $\mathcal{ID}^*$ denote the party that is submitted to the Test query, and let $\widetilde{\mathcal{ID}^*}$ denote the identity that is required to provide explicit authentication. That is, $\widetilde{\mathcal{ID}^*}$ denotes $\mathsf{id_S}^*$ when $\mathsf{MN} = \mathsf{woFS}$, and $\widetilde{\mathcal{ID}^*}$ denotes either $\mathsf{id_S}^*$ or $\mathsf{id_C}^*$ when $\mathsf{MN} = \mathsf{FS}$.

Definition 2 (HMAKE Security). *We say a PPT adversary $\mathcal{A}$ $(t, \epsilon, q_l, \mathsf{MN})$-breaks an HMAKE protocol Π 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 ϵ there exists an oracle $\pi^u_{\widetilde{\mathcal{ID}^*}}$ such that*
  - $\mathsf{FullC}(\widetilde{\mathcal{ID}^*}, \widetilde{\mathcal{ID}^*}, \mathsf{pid}^u_{\widetilde{\mathcal{ID}^*}}, q_l) = 0$ *when $\pi^u_{\widetilde{\mathcal{ID}^*}}$ accepts, and*
  - $\pi^u_{\widetilde{\mathcal{ID}^*}}$ *has no unique partner oracle at the party $\mathsf{pid}^u_{\widetilde{\mathcal{ID}^*}}$.*

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

  and then the probability b' equals to the bit b sampled in the Test query satisfies $|\Pr[b' = b] - 1/2| \geq \epsilon$. We say that $\mathcal{A}$ answers the session-key-challenge correctly *if $b' = b$ and all the above conditions are met.*

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

6 An Efficient HMAKE Protocol

In this section, we develop an efficient HMAKE Protocol in the random oracle model denoted by Π_{woFS}. The main construction idea of Π_{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 ℓ_p, and $\mathbb{Z}_p^* = \mathbb{Z}_p/\{0\}$. In our protocol, we need a cryptographic hash function $h : \{0,1\}^* \to \mathbb{Z}_p$. 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 ℓ_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 be unpredictable and have some min-entropy[3]. As stated in [4], any unpredictable string (regardless of its min-entropy) with

[3] 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

bit-length that is larger than ℓ_p, in the random oracle model, can be used to extract an unpredictable ℓ_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 SI be an 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 SI can be used at most ϕ times, so once they have been use ϕ times, we will refresh SI with the next L unused historical data and tags from $(\mathcal{D}_1, \mathcal{D}_2)$.

client $\mathsf{id_C}$		server $\mathsf{id_S}$
	Initialization	
$sk^1_{\mathsf{id_C},\mathsf{id_S}} = mk \xleftarrow{\$} \{0,1\}^\kappa$	$\xrightarrow{sk^1_{\mathsf{id_C}}}$	$sk^1_{\mathsf{id_S},\mathsf{id_C}} := sk^1_{\mathsf{id_C},\mathsf{id_S}}$
$sk^2_{\mathsf{id_C},\mathsf{id_S}} = K \xleftarrow{\$} \mathbb{Z}_p^*$	secure channel	$sk^2_{\mathsf{id_S},\mathsf{id_C}} = \mathcal{D}_1 = \emptyset$
$cnt := 0$		$sk^3_{\mathsf{id_S},\mathsf{id_C}} = \mathcal{D}_2 = \emptyset$
	Tag Generation: for the i-th data	
$k_i := h(K \Vert i)$	$\xrightarrow{i, d_i, t_i}$	
$t_i := K \cdot h(d_i \Vert i) + k_i$	secure channel	
$cnt := cnt + 1$		store $(i, t_i) \rightarrow \mathcal{D}_2$
	Online Authentication and Key Exchange	
Sample z distinct random indices:		Sample z distinct random indices:
$\mathsf{I}_C = (i_1, i_2, \ldots, i_z) \xleftarrow{\$} \mathsf{SI}$		$\mathsf{I}_S \xleftarrow{\$} \mathsf{SI} \backslash \mathsf{I}_C$
$r_1 \xleftarrow{\$} \{0,1\}^{\ell_r}$	$\xrightarrow{\mathsf{I}_C, r_1}$	$r_2 \xleftarrow{\$} \{0,1\}^{\ell_r}$
$\mathsf{I} = \mathsf{I}_C \cup \mathsf{I}_S$	$\xleftarrow{\mathsf{I}_S, r_2, X, M}$	$\mathsf{I} = \mathsf{I}_C \cup \mathsf{I}_S$
$\mathsf{sid} := \mathsf{id_C} \Vert r_1 \Vert \mathsf{id_S} \Vert X \Vert r_2 \Vert \mathsf{I}$		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 $K_s := h(mk \Vert Y' \Vert \mathsf{sid} \Vert \text{'SeK'})$		accept $K_s := h(mk \Vert Y \Vert \mathsf{sid} \Vert \text{'SeK'})$

Fig. 3: An Efficient HMAKE Protocol $\mathit{\Pi}_{\mathsf{woFS}}$.

The protocol $\mathit{\Pi}_{\mathsf{woFS}}$ running between a client $\mathsf{id_C}$ and a server $\mathsf{id_S}$ is shown in Fig. 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 $sk^1_{\mathsf{id_C},\mathsf{id_S}} = sk^1_{\mathsf{id_S},\mathsf{id_C}} := mk \xleftarrow{\$} \{0,1\}^\kappa$ which is used as the first authentication factor. The second authentication factor of $\mathsf{id_C}$ is randomly chosen as $sk^2_{\mathsf{id_C},\mathsf{id_S}} = K \xleftarrow{\$} \mathbb{Z}_p^*$, whereas the second and third authentication factors of $\mathsf{id_S}$ are initialized (temporarily) with empty sets $(sk^2_{\mathsf{id_S},\mathsf{id_C}}, sk^3_{\mathsf{id_S},\mathsf{id_C}}) = (\mathcal{D}_1, \mathcal{D}_2) = (\emptyset, \emptyset)$. However, we assume that before the protocol is running in practice, the client should generate enough authentication tokens for the server with random data via the following tag generation procedure.

treatment system [23]. The min-entropy of each individual sensor data is in the range between 5.518 and 8.848.

- **Tag Generation**. When the client $\mathsf{id_C}$ sends a data 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, and the tuple (i, d_i, t_i) is sent to $\mathsf{id_S}$ over a secure channel. Then $\mathsf{id_S}$ 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 $sk^3_{\mathsf{id_S},\mathsf{id_C}}(i) = (i, t_i)$. Meanwhile, the secure channel might be established by out-of-band mechanism (at the initialization phase) or the session key established during the following online authentication and key exchange phase.
- **Authentication and Key Exchange Phase**. The client $\mathsf{id_C}$ and the server $\mathsf{id_S}$ would interactively run the authenticated key exchange protocol online to generate a session key K_s as shown in Fig. 3. The established session key will be used to protect the underlying data and tag transmission. During this phase, both parties would first respectively 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 distinct 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, $\mathsf{id_S}$ makes use of its historical data (indexed by 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 sid is the session identifier concatenating the protocol messages and identities of participants. The messages (X, M) are sent to $\mathsf{id_C}$ for authentication. To verify M, $\mathsf{id_C}$ 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 $\mathsf{id_C}$ rejects the session. Otherwise, it generates the session key as $\mathsf{id_S}$. We assume that two parties synchronize a variable ξ which stores the times of the selection set SI that has been used. If $\xi = \phi$ then all indices in SI plus L.

Construction Discussions. To improve upon the CWZT protocol, we modify and add several critical steps to fix the vulnerabilities of the CWZT protocol and achieve the HMAKE functionality. We highlight our main differences with the state-of-the-art CWZT protocol [9] below.

- *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 Π_{woFS} contributes a random set I_S, such that the subset of selected historical data is determined by both parties (see Fig. 3), instead of only relying on the client.
- *New session key exchange feature.* Unlike the CWZT protocol, our protocol realizes the full-fledged authenticated key exchange (achieving both authentication and session key security goals). Our protocol enables both parties to establish a session key for securely transmitting the new authentication factors (i.e. data and its 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 then only corrupts either the tags or the data cannot actively impersonate as 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 $3\times$ hash operations comparing to the CWZT protocol, although we provide an additional key exchange functionality.

Limitations. Nevertheless, one of the limitations of Π_{woFS} is that it cannot provide forward secrecy, when all secrets used to compute a session key K_s are compromised from either player. 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., $z = 161$ for 128 bits security). Thus, the attacker needs to either keep stealing the second and third authentication factors or try to compromise the client's device which might be located in a more physically secure place in CPSs.

Another limitation of Π_{woFS} is that it can only satisfy static historical tag leakage. When the HTLeak query can be asked adaptively, the adversary will be able to ask HTLeak queries with indices appeared in the Test query to break the session key 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 Π_{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 modeled as a random oracle that can be asked at most q_h times, and each data piece is unpredictable. Then Π_{woFS} is $(t', \epsilon_{\Pi_{\mathsf{woFS}}}, q_l, \mathsf{woFS})$-secure with $t' \leq t$, $\phi \leq q_l$, and $\epsilon_{\Pi_{\mathsf{woFS}}} \leq \frac{\rho^2}{2^{\ell_r - 1}} + 14\rho \cdot (\frac{q_l}{L-z})^z + \frac{(14\rho+22+6L)\cdot q_h}{2^{\ell_p}}$.*

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) *Session-Key-adversary* is able to answer the session-key-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 ϵ_{auth} of authentication-adversaries, and Lemma 2 bounds the success probability ϵ_{skey} of session-key-adversaries. Then we have $\epsilon_{\Pi_{\mathsf{woFS}}} \leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{skey}}$.

The full proof of Theorem 1 is given in Appendix A. 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 $\Pi^u_{\mathsf{idC}^*}$ that accepts maliciously is at most $\epsilon_{\mathsf{auth}} \leq \frac{\rho^2}{2^{\ell_r}} + 6\rho \cdot (\frac{q_l}{L})^z + \frac{(6\rho+9+3L)\cdot q_h}{2^{\ell_p}}$.*

The proof of this lemma has three 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 S be the set of indices that are submitted to the 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 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 break the security of the protocol by its random guesses.

Lemma 2. *For any adversary $\mathcal{A}$ running in time $t' \approx t$, the probability that there exists an adversary $\mathcal{A}$ which answers session-key-challenge correctly is at most* $\epsilon_{\mathsf{skey}} \leq \frac{\rho^2}{2^{\ell_r}} + 8\rho \cdot (\frac{q_l}{L-z})^z + \frac{(8\rho+13+3L)\cdot q_h}{2^{\ell_p}}$.

The proof of this lemma mainly relies on the authentication security and the 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 I_C which is not authenticated. Hence, the probability that the selection set 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.

7 A HMAKE Protocol with Stronger Security

In this section, we propose an HMAKE protocol called Π_{FS}, which overcomes the limitations of Π_{woFS}. The idea of the construction of this protocol is to make use of the authentication procedure as a compiler to transform a general passively secure two-message key exchange protocol to achieve HMAKE security. To realize our idea, we need to add one more authentication message to achieve mutual explicit authentication for both parties. Comparing with Π_{woFS}, the protocol Π_{FS} can achieve not only PFS but also the resilience of adaptive historical tag leakage. Also, Π_{FS} can still guarantee authentication and key exchange security when *all* tags are corrupted but the historical data is not corrupted. It is because that the session key in Π_{FS} does not depend on the tags anymore.

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

Theorem 2. *Suppose that the hash function h is modeled as a random oracle that can be asked at most q_h times, each data piece is unpredictable, and the two-message key exchange protocol* TKE *is $(t, \epsilon_{\mathsf{TKE}})$-passively-secure. Then Π_{FS} is $(t', \epsilon_{\Pi_{\mathsf{FS}}}, q_l, \mathsf{FS})$-secure with $t' \leq t$, $\phi \leq q_l$, and* $\epsilon_{\Pi_{\mathsf{FS}}} \leq \frac{\rho^2}{2^{\ell_r - 1}} + 18\rho \cdot (\frac{q_l}{L-z})^z + \frac{(18+6L+18\rho)\cdot q_h}{2^{\ell_p}} + 2\rho \cdot (2\rho + 2) \cdot \epsilon_{\mathsf{TKE}}$.

Similarly, we prove Theorem 2 via the following two lemmas.

To prove Theorem 2, we present two lemmas. Lemma 3 bounds the success probability ϵ_{auth} of authentication-adversaries, and Lemma 4 bounds the success probability ϵ_{skey} of session-key-adversaries. Then we have $\epsilon_{\Pi_{\mathsf{FS}}} \leq \epsilon_{\mathsf{auth}} + \epsilon_{\mathsf{skey}}$.

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

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

Fig. 4: An HMAKE Protocol $\mathit{\Pi}_{\mathsf{FS}}$ with Perfect Forward Secrecy.

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

Lemma 4. *For any adversary $\mathcal{A}$ running in time $t' \approx t$, the probability that there exists an adversary $\mathcal{A}$ which answers session-key-challenge correctly is at most* $\frac{\rho^2}{2^{\ell_r}} + 9\rho\cdot(\frac{q_l}{L-z})^z + \frac{(9+3L+9\rho)\cdot q_h}{2^{\ell_p}} + 2\rho\cdot(2\rho+1)\cdot\epsilon_{\mathsf{TKE}}$.

Basically, the proof of this theorem can be extended from the proof of Theorem 1. We outline our proof idea as follows. In contrast to $\mathit{\Pi}_{\mathsf{woFS}}$, $\mathit{\Pi}_{\mathsf{FS}}$ can provide mutual explicit authentication. The authentication message M_1 sent from the client is computed in a similar way as M in $\mathit{\Pi}_{\mathsf{woFS}}$ and M_2 in $\mathit{\Pi}_{\mathsf{woFS}}$, therefore we can reduce the authentication security regarding M_1 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 $\mathit{\Pi}_{\mathsf{FS}}$ is twice of breaking the authentication of $\mathit{\Pi}_{\mathsf{woFS}}$. Also, the random values r_1 and r_2 in $\mathit{\Pi}_{\mathsf{woFS}}$ are replaced with m_1 and m_2 in $\mathit{\Pi}_{\mathsf{FS}}$, because of the security of the TKE protocol [30, Lemma1].

Moreover, if there is no adversary that can break the authentication property of $\mathit{\Pi}_{\mathsf{FS}}$, then there would be only 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 key exchange security of $\mathit{\Pi}_{\mathsf{FS}}$ to the security of TKE. We present the specific security reduction in Appendix B.

8 Security Enhancement for Legacy Devices

In this section, we show an important practical aspect of our HMAKE protocols, i.e. they are able to strengthen the security of existing legacy devices without modifying them.

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)[4]. A common (toy) AKE solution deployed on a legacy device might be like 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 (in the toy AKE scheme), and aux may contain other protocol messages if any (e.g., Diffie-Hellman public keys). Our HMAKE protocols can be simply adapted to enhance the security of such a legacy device with the above toy AKE 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 HMAKE protocol steps except 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 AKE scheme. The server can compute the same session key in the exactly 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[5].

To apply the above security enhancement in practice, we only need to check whether the legacy device runs an AKE protocol (or its variant – Authenticated Confidential Channel Establishment [31]) in the above form of toy example. One famous protocol instance meeting our requirement is the Transport Layer Security (TLS) Protocol with pre-shared key cipher-suits [15, 34, 31, 17] which are proposed for power-constrained devices (such as EMV card [33]). For example, our first protocol $\mathit{\Pi}_{\mathsf{woFS}}$ can be used to enhance the security of TLS_PSK, and the second protocol $\mathit{\Pi}_{\mathsf{FS}}$ 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 Implementation and 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 $\mathit{\Pi}_{\mathsf{woFS}}$ and $\mathit{\Pi}_{\mathsf{FS}}$ based on the corresponding security levels: (i) for the security level $\kappa = 80$, we use $\ell_r = 141$, $z = 113$, $\ell_p = 145$ for $\mathit{\Pi}_{\mathsf{woFS}}$, and $\ell_p = 224$ for $\mathit{\Pi}_{\mathsf{FS}}$; (ii) for the security level $\kappa = 128$, we use $\ell_r = 189$, $z = 161$, $\ell_p = 193$ for $\mathit{\Pi}_{\mathsf{woFS}}$, and $\ell_p = 320$ for $\mathit{\Pi}_{\mathsf{FS}}$.

[4] In case the legacy devices do not have an AKE built in, it becomes trivial for us to enhance their security. We can simply add a new device like what the authors did in [5] to intercept the traffic of legacy devices and run the complete HMAKE protocols with the server. This is still legacy-compliant. However, the practical difficulty is how to be compatible with legacy devices which run common AKE protocols.

[5] 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.

Experiments Setup. We used one PC (with Intel Core i7-8750H processor) as a server, and a Raspberry Pi 3 (with Quad Core 1.2GHz Broadcom BCM2837 CPU and 1GB RAM) is taken as a client. Our implementation is based on MIRACL cryptographic library [32], where the hash function used is SHA256 in Π_{woFS} and SHA384 in Π_{FS} , and the TKE protocol used in our second protocol is the Diffie-Hellman key exchange protocol based on the standard elliptic curve (over $GF(p)$) provided by MIRACL. **Performance Evaluation.** We first measured the tag generation time on the client. It takes 0.55 ms per tag, assuming data size is 1KB. Also, we measured the time consumed by the authentication protocol and the key generation procedure separately on both the server and the client. The performance is reported in milliseconds in Table 1; 'KE' denotes the time for ephemeral key and the session key generations, and 'Auth' denotes the performance of all other steps in authentication. 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 performance of the client in Π_{woFS} is only 24.695 ms, which is efficient enough to be deployed in real-world applications.

	Π_{woFS}		Π_{FS}	
	Server	Client	Server	Client
Auth	0.137/0.213	17.184/24.336	1.986/4.056	65.561/82.795
KE	0.030/0.045	0.299/0.359	1.827/3.879	54.530/69.842

Table 1: The performance of the proposed HMAKE protocols for (80-bit security/128-bit security), measured in ms.

10 Conclusions and Open Problems

In this paper, we have shown two ways to build multi-factor AKE 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 a HMAKE protocol in the standard model. Its challenge is to generate a pseudo-random seed from the authentication tags.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No. 61872051, the Research Project of the Humanities and Social Sciences of the Ministry of Education of China under Grant No. 16YJC870018, and the SUTD start-up research under Grant No.: SRG-ISTD-2017-124. Chenglu Jin was partially supported by AFOSR MURI under award number FA9550-14-1-0351.

References

1. Y. Aumann, Y. Z. Ding, and M. O. Rabin. Everlasting security in the bounded storage model. *IEEE Trans. Info. Theory*, 48(6):1668–1680, 2002.
2. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *CRYPTO*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.
3. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73. ACM, 1993.
4. D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *CRYPTO*, volume 10991 of *LNCS*, pages 757–788. Springer, 2018.
5. J. H. Castellanos, D. Antonioli, N. O. Tippenhauer, and M. Ochoa. Legacy-compliant data authentication for industrial control system traffic. In *ACNS*, pages 665–685. Springer, 2017.
6. 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*, 5:3028–3043, 2017.
7. 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*, 5:3028–3043, 2017.
8. A. C. Chan. Efficient defence against misbehaving TCP receiver dos attacks. *Computer Networks*, 55(17):3904–3914, 2011.
9. A. C. Chan, J. W. Wong, J. Zhou, and J. C. M. Teo. Scalable two-factor authentication using historical data. In *ESORICS*, volume 9878 of *LNCS*, pages 91–110. Springer, 2016.
10. D. Chattaraj, M. Sarma, and A. K. Das. A new two-server authentication and key agreement protocol for accessing secure cloud services. *Computer Networks*, 131:144–164, 2018.
11. L. Chen and C. J. Mitchell, editors. *Security Standardisation Research - First International Conference, SSR 2014, London, UK, December 16-17, 2014. Proceedings*, volume 8893 of *Lecture Notes in Computer Science*. Springer, 2014.
12. 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*, 2018:7574238, 2018.
13. 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.
14. 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*, 9(16):3670–3687, 2016.
15. T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. Technical report, Internet Engineering Task Force (IETF), 2008.
16. Y. Dodis, S. Guo, and J. Katz. Fixing cracks in the concrete: Random oracles with auxiliary input, revisited. In *EUROCRYPT*, volume 10211 of *LNCS*, pages 473–495, 2017.
17. B. Dowling, M. Fischlin, F. Günther, and D. Stebila. A cryptographic analysis of the TLS 1.3 handshake protocol candidates. In *ACM CCS*, pages 1197–1210. ACM, 2015.
18. A. Dua, N. Kumar, A. K. Das, and W. Susilo. Secure message communication protocol among vehicles in smart city. *IEEE Trans. Vehicular Technology*, 67(5):4359–4373, 2018.
19. S. Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, volume 3876 of *LNCS*, pages 207–224. Springer, 2006.
20. 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 Internet of Things Journal*, 6(1):288–296, 2019.
21. N. Falliere, L. O. Murchu, and E. Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.

22. N. Fleischhacker, M. Manulis, and A. Azodi. A modular framework for multi-factor authentication and key exchange. In *Security Standardisation Research - First International Conference, SSR 2014, London, UK, December 16-17, 2014. Proceedings*, pages 190–214, 2014.
23. 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*, pages 88–99. Springer, 2016.
24. 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*, 14(1):1–11, 2018.
25. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. Generic compilers for authenticated key exchange. In *ASIACRYPT*, volume 6477 of *LNCS*, pages 232–249. Springer, 2010.
26. T. Jager, F. Kohlar, S. Schäge, and J. Schwenk. On the security of TLS-DHE in the standard model. In *CRYPTO*, volume 7417 of *LNCS*, pages 273–293. Springer, 2012.
27. S. Jarecki, H. Krawczyk, M. Shirvanian, and N. Saxena. Two-factor authentication with end-to-end password security. In *PKC*, volume 10770 of *LNCS*, pages 431–461. Springer, 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.*, 77(14):18355–18382, 2018.
29. C. Jin, S. Valizadeh, and M. van Dijk. Snapshotter: Lightweight intrusion detection and prevention system for industrial control systems. In *ICPS*, pages 824–829. IEEE, 2018.
30. Y. Li, S. Schäge, Z. Yang, C. Bader, and J. Schwenk. New modular compilers for authenticated key exchange. In *ACNS*, volume 8479 of *LNCS*, pages 1–18. Springer, 2014.
31. Y. Li, S. Schäge, Z. Yang, F. Kohlar, and J. Schwenk. On the security of the pre-shared key ciphersuites of TLS. In *PKC*, volume 8383 of *LNCS*, pages 669–684. Springer, 2014.
32. Miracl cryptographic library, 2018.
33. L. P. Urien and P. Martin. EMV support for TLS-PSK. draft-urien-tls-psk-emv-02, Feb. 2011.
34. E. Rescorla. The transport layer security (tls) protocol version 1.3. Technical report, Internet Engineering Task Force (IETF), 2018.
35. H. Shacham and B. Waters. Compact proofs of retrievability. *J. Cryptology*, 26(3):442–483, 2013.
36. SIEMENS. Simatic process historian.
37. 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 Internet of Things Journal*, 5(1):269–282, 2018.
38. Z. Yang and J. Lai. New constructions for (multiparty) one-round key exchange with strong security. *SCIENCE CHINA Information Sciences*, 61(5):059102:1–059102:3, 2018.
39. 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.*, 60(8):1145–1160, 2017.
40. Z. Yang, J. Lai, Y. Sun, and J. Zhou. A novel authenticated key agreement protocol with dynamic credential for wsns. *ACM Trans. Sen. Netw.*, 15(2), March 2019.
41. 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*, 18(6):1109–1121, 2016.
42. R. Zhang, Y. Xiao, S. Sun, and H. Ma. Efficient multi-factor authenticated key exchange scheme for mobile communications. *IEEE Trans. on Dependable and Secure Computing*, 2017.

A Proof of Theorem 1

The proof of Theorem 1 has two parts: (i) the proof of Lemma 1 for authentication security, and (ii) the proof of Lemma 2 for key exchange security.

A.1 Proof of Lemma 1

In the following, we show the proof of Lemma 1 in a sequence of games. Let $\mathsf{S}_i^{\text{auth}}$ denote an event that there exists an authentication-adversary wins in Game i.

Game 0. This game equals the real security experiment described in Section 5. Meanwhile, all oracle queries are answered honestly according to our protocol specification. Thus, we have that

$$\Pr[\mathsf{S}_0^{\text{auth}}] = \epsilon_{\mathsf{auth}}.$$

Game 1. In this game, the challenger $\mathcal{C}$ proceeds exactly like the previous game, but adds an abort rule to all Send queries that it aborts if: two oracles generate the same nonce (i.e., either r_1 or r_2). Note that there are ρ oracles at each honest party. By applying the birthday paradox, we have that

$$\Pr[\mathsf{S}_0^{\text{auth}}] \leq \Pr[\mathsf{S}_1^{\text{auth}}] + \frac{\rho^2}{2^{\ell_r}}.$$

Due to the modification in this game, each session identifier sid including $r_1||r_2$ is uniquely shared with its partner oracle. The unique sid ensures the uniqueness of each authentication message M generated involving sid is unique as well (even though the selection set I has collision).

Game 2. Note that the adversary can choose to corrupt either the first authentication factor or the second authentication factor, but not both of them. Hence we need to proceed with the proof following one of the following corruption cases:

- **Corruption Case 1**: $\mathcal{A}$ did not ask any $\mathsf{Corrupt1}(\cdot)$ query;
- **Corruption Case 2**: $\mathcal{A}$ did not ask any $\mathsf{Corrupt2}(\cdot)$ query;
- **Corruption Case 3**: $\mathcal{A}$ did not ask any $\mathsf{Corrupt3}(\cdot)$ query, and $\mathcal{A}$ asked no more than q_l queries to $\mathsf{HTLeak}(3, \cdot)$.

In this game, $\mathcal{C}$ guesses which the above corruption case would occur. If $\mathcal{C}$ guesses incorrectly, then it halts the game. The probability that $\mathcal{C}$ succeeds in guessing the corruption case is bounded by 1/3. Thus, we have that

$$\Pr[\mathsf{S}_1^{\text{auth}}] = 3 \cdot \Pr[\mathsf{S}_2^{\text{auth}}].$$

Game 3. Let S be the set of indices that are submitted to the HTLeak query. In this game, when the corruption case 2 or case 3 occurs, then we add an abort rule: $\mathcal{C}$ aborts if there is an oracle $\pi_{\mathsf{id_C}^*}^u$ which accepts maliciously and sends out a selection set I_C^* such that $\mathsf{I}_C^* \subseteq \mathsf{S}$ (which means all secrets associated with indices in I_C^* are leaked). Note that the size of I_C^* is z and the size of S is $q_l >> z$. We bound the probability

$$\Pr[\mathsf{I}_C^* \subseteq \mathsf{S}] = \frac{\binom{q_l}{z}}{\binom{L}{z}} < (\frac{q_l}{L})^z.$$

Let $\mathsf{abort}_{MS}^{auth}$ denote the event that $\mathcal{C}$ aborts in this game. Since there are at most ρ oracles at $\mathsf{id_C}^*$ we have that

$$\begin{aligned}\Pr[\mathsf{abort}_{MS}^{auth}] &= 1 - (\Pr[\overline{\mathsf{I}_C^* \subseteq \mathsf{S}}])^{\rho} \\ &= 1 - (1 - (\frac{q_l}{L})^z)^{\rho} < \rho \cdot (\frac{q_l}{L})^z,\end{aligned}$$

with sufficient large z. Therefore, we have that

$$\Pr[\mathsf{S}_2^{\mathrm{auth}}] = \Pr[\mathsf{S}_3^{\mathrm{auth}}] + \Pr[\mathsf{abort}_{MS}^{auth}] < \Pr[\mathsf{S}_3^{\mathrm{auth}}] + \rho \cdot (\frac{q_l}{L})^z.$$

Note that here we only consider the selection set I_C^* not I_S since I_S might be chosen by the adversary in an impersonation attack. If $\mathcal{C}$ does not abort in this game, then it implies that each session must choose a I_C^* containing at least one uncompromised index when most of the secret tags are uncompromised.

Game 4. In this game, $\mathcal{C}$ aborts if one of the uncompromised secrets (which could be any factors) is asked by the adversary $\mathcal{A}$ in a random oracle query. This implies $\mathcal{A}$ knows the uncompromised secret. Note that the input of each hash operation is unique (by assumptions) that would result in a unique random hash value. To learn an uncompromised secret, an adversary may test many random oracle queries with its own inputs. Specifically, $\mathcal{C}$ aborts if and only if one of the following condition holds:

- When the corruption case 1 occurs, $sk^1_{\mathsf{id_C}^*,\mathsf{id_S}^*}$ is asked by $\mathcal{A}$ in a random oracle query;
- When the corruption case 2 occurs, either K of $\mathsf{id_C}^*$ or one of the uncompromised data pieces is asked by $\mathcal{A}$ in a random oracle query.
- When the corruption case 3 occurs, either K of $\mathsf{id_C}^*$ or one of the uncompromised tags is asked by $\mathcal{A}$ in a random oracle query.

Since each data piece is not known by the adversary (under the corruption case 2) and unpredictable, and all the confidential secretes $\{K, d_i, sk^1_{\mathsf{id_C}^*,\mathsf{id_S}^*}\}$ are chosen uniformly at random with bit-length at least ℓ_p, and $\mathcal{A}$ can only guess them with q_h trials in conjunction with her random oracle queries. Thus we have that

$$\Pr[\mathsf{S}_3^{\mathrm{auth}}] = \Pr[\mathsf{S}_4^{\mathrm{auth}}] + \frac{(3+L)q_h}{2^{\ell_p}}.$$

Game 5. $\mathcal{C}$ proceeds this game exactly as before, but aborts if an oracle $\pi^u_{\mathsf{id_C}^*}$ such that $\mathsf{FullC}(\mathsf{id_C}^*, \mathsf{pid}^u_{\mathsf{id_C}^*}) = 0$ (throughout the game) received an $X^u_{\mathsf{id_C}^*}$ which is not sent from its partner oracle. As each selection set SI is only used for $\phi \leq q_l$ times, the maximum hashed data $h(d_i||i)$ leaked from X is bound to q_l. With the similar argument in the Game 3, we have that the $X^u_{\mathsf{id_C}^*}$ should be computed involving a secret value $h(d_i^*||i^*)$ that is not compromised (under the corruption case 2) with probability $\rho \cdot (\frac{q_l}{L})^z$. Since $X^u_{\mathsf{id_C}^*}$ is computed based on the distinct selection set SI and uniform random hash values (due to the random oracle queries with unique inputs), each $X^u_{\mathsf{id_C}^*}$ is unique as well. So that $X^u_{\mathsf{id_C}^*}$ cannot be forged or replayed with non-negligible probability. Namely, $\mathcal{A}$ can only randomly guess $X^u_{\mathsf{id_C}^*}$. Hence, we have that

$$\Pr[\mathsf{S}_4^{\mathrm{auth}}] \leq \Pr[\mathsf{S}_5^{\mathrm{auth}}] + \rho \cdot (\frac{q_l}{L})^z + \frac{\rho \cdot q_h}{2^{\ell_p}}.$$

Game 6. In this game, $\mathcal{C}$ aborts if $\mathcal{A}$ asked a random oracle query with the value $Y^u_{\mathsf{id_C}^*}$ of an oracle $\pi^u_{\mathsf{id_C}^*}$ such that $\mathsf{FullC}(\mathsf{id_C}^*, \mathsf{pid}^u_{\mathsf{id_C}^*}) = 0$ throughout the game. Recall that $Y^u_{\mathsf{id_C}^*}$ should be computed involving a secret tag t_{i^*} with index i^* that has not been submitted to the HTLeak query. Furthermore, $Y^u_{\mathsf{id_C}^*}$ is hidden by the hash function. Hence, $\mathcal{A}$ who does not know t_{i^*} cannot compute $Y^u_{\mathsf{id_C}^*}$ (respectively) due to the modification in the previous game. So that $\mathcal{A}$ can only randomly guess $Y^u_{\mathsf{id_C}^*}$. Analogously, we have that

$$\Pr[\mathsf{S}^{\mathrm{auth}}_5] = \Pr[\mathsf{S}^{\mathrm{auth}}_6] + \frac{\rho \cdot q_h}{2^{\ell_p}}.$$

Game 7. In this game, for each oracle $\pi^u_{\mathsf{id_C}^*}$ such that $\mathsf{FullC}(\mathsf{id_C}^*, \mathsf{pid}^u_{\mathsf{id_C}^*}, q_l) = 0$, $\pi^u_{\mathsf{id_C}^*}$ rejects if it receives a message which is not sent by its partner oracle having a matching conversation to $\pi^u_{\mathsf{id_C}^*}$. Since $\mathcal{A}$ cannot compute $Y^u_{\mathsf{id_C}^*}$ used by $\pi^u_{\mathsf{id_C}^*}$ for verification, it is unable to distinguish this game from the previous game. Thus the advantage of $\mathcal{A}$ in this game is zero.

Summing up the probabilities in all the above games, we have the result of Lemma 1.

A.2 Proof of Lemma 2

Let $\mathsf{S}^{\mathrm{ke}}_i$ denote an event that there exists a session-key-adversary answers the session-key-challenge correctly in Game i. The proof of this lemma is quite similar to the proof of Lemma 1. We may omit some similar details to avoid repetition. We show the proof of this lemma by the following games.

Game 0. This game equals the real security experiment described in Section 5. We have that

$$\Pr[\mathsf{S}^{\mathrm{ke}}_0] - 1/2 = \epsilon_{\mathsf{skey}}.$$

Game 1. In this game, $\mathcal{C}$ aborts if the owner of the test oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathcal{ID}^* = \mathsf{id_C}^*$, $\mathsf{FullC}(\mathcal{ID}^*, \mathsf{pid}^u_{\mathcal{ID}^*}, q_l) = 0$, and $\pi^u_{\mathcal{ID}^*}$ accepts without a partner oracle at $\mathsf{pid}^u_{\mathcal{ID}^*}$. Due to the authentication property of the protocol, we have that

$$\Pr[\mathsf{S}^{\mathrm{ke}}_0] = \Pr[\mathsf{S}^{\mathrm{ke}}_1] + \epsilon_{\mathsf{auth}}.$$

Hence, if the owner of the test oracle is the honest client then it must have a matching conversation at the server.

Game 2. In this game, $\mathcal{C}$ would guess in advance which corruption case will occur to the test oracle and its partner oracle. Note that the corruption case 3 will never occur by the security definition. $\mathcal{C}$ aborts if it guesses incorrectly. Thus we have

$$\Pr[\mathsf{S}^{\mathrm{ke}}_1] = 2 \cdot \Pr[\mathsf{S}^{\mathrm{ke}}_2].$$

Game 3. Recall that S is assumed to be the set of indices that are submitted to the HTLeak query. We add an abort rule: $\mathcal{C}$ aborts if the test oracle's owner is $\mathsf{id_S}^*$ and $\pi^u_{\mathsf{id_S}^*}$ sends out a selection set I^*_S such that $\mathsf{I}^*_S \subseteq \mathsf{S}$. Note that the I^*_S is chosen from $\mathsf{SI} \backslash \mathsf{I}_{C^*}$, where I_{C^*} is the selection set received by $\pi^u_{\mathsf{id_S}^*}$ (that may be chosen by $\mathcal{A}$). Similarly, we bound the probability

$$\Pr[\mathsf{I}^*_S \subseteq \mathsf{S}] = \frac{\binom{q_l}{z}}{\binom{L-z}{z}} < (\frac{q_l}{L-z})^z.$$

Let abort^{ke}_{MS} denote the event that $\mathcal{C}$ aborts in this game. Therefore, we have

$$\Pr[\mathsf{S}^{\mathrm{ke}}_2] = \Pr[\mathsf{S}^{\mathrm{ke}}_3] + \Pr[\mathsf{abort}^{ke}_{MS}] < \Pr[\mathsf{S}^{\mathrm{ke}}_3] + \rho \cdot (\frac{q_l}{L-z})^z.$$

Game 4. In this game, $\mathcal{C}$ aborts if one of the uncompromised secrets is asked by the adversary $\mathcal{A}$ in a random oracle query. Thus we have that

$$\Pr[\mathsf{S}^{\mathrm{ke}}_3] = \Pr[\mathsf{S}^{\mathrm{ke}}_4] + \frac{2q_h}{2^{\ell_p}}.$$

Game 5. In this game, $\mathcal{C}$ aborts if $\mathcal{A}$ asked a random oracle query with the value $Y^u_{\mathcal{ID}^*}$ of the test oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathsf{FullC}(\mathcal{ID}^*, \mathsf{pid}^u_{\mathcal{ID}^*}) = 0$ (throughout the game). Therefore, we have that

$$\Pr[\mathsf{S}^{\mathrm{ke}}_4] = \Pr[\mathsf{S}^{\mathrm{ke}}_5] + \frac{\rho \cdot q_h}{2^{\ell_p}}.$$

Game 6. We replace the session key of the test oracle and its partner oracle (if any) with a truly random key that is independent of the bit chosen by the test oracle. Thus the adversary gains no advantage in this game, i.e.,

$$\Pr[\mathsf{S}^{\mathrm{ke}}_5] = \Pr[\mathsf{S}^{\mathrm{ke}}_6] = 0.$$

Summing up the probabilities in the above games, we obtain the result of Lemma 2.

B Proof of Theorem 2

The proof of Theorem 2 consists of the proof of Lemma 3 and the proof of Lemma 4.

B.1 Proof of Lemma 3

Let $\mathsf{S}^{\mathrm{auth}}_i$ denote an event that there exists an authentication-adversary wins in Game i.
Game 0. This game equals the real security experiment described in Section 5. We have that

$$\Pr[\mathsf{S}^{\mathrm{auth}}_0] = \epsilon_{\mathsf{auth}}.$$

Game 1. In this game, the challenger $\mathcal{C}$ proceeds exactly like the previous game, but aborts if two oracles generate the same randomness (i.e., either $\tilde{r}_1$ or $\tilde{r}_2$). Due to the birthday paradox, we have that

$$\Pr[\mathsf{S}^{\mathrm{auth}}_0] \leq \Pr[\mathsf{S}^{\mathrm{auth}}_1] + \frac{\rho^2}{2^{\ell_r}}.$$

So that each invocation of $\mathsf{TKE.MSG}$ takes as input a unique randomness.
Game 2. In this game, $\mathcal{C}$ proceeds as the previous game, but aborts if two oracles generate the same ephemeral public key of TKE (i.e., either m_1 or m_2). Let ϵ_{coll} note the event that two oracles have the identical ephemeral public keys. From [30, Lemma1], we have that if TKE is $(t, \epsilon_{\mathsf{TKE}})$-passively-secure without long-term key, then all ephemeral public keys generated by $\mathsf{TKE.MSG}$ in the runs of TKE are $(\rho, t, \epsilon_{coll})$-distinct such that $\epsilon_{coll} \leq \rho \cdot \epsilon_{\mathsf{TKE}}$. Since there are two honest parties and each party has ρ oracles, we have that

$$\Pr[\mathsf{S}^{\mathrm{auth}}_1] \leq \Pr[\mathsf{S}^{\mathrm{auth}}_2] + 2\rho \cdot \epsilon_{\mathsf{TKE}}.$$

As a result, each session identifier sid including $m_1||m_2$ is uniquely shared with its partner oracle.

Game 3. In this game, $\mathcal{C}$ guesses which the corruption case would occur (as in the proof of Lemma 1). $\mathcal{C}$ aborts if it fails in such a guess. The probability that $\mathcal{C}$ succeeds in guessing the corruption case is bounded by 1/3. Thus, we have that

$$\Pr[\mathsf{S}_2^{\text{auth}}] = 3 \cdot \Pr[\mathsf{S}_3^{\text{auth}}].$$

Game 4. Let S be the set of indices that are submitted to the HTLeak query. In this game, when the corruption case 2 or case 3 occurs, then $\mathcal{C}$ aborts if there is an oracle $\pi^u_{\mathcal{ID}^*}$ which accepts maliciously and sends out a selection set I^*_P such that $\mathsf{I}^*_P \subseteq \mathsf{S}$ (which means all secrets associated with indices in I^*_P are leaked), where $P^* \in \{C, S\}$. Note that I^*_P is chosen from an index set with size at least $L - z$. As in Game 3 in the proof of Lemma 1, We bound the probability

$$\Pr[\mathsf{I}^*_P \subseteq \mathsf{S}] = \frac{\binom{q_l}{z}}{\binom{L-z}{z}} < (\frac{q_l}{L-z})^z.$$

Since there are at most 2ρ such honest selection sets (for either prover or verifier) would be chosen, we have that

$$\Pr[\mathsf{S}_3^{\text{auth}}] < \Pr[\mathsf{S}_4^{\text{auth}}] + 2\rho \cdot (\frac{q_l}{L-z})^z.$$

Game 5. In this game, $\mathcal{C}$ aborts if one of the uncompromised secrets (which could be any factors) is asked by the adversary $\mathcal{A}$ in a random oracle query.

With the similar argument in Game 4 in the proof of Lemma 1, we have that

$$\Pr[\mathsf{S}_4^{\text{auth}}] = \Pr[\mathsf{S}_5^{\text{auth}}] + \frac{(3+L)q_h}{2^{\ell_p}}.$$

Game 6. $\mathcal{C}$ proceeds this game exactly as before, but aborts if an oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathsf{FullC}(\mathsf{id_C}^*, \mathsf{pid}^u_{\mathsf{id_C}^*}) = 0$ (throughout the game) received an $X^u_{\mathsf{id_C}^*}$ which is not sent from its partner oracle. We would like to bound the probability that an adversary's probability on forging $X^u_{\mathsf{id_C}^*}$. As stated in Game 5 in the proof of Lemma 1, we have that

$$\Pr[\mathsf{S}_5^{\text{auth}}] < \Pr[\mathsf{S}_6^{\text{auth}}] + \rho \cdot (\frac{q_l}{L})^z + \frac{\rho \cdot q_h}{2^{\ell_p}}.$$

Game 7. In this game, $\mathcal{C}$ aborts if $\mathcal{A}$ asked a random oracle query with the value $Y^u_{\mathcal{ID}^*}$ of an oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathsf{FullC}(\mathcal{ID}^*, \mathsf{pid}^u_{\mathcal{ID}^*}) = 0$ throughout the game. As $Y^u_{\mathcal{ID}^*}$ is computed involving a secret tag t_{i^*} which is not exposed. Furthermore, $Y^u_{\mathcal{ID}^*}$ is hidden by the hash function. Hence, $\mathcal{A}$ can only submit guessed $Y^u_{\mathcal{ID}^*}$ to random oracle queries. Since there are at most 2ρ honest oracles that an adversary may try to attack, we have that

$$\Pr[\mathsf{S}_6^{\text{auth}}] = \Pr[\mathsf{S}_7^{\text{auth}}] + \frac{2\rho \cdot q_h}{2^{\ell_p}}.$$

Game 8. In this game, for each oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathsf{FullC}(\mathcal{ID}^*, \mathsf{pid}^u_{\mathcal{ID}^*}, q_l) = 0$, $\pi^u_{\mathcal{ID}^*}$ rejects if it receives a message which is not sent by its partner oracle having a matching conversation to $\pi^u_{\mathcal{ID}^*}$. Since $\mathcal{A}$ cannot compute $Y^u_{\mathcal{ID}^*}$ used by $\pi^u_{\mathcal{ID}^*}$ for verification, it is unable to distinguish this game from the previous game. Thus the advantage of $\mathcal{A}$ in this game is zero.

Summing up the probabilities in all the above games, we have the result of Lemma 3.

B.2 Proof of Lemma 4

Let $\mathsf{S}_i^{\mathsf{ke}}$ denote an event that there exists a session-key-adversary answers the session-key-challenge correctly in Game i. We show the proof of this lemma by the following games.

Game 0. This game equals the real security experiment described in Section 5. We have that

$$\Pr[\mathsf{S}_0^{\mathsf{ke}}] - 1/2 = \epsilon_{\mathsf{skey}}.$$

Game 1. In this game, $\mathcal{C}$ aborts if the owner of the test oracle $\pi^u_{\mathcal{ID}^*}$ such that $\mathsf{FullC}(\mathcal{ID}^*, \mathsf{pid}^u_{\mathcal{ID}^*}, q_l) = 0$, and $\pi^u_{\mathcal{ID}^*}$ accepts without a partner oracle at $\mathsf{pid}^u_{\mathcal{ID}^*}$. Due to the authentication property of the protocol, we have that

$$\Pr[\mathsf{S}_0^{\mathsf{ke}}] = \Pr[\mathsf{S}_1^{\mathsf{ke}}] + \epsilon_{\mathsf{auth}}.$$

Hence, the test oracle must have a matching conversation at the server.

Game 2. This game proceeds exactly as the previous game but $\mathcal{C}$ aborts if it fails to guess the test oracle $\pi^u_{\mathcal{ID}^*}$ and its partner oracle $\pi^s_{\mathsf{pid}^u_{\mathcal{ID}^*}}$ such that they have matching conversations. Since there are 2 honest parties and ρ oracles for each party, the probability that $\mathcal{C}$ guesses correctly is at least $1/(2\rho)^2$. Thus we have that

$$\Pr[\mathsf{S}_1^{\mathsf{ke}}] \leq 4\rho^2 \cdot \Pr[\mathsf{S}_2^{\mathsf{ke}}].$$

Game 3. In this game, $\mathcal{C}$ replaces the key $\underset{\sim}{k^{ke,*}}$ of the test oracle $\pi^u_{\mathcal{ID}^*}$ and its partner oracle $\pi^s_{\mathsf{pid}^u_{\mathcal{ID}^*}}$ with the same random value $\widetilde{K}^{ke,*}$. Note that the TKE protocol instance executed between the test oracle and its partner oracle only allows for passive adversaries due to the change in the previous game. If there exists an adversary $\mathcal{A}$ which can distinguish this game from the previous game, then we use it to construct an algorithm $\mathcal{B}$ to break the passive security of TKE as follows. We assume that $\mathcal{B}$ interacts with the TKE challenger $\mathcal{C}_{\mathsf{TKE}}$ through $\mathsf{Execute}$ query. Meanwhile, $\mathcal{B}$ simulates the AKE challenger in this game for $\mathcal{A}$ as follows:

- Initially, $\mathcal{B}$ implements all honest oracles.
- Meanwhile, $\mathcal{B}$ generates the ephemeral key (i.e., m_1 or m_2) for each oracle $\pi^s_{\mathcal{ID}_i}$ using the ephemeral randomness of her own choice and answers all oracle queries honestly except for the test oracle and its partner oracle.
- As for the correctly guessed test oracle $\pi^s_{\mathsf{pid}^u_{\mathcal{ID}^*}}$ and its partner oracle $\pi^{t^*}_j$, $\mathcal{B}$ queries $\mathcal{C}_{\mathsf{TKE}}$ for executing a TKE test protocol instance and obtains (T^*, K_b^*) from $\mathcal{C}_{\mathsf{TKE}}$. Otherwise $\mathcal{B}$ simulates the ephemeral keys of $\pi^u_{\mathcal{ID}^*}$ and $\pi^s_{\mathsf{pid}^u_{\mathcal{ID}^*}}$ using the transcript T^*, and uses K_b^* to compute the session key of the test oracle.
- Eventually, $\mathcal{B}$ returns the bit b' given by $\mathcal{A}$ to $\mathcal{C}_{\mathsf{TKE}}$.

The simulation of $\mathcal{B}$ is perfect since $\mathcal{B}$ can always correctly answer all queries from $\mathcal{A}$. If $\mathcal{A}$ can correctly answer the bit b of the Test query with non-negligible probability, so can $\mathcal{B}$. By applying the security of TKE, we obtain that

$$\Pr[\mathsf{S}_2^{\mathsf{ke}}] \leq \Pr[\mathsf{S}_3^{\mathsf{ke}}] + \epsilon_{\mathsf{TKE}}.$$

Game 4. We replace the session key of the test oracle and its partner oracle (if any) with a truly random key that is independent of the bit chosen by the test oracle. This is possible since the test oracle would submit a random key material $\widetilde{K^{ke,*}}$ to the random oracle which results in a random session key as well. Thus the adversary gains no advantage in this game, i.e.,

$$\Pr[\mathsf{S}_3^{\mathrm{ke}}] = \Pr[\mathsf{S}_4^{\mathrm{ke}}] = 0.$$

Summing up the probabilities in the above games, we obtain the result of Lemma 4.

C Comparison

In this section, we briefly compare our proposed schemes with two recent typical lightweight authenticated key exchange protocols, i.e., Das et al., [14], He et al. [24] and Challa et al. [7], just for reference. Although these protocols are designed for the three-party case, two-party AKE procedure is also involved. We compare these four protocols from the following perspectives: (i) authentication factors, (ii) main security properties, (iii) number of communication passes, and (iv) computation cost. To compare the computational cost, we instantiate our protocol Π_{FS}^{RO} with the elliptic curve cryptography (ECC) based Diffie-Hellman key exchange protocol (as the other compared protocols). Furthermore, we let 'FE' denote a fuzzy extractor operation to obtain a secret from biometrics. We let 'S-Auth' denote single-side explicit authentication, 'M-Auth' denote mutual authentication, 'B-Leak' denote bounded leakage, 'IA-SKS' denote implicit authentication with session key security. To compare the computation cost, let 'H' denote hash function operation and 'MUL' denote an ECC multiplication. Let 'Bio' denote the biometric authentication factor, 'PW' denote password, 'HD' denote historic data, 'HT' denote historic tags, 'LSK' denote long-term symmetric key, and 'LPK' denote the long-term public key.

We summarize the comparison in Table 2.

Protocol	Authentication Factors	Security Properties					Comm. Pass	Comp. Cost
		S-Auth	M-Auth	IA-SKS	B-Leak	PFS		
Das et al.[14]	Bio+PW +LSK	√	×	√	×	√	2	31H+1FE+4MUL
He et al.[24]	PW+LSK	√	×	√	×	√	2	21H+4MUL
Challa et al. [7]	Bio+PW+LPK	√	×	√	×	×	2	1Fe+14Mul+12h
Π_{woFS}^{RO}	LSK+HD	√	×	√	√ (static)	×	2	326 H
Π_{FS}^{RO}	LSK+HT+HD	√	√	√	√ (adaptive)	√	3	328H+4MUL

Table 2: Comparison

Though our protocols are less efficient than Das et al. and He 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 Π_{woFS} is still practical (as shown in Table 1) for constrained devices.