短期秘密鍵漏洩耐性を持つIoT向け耐量子認証鍵交換

石橋　錬[1,a)]　米山　一樹[1,b)]

概要：認証鍵交換（AKE）は複数のパーティ間で共通のセッション鍵を共有するための暗号プロトコルである。本研究では，双方のリソースが非対称であるような，IoT デバイスとサーバ間で実行される IoT-oriented AKE に注目する。Liu らは，perfect forward secrecy (PFS)，key compromise impersonation (KCI)，server compromise impersonation (SCI) に対して安全なビッグデータを用いた IoT 向け AKE (IoT-AKE) の安全性モデル (LTZ モデル) を定式化した。また，彼らは IoT-AKE のフレームワークを提案し，具体的な耐量子方式を与えた。しかし，LTZ モデルは短期秘密鍵漏洩が考慮されていないモデルであり，また彼らの方式はランダムオラクルモデルを仮定している。本稿では，LTZ モデルを短期秘密鍵漏洩を捉えたモデルへ拡張し，ランダムオラクルモデルと標準モデルにおける IoT-AKE の一般構成をそれぞれ提案する。我々の一般構成により，初めての標準モデルで安全な耐量子 IoT-AKE 方式が得られる。

キーワード：認証鍵交換, IoT-eCK, IoT, SCI, 耐量子

Post-Quantum Authenticated Key Exchange Resilient to Ephemeral Key Leakage in the Bounded-Retrieval Model for IoT

Ren Ishibashi[1,a)]　Kazuki Yoneyama[1,b)]

Abstract: Authenticated Key Exchange (AKE) is a cryptographic protocol to share a common session key among multiple parties. Recently, not only the ordinary PKI-based AKE but also the Internet of Things (IoT) setting has been studied. We focus on an IoT-oriented AKE (IoT-AKE) where the resources of both parties are asymmetric, such that the IoT-device runs the protocol to establish a session key with a remote server. Liu et al. formulated a security model of the IoT-AKE (LTZ model) using big-data in the bounded-retrieval model, which guarantees perfect forward secrecy (PFS), key compromise impersonation (KCI) and server compromise impersonation (SCI). Also, they proposed a modular IoT-AKE framework and concrete post-quantum schemes for IoT-AKE. However, the LTZ model does not consider the compromise of ephemeral secret keys, and their schemes rely on the random oracle model (ROM). In this paper, we extend the LTZ model to capture the compromise of ephemeral secret keys, and propose generic constructions of IoT-AKE in the ROM and in the standard model (StdM), respectively. Our constructions allow us to construct the first post-quantum IoT-AKE scheme (from isogenies, lattice, etc.) in the StdM.

Keywords: authenticated key exchange, IoT-eCK, Internet of Things, server compromise impersonation resilience, post-quantum

1. Introduction

Authenticated Key Exchange (AKE) is a cryptographic protocol to share a common session key among multiple parties through an unauthenticated channel such as the Internet. We focus on the 2-party AKE. In ordinary PKI-based AKE, each party locally keeps a static secret key (SSK) and publishes a static public key (SPK) corresponding to the SSK. The validity of the SPK is guaranteed by a certificate issued by the certification authority. In a key exchange session, each party generates an

[1] 茨城大学
Ibaraki University
[a)] 21nm706r@vc.ibaraki.ac.jp
[b)] kazuki.yoneyama.sec@vc.ibaraki.ac.jp

ephemeral secret key (ESK) and sends an ephemeral public key (EPK) corresponding to the ESK to the other party. The session key is derived from these keys and some key derivation function like a hash function. Ordinary AKE is intended to satisfy session key secrecy and mutual authentication, and the provable security is formulated by security models such as the eCK model [13]. The basic security properties for AKEs are the known-key security in which no information about the session key is revealed if other session keys are revealed, and the (perfect) forward secrecy in which the adversary cannot obtain any information of the session key even if SSKs are revealed. Most existing AKE protocols have only a single type of authentication factor (e.g. certified secret/public keys or passwords). Therefore, if the single authentication factor is compromised, the security of the AKE protocol will be broken after the compromise. For example of the classification of the authentication factors, there are human-memorable passwords, secret keys in a secure module, biometrics, etc. By using two or multiple factors in the protocol, we can make the protocols more secure against the compromise. Also, in ordinary AKE, each party has the same type of the authentication factor, and key exchange sessions are performed with almost the same computational complexity for each party. Therefore, the protocols often distinguishes parties by the initiator or the responder.

On the other hand, there are situations where mutual authentication is performed in an environment in which the resources of both parties are asymmetric like the communication between a server and an IoT-device. In this paper, we focus on the AKE protocols for IoT systems (IoT-AKE), where an IoT-device and a server run the protocol to authenticate each other and share a session key without human involvement. We suppose that the IoT-device is standalone and no human user is necessarily present when it is engaged in the protocol execution. In the IoT environment, the IoT-devices are well known to be constrained with computation capability, network bandwidth, and battery life. Thus, lightweight designs are desirable. Regarding the security requirements for IoT-AKE, we argue that the following properties are required in addition to the standard properties described above.

- Key Compromise Impersonation (KCI): Even if an adversary has compromised one party's SSK, then it still cannot impersonate the other party to this party.
- Server Compromise Impersonation (SCI): Even if an adversary has compromised the server, then it still cannot impersonate the server to the IoT-device.

These attacks are usually caused in real-world applications. Regarding the KCI, an IoT-device is very likely to be compromised and has the stored credentials leaked by attacks such as side-channel attacks. Regarding the SCI, the server can be deployed in critical infrastructure, it is necessary that an adversary should not possibly impersonate the server to the IoT-device even if the adversary has compromised the server. The ordinary security models of AKE cannot cover such properties. These properties are difficult to achieve within the single authentication factor. Thus, we focus on the AKE based on two or multi-factors.

1.1 Related Work

1.1.1 IoT-AKE

Chan et al. [5] argued that big-data could be a candidate of the good authentication factor, and proposed a big-data based unilateral two-factor authentication protocol. Liu et al. [14] formulated a security model (LTZ model) for IoT-AKE using big-data in the bounded-retrieval model (BRM). It means that the server uses big-data as a secret factor and the adversary can only reveal a subset of big-data. The LTZ model captures the security properties including KCI and SCI, and they proposed two concrete schemes based on Diffie-Hellman (DH) KE. Then, Liu et al. [15] proposed a modular IoT-AKE framework by replacing the part of the DH KE in the concrete schemes to a generic PKE and a passively-secure KE, and showed post-quantum instantiations in the LTZ model and in the ROM. However, there are three problems in their schemes. First, their schemes are proved in the random oracle model (ROM). Random oracles do not exist in the real world, and cannot always be instantiated by real hash functions. Indeed, Canetti et al. [4] show that there are primitives which are secure in the ROM but insecure if random oracles are instantiated by real hash functions. Next, since the LTZ model does not consider the compromise of the ESKs used in the session, their schemes cannot guarantee the security against the compromise of ESKs. For example, an ESK can be obtained to the adversary if a weak pseudo-random number generator is implemented. Hence, it is desirable to consider not only the leakage of SSKs but also the leakage of ESKs. Finally, though their schemes use a message authentication code (MAC) for explicit authentication, implicit authentication is enough to satisfy the LTZ model. Thus, removing such a MAC can make IoT-AKE schemes more simple and efficient.

1.2 Our Contribution

In this paper, we achieve the first post-quantum IoT-AKE scheme without random oracles. Our contribution is three fold.

- We introduce the IoT-eCK model by extending the LTZ model based on the eCK model [13]. Thus, our model can capture the compromise of ESKs, and it can guarantee stronger security.
- We propose a generic construction (GC-StdM) in the IoT-eCK model in the standard model (StdM) from an IND-CCA secure KEM and an IND-CPA secure KEM. By instantiating GC-StdM with DH-based or factoring-based KEM schemes, we can obtain the first IoT-AKE in the StdM. Also, by instantiating GC-StdM with lattice-based or isogeny-based KEM schemes, we can obtain the first post-quantum IoT-AKE in the StdM. Moreover, we also propose a generic construction (GC-ROM) in the IoT-eCK model in the ROM from an OW-CCA secure KEM and an OW-CPA secure KEM.
- By omitting the SSK for MAC of both parties and the seed, our generic constructions are simplified compared to existing IoT-AKE schemes. For more details, please see Section 3.2.

2. Security Model for IoT-AKE

In this section, we extend the LTZ security model [14] for IoT-AKE to the IoT-eCK model with SCI and KCI resistance, based on the eCK model [13] by LaMacchia, Lauter, and Mityagin.

2.1 System Model and Adversarial Capacity

2.1.1 Notation

- $x \in_R X$: The element x is sampled uniformly randomly from the set X.
- π_P^i : The i-th instance of a party U_P.
- pid : The partner identifier, where the server' s identifier is denoted as ID_S and the IoT-device's identifier is denoted as ID_C.
- sid : The session identifier, and each sid should be unique within the party.
- sk : The session key derived by π_P^i at the end of the protocol execution. It is initialized as $\perp$.
- acc : The state of acceptance $acc \in \{accepted, rejected, \perp\}$ which represents the state of π_P^i at the end of the protocol execution. It is initialized as $\perp$, is set as $accepted$ if the instance successfully completes the protocol execution, and is set as $rejected$ otherwise.

2.1.2 IoT-AKE Setting

Parties are modeled as probabilistic polynomial-time Turing machines (PPTM). Each party is activated by receiving an initialization message and returns a message defined by the protocol.

IoT-AKE uses the following two authentication factors in the two-party communication between the server U_S and the IoT-device U_C.

(1) Secret keys in a secure module such as a static shared secret key between U_C and U_S and a static certified secret key of the server.

(2) A dataset $\mathbb{D}$ of big-data, which contains a large number of data items denoted as d_i $(1 \le i \le m)$ for some m. In the setup phase on U_C with ID_C, U_S chooses L data from $\{d_i\}$ randomly as a subset of $\mathbb{D}$ to be used in the protocol, generates a tag $t_{ID_C,i}$ for each d_i using the secret keys, and stores it in $\mathbb{D}^*$.

U_S sends a subset of its secret keys to U_C as SSK_C, and U_S keeps some secret keys and $\mathbb{D}^*$ as SSK_S. An example of a dataset $\mathbb{D}^*$ in the case of using n IoT-devices and big-data which contains m data is shown in the following Table 1.

表 1 An example of a dataset $\mathbb{D}^*$

d_i \ ID	ID_1	ID_2	...	ID_{n-1}	ID_n
d_1	$t_{ID_1,1}$	-	...	$t_{ID_{n-1},1}$	-
d_2	$t_{ID_1,2}$	-	...	-	$t_{ID_n,2}$
d_3	-	$t_{ID_2,3}$	...	$t_{ID_{n-1},3}$	-
$\vdots$	...	...	...	...	...
d_{m-2}	-	$t_{ID_2,m-2}$	...	-	$t_{ID_n,m-2}$
d_{m-1}	$t_{ID_1,m-1}$	-	...	-	$t_{ID_n,m-1}$
d_m	-	$t_{ID_2,m}$	...	$t_{ID_{n-1},m}$	-

2.1.3 Protocol and sessions

Each execution of the protocol is called a session, and each session has an instance π assigned to the party, where each π must be unique within the party. Each instance is associated with a session state containing intermediate values, and the i-th instance of π by party U_P is denoted by π_P^i. Also, if the instance π_P^i completes the session by computing the session key sk, π_P^i is set as $\pi_P^i.acc = accepted$.

In addition, we define the matching conversation for the IoT-device instance π_C^i and the server instance π_S^j as follows.

Difinition2.1 (Matching conversation) Two instances π_C^i and π_S^j are said to be matching if they satisfy

the following conditions.

(1) $\pi_C^i.acc = \pi_S^j.acc = accepted$

(2) $\pi_C^i.sid = \pi_S^j.sid$

(3) $\pi_C^i.sk = \pi_S^j.sk$

(4) $\pi_C^i.pid = U_S \wedge \pi_S^j.pid = U_C$

2.1.4 Adversary

Let *params* be a public parameter. The adversary $\mathcal{A}$ is modeled as a PPTM, which takes *params* as input and has oracle access to parties $U_{P_1}, \ldots, U_{P_n}$. $\mathcal{A}$ controls all communications among users including the session activation. $\mathcal{A}$ can interfere in party U_P to execute a specific action using the following adversary's queries.

- $\mathsf{Send}(msg, \pi_P^i)$: The adversary sends an arbitrary message msg to the instance π_P^i. π_P^i executes the protocol according to the given message and returns the response to $\mathcal{A}$. When activating a session, the adversary queries $msg = null$.
- $\mathsf{SKReveal}(\pi_P^i)$: When $\pi_P^i.acc = accepted$, the adversary obtains the session key SK of the instance π_P^i.
- $\mathsf{ESKReveal}(\pi_P^i)$: If $\pi_P^i.acc \neq accepted$, the adversary obtains the ephemeral secret key ESK_P^i of the instance π_P^i.
- $\mathsf{Corrupt}_C(ID_C)$: The adversary obtains the static secret key SSK_C of the IoT-device U_C with ID_C.
- $\mathsf{Corrupt}_S(\mathbb{I}_\mathcal{A}, ID_S)$: The adversary obtains the static secret key SSK_S of the server S, where it allows leakage of the dataset $(d_i, (t_{ID_1,i}, \ldots, t_{ID_n,i}))$ indicated by the index $i \in \mathbb{I}_\mathcal{A}$ in $\mathbb{D}^*$ and secret keys. In the BRM, $\mathbb{I}_\mathcal{A}$ has a limited size.

2.2 IoT-eCK Security

For defining the IoT-eCK security, we need the notion of freshness.

Difinition2.2 (Freshness) Let π_P^i $(P \in \{U_C, U_S\})$ and $\pi_{P'}^j$ be instances between an IoT-device U_C and a server U_S, where $\pi_P^i.acc = \pi_{P'}^j.acc = accepted$. If there exists a matching conversation of π_P^i, let $\pi_{P'}^j$ be the matching instance of π_P^i. We say that the instance π_P^i is IoT-eCK fresh if none of the following conditions hold:

(1) The adversary issues $\mathsf{SKReveal}(\pi_P^i)$, or $\mathsf{SKReveal}(\pi_{P'}^j)$ query if $\pi_{P'}^j$ exists,

(2) The adversary $\mathcal{A}$ issues $\mathsf{Corrupt}_S(\mathbb{I}_\mathcal{A}, ID_S)$ query after issuing $\mathsf{Send}(\pi_P^i)$ query.

(3) $\pi_{P'}^j$ exists and the adversary makes either of the following queries
- both $\mathsf{Corrupt}_P$ and $\mathsf{ESKReveal}(\pi_P^i)$ query, or
- both $\mathsf{Corrupt}_{P'}$ and $\mathsf{ESKReveal}(\pi_{P'}^j)$ query,

(4) π_C^j does not exist and the adversary makes either of the following queries
- $\mathsf{Corrupt}_C(ID_C)$ query, or
- both $\mathsf{Corrupt}_S(\mathbb{I}_\mathcal{A}, ID_S)$ and $\mathsf{ESKReveal}(\pi_S^j)$ query,

(5) π_S^j does not exist and the adversary makes the following query
- both $\mathsf{Corrupt}_C(ID_C)$ and $\mathsf{ESKReveal}(\pi_C^i)$ query, or
- both $\mathsf{Corrupt}_C(ID_C)$ and $\mathsf{Corrupt}_S(\mathbb{I}_\mathcal{A}, ID_S)$ query.

The goal of the adversary $\mathcal{A}$ in the IoT-eCK security game is to distinguish the true session key from a random key. $\mathcal{A}$ can make any sequence of the queries described above. During the experiment, $\mathcal{A}$ makes the following query.

- $\mathsf{Test}(\pi_P^i)$: Here, π_P^i must be IoT-eCK fresh. The oracle chooses $b \in_R \{0, 1\}$. If $b = 0$, then it returns $\pi_P^i.sk$. Otherwise, it returns a random key. This query can be issued only once.

The adversary $\mathcal{A}$ obtains either the session key of π_P^i or a random key with probability 1/2 respectively. After issuing the Test query, the game continues until $\mathcal{A}$ outputs b' as a result of guessing whether the received key is random or not. If π_P^i is IoT-eCK fresh by the end and the guess of $\mathcal{A}$ is correct (i.e., $b = b'$), then $\mathcal{A}$ wins the game.

Difinition2.3 (IoT-eCK security) The advantage of the adversary $\mathcal{A}$ in the above game with the IoT-AKE protocol Π is defined as follows.

$$Adv_\Pi^{IoT-AKE}(\mathcal{A}) = |\Pr[\mathcal{A}\ win] - 1/2|$$

Let κ be a security parameter. We say that an IoT-AKE protocol Π is secure in the IoT-eCK model, if the following conditions hold:

(1) Correctness : If two parties are complete matching instances, they both compute the same session key except with a negligible probability in κ.

(2) Security : For any PPT adversary $\mathcal{A}$, $Adv_{\Pi,\kappa}^{IoT-eCK}(\mathcal{A})$ is negligible in κ.

Remark1 In comparison to the eCK model [13], the conditions 2 and 5 of the Definition 2.2 are different. The condition 2 prevents the trivial attack of $\mathcal{A}$ querying the $\mathsf{Corrupt}_S$ query for $\mathbb{I}$ of matching instances. The condition 5 is less restrictive than the original eCK model to capture the SCI.

Remark2 In Definition 2.2, PFS by the leakage pattern of both parties' SSKs in the condition 3, KCI for impersonating an IoT-device by the leakage pattern of the server's SSK in the condition 4, KCI for impersonating a server by the leakage pattern of the IoT-device's SSK in the condition 5, and SCI by the leakage pattern of the server's SSK and the ESK of π_C^i in the condition 5. In comparison to the LTZ model [14], $\mathcal{A}$ can obtain the

ESK of the impersonating server's peer, thus it captures stronger SCI than the original one.

3. Our Generic Constructions

In this section, we propose two generic constructions of IoT-AKE from KEM in the StdM (GC-StdM) and in the ROM (GC-ROM). GC-StdM is based on IND-CCA secure KEM and IND-CPA secure KEM, and GC-ROM is based on OW-CCA secure KEM and OW-CPA secure KEM. Our constructions are secure in the IoT-eCK model. The protocol of GC-StdM is shown in Section. 3.3. Since GC-ROM can be constructed in a similar way as GC-StdM, we omit the protocol of GC-ROM due to the page limitation. The proposed protocols contain two factors as SSKs, (i) the distributed credentials in the setup phase to prove the data-tag relationship and (2) the decapsulation key of the CCA secure KEM to prove its knowledge for the server.

3.1 Protocol of LTZ scheme [14]

First, we revisit the protocol of the LTZ scheme [14] using big-data based on the CDH and SDH assumptions.

Public Parameters : Let κ be a security parameter, $\mathbb{G}$ be the multiplicative group, g be a generator of $\mathbb{G}$, q be a prime order of $\mathbb{G}$, $\mathsf{F} : \{0,1\}^* \times \{0,1\}^\kappa \to \mathbb{Z}_q$ and $\mathsf{E} : \{0,1\}^\kappa \times \{0,1\}^\kappa \to \{0,1\}^\kappa$ be pseudo-random functions, $\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_q$ be a random oracle, and index parameter z is determined.

Initialization : The server U_S generates a public/private key pair $(pk = g^{sk}, sk)$ where $sk \in_R \mathbb{Z}_q$, and also generates $mk \in_R \{0,1\}^\kappa$ as a static secret shared key, $K \in_R \mathbb{Z}_q$, and generates $K' \in_R \{0,1\}^\kappa$ for tag generation and data processing. Suppose U_S possesses dataset $\mathbb{D}$ which contains L data items d_i $(1 \le i \le L)$. For each data item $d_i \in \mathbb{D}$, the server generates its tag as $t_i = K \cdot \mathsf{H}(d_i) + \mathsf{F}(i, K')$. it defines dataset $\mathbb{D}^*$ which contains all tuples (d_i, t_i) $(1 \le i \le L)$ of a data item and tag.

Key Exchange :

(1) The IoT-device U_C chooses $r_1 \in_R \mathbb{Z}_q^*$ to compute $a = pk^{r_1}$ and $g' = g^{r_1}$. Then, it chooses $r_2 \in_R \{0,1\}^\kappa$ and a random subset $\mathbb{I}_C$ of z distinct indices from $[1, L]$, and then sends $(g', r_2, \mathbb{I}_C)$ and $M_1 = \mathsf{H}(mk, a, g', r_2, \mathbb{I}_C)$ to the server U_S.

(2) Upon receiving $msg_1 = (g', r_2, \mathbb{I}_C, M_1)$, U_S computes $a^* = g'^{sk}$ and verifies whether or not $M_1 = \mathsf{H}(mk, a^*, g', r_2, \mathbb{I}_C)$ holds. If the verification passes, it chooses a random subset $\mathbb{I}_S$ of z distinct indices which should be disjoint from $\mathbb{I}_C$. Also, it computes $r_2' = \mathsf{E}(r_2, mk)$, $X = K \cdot \sum_{i \in \mathbb{I}}^{i} (\mathsf{H}(d_i) \cdot \mathsf{F}(i, r_2'))$, and $Y = \sum_{i \in \mathbb{I}}^{i} (t_i \cdot \mathsf{F}(i, r_2'))$, where $\mathbb{I} = \mathbb{I}_C \cup \mathbb{I}_S$. Besides, U_S chooses $r_3 \in_R \mathbb{Z}_q$ to compute $b = pk^{r_3}$ and $dh = a^{*r_3}$. Then U_S sends $(b, \mathbb{I}_S, X)$ and $M_2 = \mathsf{H}(a^*, b, dh, \mathbb{I}_S, X, Y, msg_1)$ to U_C.

(3) Upon receiving $msg_2 = (b, \mathbb{I}_S, X, M_2)$, U_C computes $r_2' = \mathsf{E}(r_2, mk)$ and $K_{\mathbb{I}} = \sum_{i \in \mathbb{I}}^{i} (\mathsf{F}(i, K') \cdot \mathsf{F}(i, s'))$, where $\mathbb{I} = \mathbb{I}_C \cup \mathbb{I}_S$. Also, it computes $Y = X + K_{\mathbb{I}}$ and $dh^* = b^{r_1}$, and verifies whether or not $M_2 = \mathsf{H}(a, b, dh^*, \mathbb{I}_S, X, Y, msg_1)$ holds. If the verification passes, it computes $M_3 = \mathsf{H}(a, b, dh^*, \mathbb{I}, Y, msg_1, msg_2)$ and sends it to U_S. Finally, it computes a session key $SK = \mathsf{H}(mk, a, b, dh^*, Y)$.

(4) Upon receiving M_3, U_S verifies whether or not $M_3 = \mathsf{H}(a^*, b, dh, \mathbb{I}, Y, msg_1, msg_2)$ holds. If the verification passes, it computes a session key $SK = \mathsf{H}(mk, a^*, b, dh, Y)$.

3.2 Our construction Idea

Here, we give our idea to achieve generic constructions in the ROM and in the StdM by comparing with the LTZ scheme.

3.2.1 Implicit authentication

The LTZ scheme provides the explicit authentication of the server and the IoT-device with the key confirmation by M_1, M_2 and M_3 as MACs. As discussed in Section 1.1, the implicit authentication is sufficient to satisfy the security in the IoT-eCK model, thus our generic constructions can omit the MAC authentication. If the explicit authentication is required for an implementation, it is trivial to be able to add it to our constructions by a similar key confirmation step.

3.2.2 Replacement of random oracles

Our constructions are also based on big-data operation as the LTZ scheme. In GC-StdM, by replacing the random oracle H used in big-data operation with a key derivation function or pseudo-random function, we can construct a scheme without ROs. Also, by using the technique [9] of generating session keys with pseudo-random functions, we can generate session keys without random oracles.

3.2.3 Omission of static shared secret key

In the LTZ scheme, the static shared secret key mk is used to compute r_2', and $\mathsf{F}(i, r_2')$ is used to compute X, Y, and $K_{\mathbb{I}}$. However, since the adversary $\mathcal{A}$ cannot obtain K' in the event that both parties SSKs are not compromised from the freshness definition, it can be reduced to the se-

curity of PRF F of $\mathsf{F}(i, K')$. Thus we can omit mk and the computation of r'_2. Also, in the LTZ scheme, the nonce r_2 is used to change Y explicitly per session, however, $\mathcal{A}$ can query the same r_2 in a Send query to the server. Thus the randomization by r_2 is redundant in the viewpoint of the security, and we can omit r_2.

3.2.4 Processing big-data

In the LTZ scheme, the size of $\mathbb{D}^*$ is expanded by storing the $L \times m$ pairs of the duplicated data items and tag into $\mathbb{D}^*$ in the initialization phase, and they call it big-data. This is a fairly redundant storing method in terms of the real-world implementation and does not fit the original meaning of big-data. Therefore, we change the storing method as follows. In the initialization phase, the server chooses L data for each device as a subset of big-data containing m data items, stores the generated tags in association with the data, and uses the subset in the protocol. Since $\mathbb{D}^*$ contains distinct data items, our method captures the original meaning of big-data.

3.3 GC-StdM: IoT-AKE in Standard Model

The protocol in the StdM consists of an IND-CCA secure KEM $(\mathsf{KeyGen}, \mathsf{EnCap}, \mathsf{DeCap})$ and an IND-CPA secure KEM $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ as follows.

3.3.1 Protocol

Public Parameters : Let κ be a security parameter, $\mathsf{F} : \{0,1\}^* \times \mathcal{FS} \to \mathbb{Z}_q$ and $\mathsf{PRF} : \{0,1\}^* \times \mathcal{FS} \to \{0,1\}^\kappa$ be pseudo-random functions, $\mathsf{KDF} : Salt \times \mathbb{D} \to \mathbb{Z}_q$ and $\mathsf{KDF}' : Salt' \times \mathcal{KS} \to \mathcal{FS}$ be key derivation functions and $st \in_R Salt$ and $st' \in_R Salt'$ are chosen, $\mathsf{TCR}_{\mathcal{FS}} : \mathbb{Z}_q \to \mathcal{FS}$ be a target-collision resilience hash function, and index parameter z is determined, where $\mathcal{FS}$ is a key space of the pseudo-random functions ($|\mathcal{FS}| = 2^\kappa$), $\mathcal{KS}$ is a session key space of KEM, and $Salt$ and $Salt'$ are salt space of the key derivation functions. These are provided as part of the public parameters.

Initialization : The server U_S chooses randomness $r \in_R \mathcal{RS}^{cca}_{\mathcal{G}}$, computes $(ek_S, dk_S) \leftarrow \mathsf{KeyGen}(1^\kappa; r)$, and for party U_{id} generates $K_{id} \in_R \mathbb{Z}_q$ and $K'_{id} \in_R \mathcal{FS}$ for tag generation and data processing. Suppose U_S possesses a secret dataset $\mathbb{D}$ which contains m data items d_i $(1 \le i \le m)$. U_S chooses index set $\mathcal{I}_{id}$ which contains L indices to use a subset of $\mathbb{D}$ in the protocol, generates tag as $t_{id,i} = K_{id} \cdot \mathsf{KDF}(st, d_i) + \mathsf{F}(i, K'_{id})$ for every data item d_i $(i_1 \le i \le i_L)$, stores $t_{id,i}$ into secret dataset $\mathbb{D}^*$. U_S sends K'_{id} to the IoT-device U_{id} secretly and erases K'_{id}. U_S's static secret keys are $(dk, K_{id}, \mathbb{D}^*)$, and U_{id}'s static secret keys is K'_{id}.

Key Exchange : Let U_S which has static keys $SSK_S := (dk, \{K_{id}\}_n, \mathbb{D}^*), SPK_S := (ek, \{\mathcal{I}_{id}\}_n)$ and an identifier ID_S be a server, and U_C which has a static secret key $SSK_C := K'_{ID_C}$ and an identifier ID_C be a IoT-device.

(1) U_C chooses ephemeral secret keys $r_1 \in_R \mathcal{RS}^{cca}_{\mathcal{E}}$ and $r_2 \in_R \mathcal{RS}^{cpa}_{\mathcal{G}}$, computes $(CT_C, KE_C) \leftarrow \mathsf{EnCap}(ek; r_1)$ and $(ek_T, dk_T) \leftarrow \mathsf{wKeyGen}(r_2)$, and chooses a random subset $\mathbb{I}_C \in_R \mathcal{I}_{ID_C}$ of z distinct indices for $\mathbb{D}^*$. Then, U_C sends $(ek_T, CT_C, \mathbb{I}_C, ID_C)$ to U_S.

(2) Upon receiving $(ek_T, CT_C, \mathbb{I}_C, ID_C)$, U_S chooses an ephemeral secret key $r_3 \in_R \mathcal{RS}^{cpa}_{\mathcal{E}}$, and computes $(CT_T, KE_T) \leftarrow \mathsf{wEnCap}(ek_T; r_3)$ and $KE_C \leftarrow \mathsf{DeCap}(CT_C, dk)$. Also, U_S chooses a random subset $\mathbb{I}_S \in_R \mathcal{I}_{ID_C}$ of z distinct indices which should be disjoint from $\mathbb{I}_C$, sets $\mathbb{I} = \mathbb{I}_C \cup \mathbb{I}_S$, computes $X = K_{ID_C} \cdot \sum^i_{i \in \mathbb{I}}(\mathsf{KDF}(st, d_i))$. Then U_S sends $(CT_T, \mathbb{I}_S, X)$ to U_C. U_S computes $Y = \sum^i_{i \in \mathbb{I}}(t_{ID_C,i})$, $KE'_C \leftarrow \mathsf{KDF}'(st', KE_C)$, $KE'_T \leftarrow \mathsf{KDF}'(st', KE_T)$, and $Y' \leftarrow \mathsf{TCR}_{\mathcal{FS}}(Y)$. Then, U_S sets $sid = (ek_T, CT_C, CT_T, \mathbb{I}_C, \mathbb{I}_S, X)$ and computes a session key $SK = \mathsf{PRF}(sid, KE'_C) \oplus \mathsf{PRF}(sid, KE'_T \oplus \mathsf{PRF}(sid, Y'))$.

(3) Upon receiving $(CT_T, \mathbb{I}_S, X)$, U_C sets $\mathbb{I} = \mathbb{I}_C \cup \mathbb{I}_S$, computes $K_{\mathbb{I}} = \sum^i_{i \in \mathbb{I}}(\mathsf{F}(i, K'_{ID_C}))$, $Y = X + K_{\mathbb{I}}$, $KE_T \leftarrow \mathsf{wDeCap}(CT_T, dk_T)$, $KE'_C \leftarrow \mathsf{KDF}'(st', KE_C)$, $KE'_T \leftarrow \mathsf{KDF}'(st', KE_T)$, and $Y' \leftarrow \mathsf{TCR}_{\mathcal{FS}}(Y)$. Then, U_C sets $sid = (ek_T, CT_C, CT_T, \mathbb{I}_C, \mathbb{I}_S, X)$ and computes a session key $SK = \mathsf{PRF}(sid, KE'_C) \oplus \mathsf{PRF}(sid, KE'_T \oplus \mathsf{PRF}(sid, Y'))$.

3.3.2 Security

We show the security of the proposed scheme in the StdM. An intuition of the proof is shown in Section 3.4.

Theorem3.1 If $(\mathsf{KeyGen}, \mathsf{EnCap}, \mathsf{DeCap})$ is an IND-CCA secure and κ-min-entropy KEM, $(\mathsf{wKeyGen}, \mathsf{wEnCap}, \mathsf{wDeCap})$ is an IND-CPA secure and κ-min-entropy KEM, F and PRF are pseudo-random functions, KDF and KDF' are key derivation functions, and $\mathsf{TCR}_{\mathcal{FS}}$ is a target-collision resilience hash function, GC-StdM is IoT-eCK secure.

We show the proof of Theorem 3.1 in the full version.

3.4 Sketch of proof

The strategies of an adversary $\mathcal{A}$ against GC-ROM and GC-StdM can be categorized from the freshness definition as shown in the Table 2.

表 2 Strategies of $\mathcal{A}$

strategy	matching	exist	SSK_S	ESK_S	SSK_C	ESK_C
ST1	No	U_S	✓	×	×	-
ST2	No	U_S	-	✓	×	-
ST3	No	U_C	×	-	✓	×
ST4	No	U_C	✓	-	×	✓
ST5	Yes	both	✓	×	✓	×
ST6	Yes	both	×	✓	×	✓
ST7	Yes	both	×	✓	✓	×
ST8	Yes	both	✓	×	×	✓

"Yes" means the tested instance has a peer oracles and "No" means it has none. "✓" means the secret key is revealed to the adversary, "×" means the secret key is not revealed. "-" means the instance state is not defined.

In ST1 and ST8, the adversary $\mathcal{A}$ can choose the ephemeral key or reveal the ESK for the IoT-device U_C, but cannot reveal K'_{ID_C} and all information in $\mathbb{D}^*$. Thus, $\mathcal{A}$ cannot compute Y. In ST2 and ST6, $\mathcal{A}$ can choose the ephemeral key or reveal the ESK for the server U_C, but cannot reveal K'_{ID_C} and $\mathbb{D}^*$. Thus, $\mathcal{A}$ cannot compute Y. In ST3 and ST7, $\mathcal{A}$ can choose the ephemeral key or reveal the ESK for the server U_S, but cannot reveal dk. Thus, $\mathcal{A}$ cannot compute KE_C which is the session key of the CCA secure KEM. In ST4, $\mathcal{A}$ can choose the ephemeral key the ESK for the server U_S, but cannot reveal K'_{ID_C} and all information in $\mathbb{D}^*$. Thus, $\mathcal{A}$ cannot compute Y. In ST5, $\mathcal{A}$ can reveal SSKs for both parties, but cannot reveal ESK. Thus, $\mathcal{A}$ cannot compute KE_T which is the session key of the CPA secure KEM. Therefore, the proposed constructions satisfy the IoT-eCK security.

4. Instantiations from DH, Factoring and RSA

A comparison of the efficiency among our DH-based instantiations and the LTZ scheme is shown in Table 3.

4.1 Random Oracle Model

4.1.1 DH-based

We can obtain an IoT-AKE scheme in the ROM by instantiating GC-ROM using the PSEC-KEM [19] which is an OW-CCA secure KEM, and the ElGamal KEM which is an OW-CPA secure KEM. The PSEC-KEM and the ElGamal KEM are obviously κ-min-entropy KEM. Since these KEM schemes are based on the computational DH (CDH) assumption, the instantiation is secure under the CDH assumption though the scheme [14] relies on the SDH assumption. Also, the communication complexity is smaller than the LTZ scheme.

4.2 Standard Model

4.2.1 DH-based

We can obtain an IoT-AKE scheme in the StdM by instantiating GC-StdM using CS3 [7] which is an IND-CCA secure KEM, and the ElGamal KEM which is an IND-CPA secure KEM. CS3 is obviously κ-min-entropy KEM. Since these KEM schemes are based on the decisional DH (DDH) assumption, the instantiation is also secure under the DDH assumption. This scheme is the first DH-based IoT-AKE scheme in the StdM. Moreover, the communication complexity is smaller than the LTZ scheme.

4.2.2 Factoring-based and RSA-based

We can obtain an IoT-AKE scheme based on the hardness of the integer factorization problems in the StdM. For example, we use the Hofheinz and Kiltz's PKE [10] and the Mei et al.'s PKE [16] which are IND-CCA secure PKE under the factoring assumption. Furthermore, from [8], by applying the fact [11] that it is also secure under the CDH assumption if a scheme is secure under the CDH assumption in $\mathbb{Z}_N^*$, we can obtain more efficient factoring-based KEM schemes from IND-CCA secure KEM under the CDH assumption.

Also, we can obtain an IoT-AKE scheme based on the hardness of RSA inversion in the StdM. For example, we use the Chevallier-Mames and Joye's PKE [6] and the Kiltz et al.'s PKE [12] which are IND-CCA secure PKE under the instance-independent RSA assumption.

The PKE scheme can be transformed into the KEM scheme by using internally generated randomness instead of the plaintext in the PKE. When the plaintext space in the PKE is larger than a security parameter κ and plaintexts are uniformly chosen randomness, it is obvious that such a KEM scheme is κ-min-entropy KEM. These schemes are the first IoT-AKE scheme under the factorization assumption or the RSA-type assumption in the StdM.

Remark3 We can construct an IoT-AKE from the factorization assumptions and the RSA-type assumptions as described above, however, the key size becomes larger than the DH-based scheme. Thus, the communication costs could become a bottleneck for the IoT-device.

5. Instantiations from post-quantum schemes in Standard Model

5.1 Lattice-based

We can obtain an IoT-AKE scheme based on the worst-case of the (ring-)LWE problems in the StdM. For example, we use PKE [1] which is an IND-CCA secure PKE un-

表 3 Comparison among LTZ scheme and our instantiations

Protocol	Model	Resource	Assumption	Exponentiation (IoT-Device)	Exponentiation (Server)	Communication complexity
LTZ [14]	LTZ	ROM	SDH	3	3	$\|ID\| + 6\|G\| + \kappa + \|\mathbb{I}\|$
Ours1 4.1.1	IoT-eCK	ROM	CDH	4	4	$\|ID\| + 4\|G\| + \kappa + \|\mathbb{I}\|$
Ours2 4.2.1	IoT-eCK	StdM	DDH	6.08	4.16	$\|ID\| + 6\|G\| + \|\mathbb{I}\|$

For exponentiation costs, we apply the parallel computation technique [18] for two exponentiations using the same base, which costs 1.33 exponentiations for $\kappa = 128$, and Avanzi's algorithm [3] for multi-exponentiations in the elliptic curve setting, which costs 1.08 exponentiations for $\kappa = 128$. $|ID|$ is the length of IoT-device's ID, $|G|$ is the size of a group element, and $|\mathbb{I}|$ is the size of the subset $\mathbb{I}$.

der the ring-LWE assumption. The PKE schemes can be transformed into the KEM scheme in the same way. As for factoring-based PKE, lattice-based PKE schemes are also κ-min entropy KEM. This scheme is the first IoT-AKE scheme in the StdM under the ring-LWE assumption.

Unfortunately, the obtained AKE protocols are inefficient since these PKE schemes require huge keys, say, the quadratic or cubic order of the security parameter.

5.2 Isogeny-based

We can obtain a CSIDH-based IoT-AKE scheme in the StdM by instantiating GC-StdM using the KEM from smooth projective hashing [2] which is an IND-CCA secure KEM based on the hash proof system under the existence of weak pseudorandom effective group action (wPR-EGA) (i.e., a generalization of CSIDH assumptions), and a hashed CSIDH-KEM. The hashed CSIDH-KEM is a variant of CSIDH-KEM such that the session key is computed as the output of the entropy-smoothing hash function H on inputting the result of the group action of the randomness and the public key ($K = H([\mathfrak{r}] * pk)$) or the secret key and the ciphertext ($K = H([\mathfrak{s}] * C)$). As the same as the hashed ElGamal KEM [20], it is pointed out that the hashed CSIDH-KEM is IND-CPA secure under the CSSDDH assumption [17]. This scheme is the first post-quantum IoT-AKE scheme in the StdM under the wPR-EGA and the CSSDDH assumption.

参考文献

[1] Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: EUROCRYPT 2010. pp. 553–572 (2010)

[2] Alamati, N., Feo, L.D., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: ASIACRYPT 2020. pp. 411–439 (2020)

[3] Avanzi, R.M.: The Complexity of Certain Multi-Exponentiation Techniques in Cryptography. J. Cryptology pp. 357–373 (2005)

[4] Canetti, R., Goldreich, O., Halevi, S.: The Random Oracle Methodology, Revisited. J. ACM pp. 557–594 (2004)

[5] Chan, A.C., Wong, J.W., Zhou, J., Teo, J.C.M.: Scalable Two-Factor Authentication Using Historical Data. In: ESORICS 2016. pp. 91–110 (2016)

[6] Chevallier-Mames, B., Joye, M.: Chosen-Ciphertext Secure RSA-Type Cryptosystems. In: ProvSec 2009. pp. 32–46 (2009)

[7] Cramer, R., Shoup, V.: Design and Analysis of Practical Public-key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. SIAM J. Comput pp. 167–226 (2003)

[8] Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Practical and post-quantum authenticated key exchange from one-way secure key encapsulation mechanism. In: AsiaCCS 2013. pp. 83–94 (2013)

[9] Fujioka, A., Suzuki, K., Xagawa, K., Yoneyama, K.: Strongly secure authenticated key exchange from factoring, codes, and lattices. Designs, Codes and Cryptography pp. 469–504 (2015)

[10] Hofheinz, D., Kiltz, E.: Practical Chosen Ciphertext Secure Encryption from Factoring. In: EUROCRYPT 2009. pp. 313–332 (2009)

[11] Hofheinz, D., Kiltz, E.: The Group of Signed Quadratic Residues and Applications. In: CRYPTO 2009. pp. 637–653 (2009)

[12] Kiltz, E., Mohassel, P., O'Neill, A.: Adaptive Trapdoor Functions and Chosen-Ciphertext Security. In: EUROCRYPT 2010. pp. 673–692 (2010)

[13] LaMacchia, B., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: ProvSec 2007. pp. 1–16 (2007)

[14] Liu, B., Tang, Q., Zhou, J.: Bigdata-Facilitated Two-Party Authenticated Key Exchange for IoT. In: ISC 2021. pp. 95–116 (2021)

[15] Liu, B., Tang, Q., Zhou, J.: Modular Framework for Constructing IoT-Server AKE in Post-Quantum Setting. IEEE Access 2022 pp. 71598–71611 (2022)

[16] Mei, Q., Li, B., Lu, X., Jia, D.: Chosen Ciphertext Secure Encryption under Factoring Assumption Revisited. In: PKC 2011. pp. 210–227 (2011)

[17] Moriya, T., Onuki, H., Takagi, T.: Sigamal: A supersingular isogeny-based PKE and its application to a PRF. In: ASIACRYPT 2020. pp. 551–580 (2020)

[18] M' Raíhi, D., Naccache, D.: Batch Exponentiation: A Fast DLP-Based Signature Generation Strategy. In: ACM CCS 1996. p. 58–61 (1996)

[19] Shoup, V.: A Proposal for an ISO Standard for Public Key Encryption. IACR Cryptology ePrint Archive, Report 2001/112 (2001)

[20] Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. IACR Cryptology ePrint Archive, Report 2004/332 (2004)