View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by University of Liverpool Repository

Leakage-Resilient Authenticated Key Exchange for Edge Artificial Intelligence

Jie Zhang, Futai Zhang, Xin Huang, and Xin Liu

Abstract—Edge Artificial Intelligence (AI) is a timely complement of cloud-based AI. By introducing intelligence to the edge, it alleviates privacy concerns of streaming and storing data to the cloud, enables real-time operations where milliseconds matter, and brings AI services to remote areas with poor networking infrastructures. Security is a significant problem in Edge AI applications such as self-driving cars and intelligent healthcare. Since the edge devices are empowered to process data and take actions, attacking and compromising them can cause serious damage. However, the wide deployment of computationally limited devices in edge environments and the increasing happening of side-channel (or leakage) attacks pose critical challenges to security. This article thereby aims to enhance the security for Edge AI by designing and developing lightweight and leakage-resilient authenticated key exchange (LRAKE) protocols. Compared with available LRAKE protocols, the proposed protocols in this article can be effortless applied in some mainstreaming security and communication standards. Moreover, this article realizes prototypes and presents implementation details; and a use case of applying the proposed protocol in Bluetooth 5.0 is illustrated. The theoretical design and implementation details will provide a guidance of applying the LRAKE protocols in Edge AI applications.

Index Terms—Leakage-resilience, key exchange, side-channel attacks, edge computing, Edge AI

1 Introduction

Artificial Intelligence (AI) based on the powerful cloud platform is playing a significant role in the current information-based society [1], [2], [3]. However, issues such as privacy concerns, network delays and communication quality impede its application in a wide range of scenarios where privacy is required, milliseconds matter or network infrastructures are poor [4], [5], [6], [7]. The intelligent edge computing or Edge AI [8], [9], [10] is a timely complement of current AI supported by cloud computing. By introducing intelligence to the edge, it alleviates privacy concerns of streaming and storing data to the cloud, enables real-time operations, and brings AI services to remote areas with poor networking infrastructures. Edge AI is anticipated to facilitate a wide range of applications such as self-driving cars, intelligent healthcare, deep-sea exploration and military.

Security is a significant problem in Edge AI applications. First, since the edge devices are empowered to make decisions and take actions, compromising them can cause more serious damage than ever before. Second, applications such as self-driving cars directly and closely related to individuals' lives; therefore, wrong decisions or actions caused by security attacks could lead to traffic accidents with serious injuries. Finally, in applications such as military and healthcare, privacy concern is an essential requirement.

However, providing adequate security in edge environments is often challenging. Recently the situation becomes more severe due to the increasing happening of side-channel attacks (demonstrated by a number of work on side-channel attacks reported in 2018 and 2019 in top conferences [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]). In the edge environments there are many computationally limited edge and end devices such as gateways, routers, sensors and so on. Most of these devices can only provide very basic security due to their limited computational ability, and cannot resist the side-channel attacks which leak information from long-term secrets in side-channel manners such as recording and analyzing the timing [34], power [35] or electromagnetic-emission [36].

To guarantee security in Edge AI, one of the necessary procedures is to establish secure channels among devices by authenticated key exchange (AKE). Many AKE protocols are proposed so far; however, most of them cannot resist the side-channel attacks. Some leakage-resilient AKE (LRAKE) protocols that can resist side-channel attacks are proposed in recent years [37], [38], [39], [40], [41], but none of them are adopted in practice. This reflexes a gap between theoretical achievements and realworld applications. We investigated some of these theoretical work and identified two probable reasons. First, none of these work provides prototypes or implementation details. As a result, there is no specific guidance about how to implement LRAKE protocols in practice. Second, these available LRAKE protocols are very different from existing AKE protocols in some mainstreaming security and communication standards in use such as Transport Layer

- J. Zhang is with the School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China. E-mail: jie.zhang01@xjtlu.edu.cn.
- F. Zhang is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou 350117, China. E-mail: ffttzhang@sina.com.
- X. Huang is with the Taiyuan University of Technology, Taiyuan 030024, China, Soochow University, Suzhou 215006, China, and also with Suzhou L.J.T Intelligent Technology Co., Ltd., Suzhou 215006, China. E-mail: xin@huangstudio.org.
- X. Liu is with the School of Computer Science and Technology, Nanjing Normal University, Nanjing 210097, China. E-mail: 1361liuxin@163.com.

Manuscript received 5 June 2019; revised 2 Jan. 2020; accepted 6 Jan. 2020. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Futai Zhang.)
Digital Object Identifier no. 10.1109/TDSC.2020.2967703

1545-5971 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

IEEE Proof

Security (TLS), Bluetooth and so on. For example, there is an increasing adoption of Elliptic Curve Diffie-Hellman (ECDH)-based AKE protocols in many of these standards; however, no mature ECDH-based LRAKE protocols are presented in the literature. As a result, it is impractical to include existing non-ECDH-based LRAKE protocols in the future versions of these standards.

This paper aims to enhance security of Edge AI by presenting and developing lightweight LRAKE protocols with adequate security and reasonable computational cost. The main contributions are summarized as follows. First, an ECDH-based LRAKE protocol (named π_2) is presented in detail; and its security is proved in the continuous after-the-fact leakage-resilient extended Canetti-Krawczyk (CAFLR eCK) model which is the strongest security model available for AKE protocols. Second, a lightweight method is introduced and applied to Protocol π_2 to construct its lightweight variant which is more friendly to computationally limited devices. Third, prototypes are realized to evaluate the performance of these protocols; and implementation details are presented. Finally, a use case of applying the proposed protocols in the Bluetooth communication specification is illustrated. This demonstrates in great detail how to implement the protocols in practice, and thereby is critical to narrow the gap between theoretical designs and realworld applications.

The rest of this paper is organized as follows. Section 2 investigates available work on ECDH-based AKE and LRAKE in the literature. Section 3 introduces the closely related fundamentals of cryptography. Sections 4, 5 and 6 present the security models, propose the LRAKE protocol and prove its security respectively. Section 7 illustrates the construction of a lightweight LRAKE protocol. Section 8 evaluates the performance and presents implementation details. Section 9 demonstrates the application of the proposed protocols through a use case. Finally, Section 10 briefly summarizes the paper and discusses future work.

2 Related Work

This section reviews ECDH-based AKE protocols in some mainstreaming security and communication standards, and summarizes closely related work on LRAKE available in the literature.

2.1 ECDH-Based AKE Protocols

Diffie-Hellman (DH) key exchange is the basis for a number of AKE protocols adopted in many security or communication standards. Its elliptic curve version, the Elliptic Curve Diffie-Hellman key exchange [42], can provide stronger security with short keys, and thereby has become one of the most popular technique to implement industrial-grade AKE protocols, especially in edge networking environments where there exists a large number of computationally limited devices.

Below we summarize ECDH-based AKE protocols in some international standards.

2.1.1 TLS

TLS is the most widely used security standard in the Internet. It underlies the security of many higher-level protocols such as the Hyper Text Transfer Protocol over Secure Socket Layer (HTTPs) and the Message Queuing Telemetry Transport (MQTT) protocol. TLS uses the handshake protocol to establish a secure channel between two communicating parties. The latest version TLS 1.3 [43] includes an ECDH-based handshake protocol which is essentially an ECDH-based AKE protocol.

2.1.2 Bluetooth

Bluetooth is a wireless communicating technology for portable and/or fixed electronic devices. It is featured with short-range, robustness and low cost, and thereby is suitable for edge networks. The latest version Bluetooth Specification 5.0 [44] presents four secure simple pairing protocols based on ECDH. These protocols are ECDH-based AKE protocols with different authentication measures.

2.1.3 IEEE 802.15.6

IEEE 802.15.6 [45] is the international standard for wireless body area networks (WBANs). It includes a suite of authenticated association protocols that generate authenticated shared keys for a node and a hub: the public key hidden association, the password authenticated association and the display authenticated association. These protocols are essentially ECDH-based AKE protocols with different authentication measures.

Although the aforementioned AKE protocols are widely used in practice, none of them resists side-channel attacks. One probable reason is that the leakage-resilient cryptography in particular the leakage-resilient AKE is a relatively new research. There is a reasonable gap between theoretical achievements and realworld applications. In the following subsection we summarize some excellent theoretical work to identify the key to narrowing the gap.

2.2 LRAKE Protocols

2.2.1 Related Work on Security Model

Before side-channel attacks are studied, the strongest security model for AKE protocols is the eCK model [46]. Moriyama and Okamoto propose the first leakage-resilient version of eCK model and name it λ-LR eCK model [47]. It formalizes leakage from long-term secrets in the model, but does not consider the leakage after the test session is determined. Alawatugoda *et al.* study the after-the-fact leakage-resilient (AFLR) eCK model that models the leakage attacks after the determination of the test session [48], [49], [50], [51]. They propose the bounded AFLR (BAFLR) eCK model and the continuous AFLR (CAFLR) eCK model. The later is stronger since it assumes the overall leakage is not bounded. We highly appreciate these pioneering and outstanding work. In this paper we apply the CAFLR eCK model to prove the security of the proposed protocol.

2.2.2 Related Work on LRAKE Protocols

Shin *et al.* [37] propose password-based AKE protocols between a client and a server. By dividing the password into shares using a secret sharing scheme, the protocols can tolerate leakage of password or verification data stored in servers. However, their design does not resist leakage from computations. Ruan *et al.* [38], [39] propose a leakage-resilient password-based AKE protocol that uses the Dziembowski-Faust scheme

IEEE Proof

to split the password into two parts. The protocol assumes passwords are stored in both the client and the server, while in normal scenarios passwords are memorized and input by users in every protocol run on the client side. In [40] Ruan *et al.* propose a general framework for constructing identity-based AKE protocols in the BAFLR eCK security model, and show a formal proof in the standard model. Chen *et al.* [41] propose challenge-dependent leakage-resilient eCK (CLR-eCK) model and a framework of constructing provably secure AKE protocols under that model.

Although there are some LRAKE protocols available in the literature by now, none of these work involves a prototype. In addition, there is few mature work on ECDH-based LRAKE protocols. Probably due to the lacking of prototypes or implementation details of LRAKE protocols, in particular the ECDH-based ones, none of the aforementioned security and communication standards includes LRAKE protocols so far.

3 Preliminaries

This section introduces cryptographic primitives, including underlying difficult assumptions and the Dziembowski-Faust leakage- resilient storage scheme.

3.1 Elliptic Curve Key Exchange

3.1.1 Elliptic Curve Cryptography

One type of elliptic curve E that is suitable for cryptography is defined as follows:

$$y^2 = x^3 + ax + b \bmod p,$$

with $a, b \in GF(p)$ and $4a^3 + 27b^2 \neq 0$, where $GF(p)$ is prime finite field of order p [52].

We mainly use two operations

- Point Addition. Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be two points on E. The point addition between P and Q is denoted $P + Q$. The result is also a point on E.
- Scalar Multiplication. Let t be an integer and P be a point on E. The scalar multiplication between t and P is denoted by $t \cdot P$. It is defined as

$$t \cdot P = \underbrace{P + P + \cdots + P}_{t}.$$

When t is a large integer, computing a scalar multiplication is much more time-consuming than computing a point addition.

3.1.2 ECDH-Based Key Exchange

In the basic ECDH key exchange protocol, the two participants A and B have the common public parameters (E, G, n, p) where E is an ellptic curve defined in the prime finite field $GF(p)$, G is the base point of E, and n is the order of G.

To agree a shared key, A generates a random integer x_A, computes $X_A = x_A \cdot G$, and sends X_A to B. B generates a random integer x_B, computes $X_B = x_B \cdot G$, and sends X_B to A. A can compute the shared key as $K = x_A \cdot X_B$. B can compute the shared key as $K = x_B \cdot X_A$.

3.2 Difficult Assumptions

3.2.1 ECDLOG Assumption

Definition 1 (ECDLOG Problem). *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n. The elliptic curve discrete logarithm (ECDLOG) problem over E is to find x such that $X = x \cdot P$ given P and a randomly chosen X on E if such an x exists, denoted by $x = \mathrm{ECDLOG}(P, X)$.*

The ECDLOG assumption holds in E if for all probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the probability of solving the ECDLOG problem in E is negligible for a given security parameter $\mathtt{k}$ [53].

3.2.2 ECGDH Assumption

Definition 2 (ECCDH Problem). *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n, and X and Y be randomly chosen points on E such that $X = x \cdot P$ and $Y = y \cdot P$ for some unknown $x, y \in [0, n-1]$. Given P, X and Y, the elliptic curve computational Diffie-Hellman (ECCDH) problem is to find the point $Z = \mathrm{ECDLOG}(P, X) \cdot \mathrm{ECDLOG}(P, Y) \cdot P$, denoted by $Z = \mathrm{ECCDH}(P, X, Y)$.*

Definition 3 (ECDDH Problem). *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n, and X, Y and Z be randomly chosen points on E such that $X = x \cdot P$, $Y = y \cdot P$ and $Z = z \cdot P$ for some unknown $x, y, z \in [0, n-1]$. Given P, X, Y and Z, the elliptic curve decisional Diffie-Hellman (ECDDH) problem is to output 1 if $Z = \mathrm{ECCDH}(P, X, Y)$ and 0 otherwise. We use $\mathrm{ECDDH}(P, X, Y, Z)$ to denote $Z = \mathrm{ECDDH}(P, X, Y)$.*

Definition 4 (ECGDH Problem). *Let E be an elliptic curve defined over a finite field $GF(p)$, P be a point on E of order n, and X and Y be points on E such that $X = x \cdot P$ and $Y = y \cdot P$ for some unknown $x, y \in [0, n-1]$. Given P, X, Y and an oracle access to $\mathrm{ECDDH}(\cdot,\cdot,\cdot)$, the elliptic curve gap Diffie-Hellman (ECGDH) problem is to output $\mathrm{ECCDH}(P, X, Y)$.*

The ECGDH assumption holds in E if for all PPT algorithm $\mathcal{A}$, the probability of solving the ECGDH problem in E is neglibible for a given security parameter $\mathtt{k}$.

3.3 Leakage-Resilient Storage With Refreshing Protocol

The Dziembowski-Faust leakage-resilient storage [54] contains a storage scheme and a refreshing protocol. We will use it as a building block in our protocol to protect long-term secrets from side-channel attacks.

3.3.1 (λ, ϵ)-Secure Leakage-Resilient Storage Scheme

The storage scheme contains a pair of encode and decode algorithms. The encode algorithm splits a secret key sk into two separated parts sk_L and sk_R such that sk can be recovered from sk_L and sk_R through a decode algorithm. For any $m, n \in N$, the storage scheme $\Lambda^{n,m}_{Z_q^*} = (\mathrm{Encode}^{n,m}_{Z_q^*}, \mathrm{Decode}^{n,m}_{Z_q^*})$ stores secret $sk \in (Z_q^*)^m$ in the following manner

- $\mathrm{Encode}^{n,m}_{Z_q^*}(sk)$: $sk_L \xleftarrow{R} (Z_q^*)^n \backslash \{(0^n)\}$, then $sk_R \leftarrow (Z_q^*)^{n \times m}$ such that $sk_L \cdot sk_R = sk$ and output (sk_L, sk_R).
- $\mathrm{Decode}^{n,m}_{Z_q^*}(sk_L, sk_R)$: output $sk_L \cdot sk_R$.

IEEE Proof

After the encode algorithm sk is destroyed while sk_L and sk_R are stored separately and secretly. The decode algorithm will not be used in practice.

Definition 5 (λ-limited Adversary). *An adversary is defined as a λ-limited adversary if the amount of leakage obtained by him/her from sk_L and sk_R is limited to $\lambda = (\lambda_1, \lambda_2)$ bits in total.*

Definition 6 ((λ, ϵ)-secure Leakage-resilience of a Storage Scheme). *A storage scheme $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$ is (λ, ϵ)-secure leakage-resilient if for any random secrets sk_0 and sk_1 and any λ-limited adversary, the leakage from* Encode *$(sk_0) = (sk_{0L}, sk_{0R})$ and $\mathrm{Encode}(sk_1) = (sk_{1L}, sk_{1R})$ are statistically ϵ-close.*

Theorem 1 [54]. *Given that $n > 20 \cdot m$, the storage scheme $\Lambda^{n,m}_{Z_q^*} = (\mathrm{Encode}^{n,m}_{Z_q^*}, \mathrm{Decode}^{n,m}_{Z_q^*})$ is $(\lambda, negl(n))$-secure against an λ-limited adversary for some negligible function $negl$ and $\lambda = (0.3n\log q, 0.3n\log q)$*

3.3.2 (l, λ', ϵ')-Secure Leakage-Resilient Refreshing Protocol

The refresh protocol refreshes the encoding to defend against a continuous leakage. It updates sk_L and sk_R into sk'_L and sk'_R. The refresh protocol $\mathrm{Refresh}(sk_L, sk_R)$ works as follows.

- Refreshing sk_R.
  - $\overrightarrow{A_L} \xleftarrow{R} (Z_q^*)^n \backslash \{(0^n)\}$ and $\mathbb{B}_\mathbb{L} \leftarrow (Z_q^*)^{n\times m}$ where $\mathbb{B}_\mathbb{L}$ is full rank and $\overrightarrow{A_L} \cdot \mathbb{B}_\mathbb{L} = (0^m)$.
  - $\mathbb{M}_\mathbb{L} \leftarrow (Z_q^*)^{n\times n}$ where $\mathbb{M}_\mathbb{L}$ is non-singular and $sk_L \cdot \mathbb{M}_\mathbb{L} = \overrightarrow{A_L}$.
  - Compute $\mathbb{X} = \mathbb{M}_\mathbb{L} \cdot \mathbb{B}_\mathbb{L}$ and $sk'_R = sk_R + \mathbb{X}$.
- Refreshing sk_L.
  - $\overrightarrow{A_R} \xleftarrow{R} (Z_q^*)^n \backslash \{(0^n)\}$ and $\mathbb{B}_\mathbb{R} \leftarrow (Z_q^*)^{n\times m}$ where $\mathbb{B}_\mathbb{R}$ is full rank and $\overrightarrow{A_R} \cdot \mathbb{B}_\mathbb{R} = (0^m)$.
  - $\mathbb{M}_\mathbb{R} \leftarrow (Z_q^*)^{n\times n}$ where $\mathbb{M}_\mathbb{R}$ is non-singular and $\mathbb{M}_\mathbb{R} \cdot sk'_R = \mathbb{B}_\mathbb{R}$.
  - Compute $\mathbb{Y} = \overrightarrow{A_R} \cdot \mathbb{M}_\mathbb{R}$ and $sk'_L = sk_L + \mathbb{Y}$.

The refresh protocol is run after per computation involving sk_L and sk_R.

Definition 7 ($\lambda_{\mathrm{Refresh}}$-limited Adversary). *An adversary is defined as a λ-limited adversary if the amount of leakage obtained by him/her from sk_L and sk_R is limited to $\lambda = (\lambda_1, \lambda_2)$ bits in total.*

Definition 8 ((l, λ', ϵ')-secure Leakage-resilience of a Refreshing Protocol). *For a (λ, ϵ)-secure leakage-resilient storage scheme $\Lambda = (\mathrm{Encode}, \mathrm{Decode})$, a refresh protocol $\mathrm{Refresh}(sk_L, sk_R)$ is (l, λ', ϵ')-secure leakage-resilient if for any random secrets sk_0 and sk_1 and any $\lambda'_{Refresh}$-limited adversary against the protocol up to l rounds, the leakages from* Refresh *(sk_{0L}, sk_{0R}) and $\mathrm{Refresh}(sk_{1L}, sk_{1R})$ are statistically ϵ'-close.*

Theorem 2. *[54] Given that $n \geq m/3, n \geq 16, l \in N$ and $\Lambda^{n,m}_{Z_q^*}$ is a (λ, ϵ)-secure leakage-resilient storage scheme, the refreshing protocol $\mathrm{Refresh}^{n,m}_{Z_q^*}$ is $(l, \lambda/2, \epsilon')$-secure leakage-resilient for $\Lambda^{n,m}_{Z_q^*}$.*

4 Security Models

This section summarizes two security models for AKE protocols: 1) the eCK model [46] which is the strongest model before the arising of side-channel attacks and 2) its leakage-resilient version [48] which models side-channel attacks and continuous after-the-fact leakage.

4.1 eCK Model

4.1.1 Notations and Definitions

- Parties and long-term keys: $U = \{U_1, \ldots, U_{N_P}\}$ is a set of N_P parties. Each U_i ($i \in [1, N_P]$) has a pair of long-term public and secret keys (PK_{U_i}, sk_{U_i}). Each U_i owns at most N_S protocol sessions.
- Sessions: $\Pi^j_{U,V}$ represents the jth session at the owner U with intended partner V.
- Partnering: Two sessions $\Pi^j_{U,V}$ and $\Pi^{j'}_{V,U}$ are partners if all the following hold:
  - both $\Pi^j_{U,V}$ and $\Pi^{j'}_{V,U}$ have computed session keys;
  - messages sent from $\Pi^j_{U,V}$ are identical with that received by $\Pi^{j'}_{V,U}$;
  - messages sent from $\Pi^{j'}_{V,U}$ are identical with that received by $\Pi^j_{U,V}$;
  - exactly one of U and V is the initiator and the other is the responder.

4.1.2 Adversarial Power

The adversary $\mathcal{A}$ is a probabilistic polynomial time algorithm that can adaptively ask the following queries

- $\mathrm{Send}(U, V, j, m)$ query. This query allows $\mathcal{A}$ to run the protocol by sending message m to the session $\Pi^j_{U,V}$. It returns the next message according to the protocol conversation so far.
- $\mathrm{SessionKeyReveal}(U, V, j)$ query. This query allows $\mathcal{A}$ to reveal the session key of the session $\Pi^j_{U,V}$ if $\Pi^j_{U,V}$ has accepted a session key. It returns the session key of $\Pi^j_{U,V}$.
- $\mathrm{EphemeralKeyReveal}(U, V, j)$ query. This query allows $\mathcal{A}$ to reveal all the ephemeral secrets of the session $\Pi^j_{U,V}$. It returns ephemeral secrets of $\Pi^j_{U,V}$.
- $\mathrm{Corrupt}(U)$ query. This query allows $\mathcal{A}$ to corrupt a party U. It returns the long-term secrets of U.

4.1.3 Fresh Sessions

Freshness of sessions is defined to exclude corruptions which allow the adversary to trivially break any AKE protocol. A session $\Pi^j_{U,V}$ is fresh if and only if adversaries have not asked the following queries

- if partner session does not exist
  - $\mathrm{SessionKeyReveal}(U, V, j)$
  - $\mathrm{Corrupt}(U)$ and $\mathrm{EphemeralKeyReveal}(U, V, j)$
  - $\mathrm{Corrupt}(V)$
- if partner session $\Pi^{j'}_{V,U}$ exists
  - $\mathrm{SessionKeyReveal}(U, V, j)$
  - $\mathrm{SessionKeyReveal}(V, U, j')$
  - $\mathrm{Corrupt}(U)$ and $\mathrm{EphemeralKeyReveal}(U, V, j)$
  - $\mathrm{Corrupt}(V)$ and $\mathrm{EphemeralKeyReveal}(V, U, j')$

IEEE Proof

4.1.4 eCK Security Game

The eCK game simulates the attacks conducted by a PPT adversary given the adversarial power defined in Section 4.1.2.

- Initialization. The challenger generates keys using the security parameter $\mathtt{k}$.
- Queries. The adversary asks any of Send, $\mathrm{SessionKeyReveal}$, $\mathrm{EphemeralKeyReveal}$ and $\mathrm{Corrupt}$ queries to any session at will.
- Choosing test session. The adversary chooses a fresh session as the test session. The challenger chooses a random bit $b \in \{0,1\}$. If $b=1$ the actual session key of the test session is returned to the adversary, otherwise a random string is returned.
- Queries after choosing test session. The adversary asks any of Send, $\mathrm{SessionKeyReveal}$, $\mathrm{EphemeralKeyReveal}$ and $\mathrm{Corrupt}$ to any session at will.
- Guess. The adversary output the bit $b' \in \{0,1\}$. If $b' = b$ then the adversary wins the game.

4.1.5 Security Definition

Definition 9 (eCK Security). *A protocol π is secure in the eCK model if for any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ in winning the eCK game $\mathrm{Adv}_\pi^{\mathrm{eCK}}(\mathcal{A})$ is negligible in the security parameter $\mathtt{k}$. $\mathrm{Adv}_\pi^{\mathrm{eCK}}(\mathcal{A}) = |2Pr(\mathrm{Succ}_\mathcal{A}) - 1|$ and $Pr(\mathrm{Succ}_\mathcal{A})$ is the probability of $\mathcal{A}$ winning the eCK game.*

4.2 λ-CAFL-eCK Model

4.2.1 Modelling Leakage

The continuous leakage is model by a binary tuple of leakage functions $f = (f_{1i}, f_{2i})$ and a leakage parameter $\lambda = (\lambda_1, \lambda_2)$.

- $f = (f_{1i}, f_{2i})$ leaks information from each split of the long-term secrets at occurrence i
- $\lambda = (\lambda_1, \lambda_2)$ bounds the leakage of f_{1i} and f_{2i} to λ_1 and λ_2 respectively. The overall leakage of different occurrences is not bound.

4.2.2 Adversarial Power

The adversary is a PPT algorithm that can adaptively issue the following queries

- $\mathrm{Send}(U,V,j,m,f)$ query. This query returns the next message according to the protocol conversation along with the leakage $f(sk_U)$.
- $\mathrm{Session-Keyreveal}(U,V,j)$ query. This query returns the session key of $\Pi_{U,V}^j$.
- $\mathrm{Ephemeral-Keyreveal}(U,V,j)$ query. This query returns ephemeral keys of $\Pi_{U,V}^j$.
- $\mathrm{Corrupt}(U)$ query. This query returns the long-term secrets of U.

4.2.3 Fresh Sessions

A session is fresh if and only if all of the following hold

- if partner session does not exist, adversaries have not issued the following queries
  - $\mathrm{SessionKeyReveal}(U,V,j)$
  - $\mathrm{Corrupt}(U)$ and $\mathrm{EphemeralKeyReveal}(U,V,j)$
  - $\mathrm{Corrupt}(V)$
- if partner session $\Pi_{V,U}^{j'}$ exists, adversaries have not issued the following queries
  - $\mathrm{SessionKeyReveal}(U,V,j)$
  - $\mathrm{SessionKeyReveal}(V,U,j')$
  - $\mathrm{Corrupt}(U)$ and $\mathrm{EphemeralKeyReveal}(U,V,j)$
  - $\mathrm{Corrupt}(V)$ and $\mathrm{EphemeralKeyReveal}(V,U,j')$
- for each $\mathrm{Send}(U,\cdot,\cdot,f)$ query, the leakage from each split of the long-term secrets at occurrence i is bounded by $\lambda = (\lambda_1, \lambda_2)$, i.e., $|f_{1i}(sk_{U_L})| \le \lambda_1$ and $|f_{2i}(sk_{U_R})| \le \lambda_2$
- for each $\mathrm{Send}(V,\cdot,\cdot,f)$ query, the leakage from each split of the long-term secrets at occurrence i is bounded by $\lambda = (\lambda_1, \lambda_2)$, i.e., $|f_{1i}(sk_{V_L})| \le \lambda_1$ and $|f_{2i}(sk_{V_R})| \le \lambda_2$

4.2.4 λ-CAFL-eCK Security Game

The λ-CAFL-eCK security game simulates the attacks conducted by a PPT adversary given the adversarial power defined in Section 4.2.2. The procedure is similar as that of eCK secure game.

4.2.5 Security Definition

Definition 10 (λ-CAFL-eCK Security). *A protocol π is secure in the λ-CAFL-eCK model if for any PPT adversary $\mathcal{A}$, the advantage $\mathrm{Adv}_\pi^{\lambda-\mathrm{CAFL-eCK}}(\mathcal{A})$ of $\mathcal{A}$ in winning the λ-CAFL-eCK game is negligible in the security parameter $\mathtt{k}$. $\mathrm{Adv}_\pi^{\lambda-\mathrm{CAFL-eCK}}(\mathcal{A}) = |2Pr(\mathrm{Succ}_\mathcal{A}) - 1|$ and $Pr(\mathrm{Succ}_\mathcal{A})$ is the probability of $\mathcal{A}$ winning the λ-CAFL-eCK game.*

Theorem 3. *[51] A key exchange protocol P2 is λ-CAFL-eCK-secure if the underlying key exchange protocol P1 is eCK-secure, and the underlying leakage-resilient storage scheme $\Lambda_{Z_q^*}^{n,1}$ is $(2\lambda, \epsilon)$-secure leakage-resilient and the refreshing protocol $\mathrm{Refresh}_{Z_q^*}^{n,1}$ is (l, λ, ϵ')-secure leakage-resilient for some leakage limit $\lambda = (\lambda_1, \lambda_2)$, negligible values ϵ and ϵ' and positive integer l. The advantage $\mathrm{Adv}_{P2}^{\lambda-\mathrm{CAFL-eCK}}(\mathcal{A})$ of a PPT adversary $\mathcal{A}$ against P2 in the λ-CAFL-eCK secure game is $\le N_P(\mathrm{Adv}_{P1}^{\mathrm{eCK}}(\mathcal{A}) + \epsilon')$.*

5 Leakage-Resilient AKE Protocols

In this section, we first present an underlying protocol π_1 and then construct the λ-CAFL-eCK-secure protocol π_2 based on π_1 and the leakage-resilient storage scheme.

5.1 Protocol π_1

Our underlying protocol π_1 is an enhanced version of the YS-ECDH key exchange protocol [55]. Suppose A is the initiator and B is the responder. A and B have the common public parameters (E, G, n, p, H_1, H_2), where E is an elliptic curve defined over the prime finite field $GF(p)$, G is a base point of E, n is the order of G, and $H_1 : Z_n \to Z_n$ and $H_2 : E \times E \times E \to Z_n$ are two independent hash functions. We use the assumption that ECDLOG and ECGDH hold in E. Let sk_A and PK_A be the private and public keys of A where $PK_A = sk_A \cdot G$, and sk_B and $PK_B = sk_B \cdot G$ be the private and public keys of B. Initially, A and B hold their own private and public keys and the public key of each other. The protocol is presented as follows:

IEEE Proof

- A generates a random integer r_A, computes $u_A = r_A + sk_A$, $h_A = H_1(u_A)$ and $H_A = h_A \cdot G$, and sends H_A to B.
- B generates a random integer r_B, computes $u_B = r_B + sk_B$, $h_B = H_1(u_B)$ and $H_B = h_B \cdot G$, and sends H_B to A.
- A computes the shared key as follows:

$$K_1 = sk_A \cdot H_B$$
$$K_2 = h_A \cdot PK_B$$
$$K_3 = h_A \cdot H_B$$
$$k = H_2(K_1, K_2, K_3).$$

B computes the shared key as follows

$$K_1 = h_B \cdot PK_A$$
$$K_2 = sk_B \cdot H_A$$
$$K_3 = h_B \cdot H_A$$
$$k = H_2(K_1, K_2, K_3).$$

After the protocol, A and B hold the shared key k. Below we briefly explain why the two k computed by A and B are identical.

First, the K_1 computed by A and B are identical since

$$\begin{aligned} K_1 &= sk_A \cdot H_B \\ &= sk_A \cdot h_B \cdot G \\ &= h_B \cdot sk_A \cdot G \\ &= h_B \cdot PK_A. \end{aligned}$$

Second, the K_2 computed by A and B are identical since

$$\begin{aligned} K_2 &= h_A \cdot PK_B \\ &= h_A \cdot sk_B \cdot G \\ &= sk_B \cdot h_A \cdot G \\ &= sk_B \cdot H_A. \end{aligned}$$

Finally, the K_3 computed by A and B are identical since

$$\begin{aligned} K_3 &= h_A \cdot H_B \\ &= h_A \cdot h_B \cdot G \\ &= h_B \cdot h_A \cdot G \\ &= h_B \cdot H_A. \end{aligned}$$

5.2 Protocol π_2

Protocol π_2 is based on π_1 and applies the leakage-resilient storage scheme to protect long-term secret keys sk_A and sk_B by splitting them into two parts and refreshing them per computation. It includes an initialization procedure which splits the private keys into two shares, a key exchange procedure which exchanges messages and generates the shared key, and a refreshing procedure which refreshes the private key shares. Let the public parameters (E, G, n, p, H_1, H_2) have the same meaning as in Protocol π_1. The protocol is presented as follows.

5.2.1 Initialization

- A runs the encode algorithm $\mathrm{Encode}^{n,1}_{Z_q^*}$ to encode sk_A into two n-dimensional vectors $\overrightarrow{sk_{AL}} = (sk_{AL_1}, \ldots, sk_{AL_n})$ and $\overrightarrow{sk_{AR}} = (sk_{AR_1}, \ldots, sk_{AR_n})$. Then A stores $\overrightarrow{sk_{AL}}$ and $\overrightarrow{sk_{AR}}$ independently and destroys sk_A.
- B runs the encode algorithm $\mathrm{Encode}^{n,1}_{Z_q^*}$ to encode sk_B into two n-dimensional vectors $\overrightarrow{sk_{BL}} = (sk_{BL_1}, \ldots, sk_{BL_n})$ and $\overrightarrow{sk_{BR}} = (sk_{BR_1}, \ldots, sk_{BR_n})$. Then B stores $\overrightarrow{sk_{BL}}$ and $\overrightarrow{sk_{BR}}$ independently and destroys sk_B.

5.2.2 Key Exchange

- A chooses a random value r_A, sets

$$\overrightarrow{v_A} = (r_A, sk_{AL_1}, \ldots, sk_{AL_n})$$
$$\overrightarrow{w_A} = (1, sk_{AR_1}, \ldots, sk_{AR_n}),$$

and computes the following values

$$u_A = \overrightarrow{v_A} \cdot \overrightarrow{w_A}^\top$$
$$h_A = H_1(u_A)$$
$$H_A = h_A \cdot G.$$

Then A sends H_A to B.

- B chooses a random value r_B, sets

$$\overrightarrow{v_B} = (r_B, sk_{BL_1}, \ldots, sk_{BL_n})$$
$$\overrightarrow{w_B} = (1, sk_{BR_1}, \ldots, sk_{BR_n}),$$

and computes the following values

$$u_B = \overrightarrow{v_B} \cdot \overrightarrow{w_B}^\top$$
$$h_B = H_1(u_B)$$
$$H_B = h_B \cdot G.$$

Then B sends H_B to A.

- A computes the shared key k as follows[1]

$$temp_A \xleftarrow{R} Z_q^*$$
$$\overrightarrow{k_1^1} = temp_A \cdot \overrightarrow{sk_{AL}}$$
$$k_1^2 = \overrightarrow{k_1^1} \cdot \overrightarrow{sk_{AR}}^\top$$
$$K_1^3 = k_1^2 \cdot H_B$$
$$K_1 = \frac{1}{temp_A} \cdot K_1^3$$
$$K_2 = h_A \cdot PK_B$$
$$K_3 = h_A \cdot H_B$$
$$k = H_2(K_1, K_2, K_3),$$

1. We use a temporary random value $temp_A$ to hind sk_A in compuations.

IEEE Proof

B computes the shared key k as follows

$$K_1 = h_B \cdot PK_A$$

$$temp_B \xleftarrow{R} Z_q^*$$

$$\overrightarrow{k_2^1} = temp_B \cdot \overrightarrow{sk_{BL}}$$

$$k_2^2 = \overrightarrow{k_2^1} \cdot \overrightarrow{sk_{BR}}^\top$$

$$K_2^3 = k_2^2 \cdot H_A$$

$$K_2 = \frac{1}{temp_B} \cdot K_2^3$$

$$K_3 = h_B \cdot H_A$$

$$k = H_2(K_1, K_2, K_3).$$

5.2.3 Refreshing

- A runs the refresh protocol $\mathrm{Refresh}_{Z_q^*}^{n,1}$ to refresh (sk_{AL}, sk_{AR}).
- B runs the refresh protocol $\mathrm{Refresh}_{Z_q^*}^{n,1}$ to refresh (sk_{BL}, sk_{BR}).

5.2.4 Correctness of Protocol π_2

The session keys computed by A and B are identical.

First, K_1 computed by A and B are identical since

$$\begin{aligned} K_1 &= \frac{1}{temp_A} \cdot K_1^3 \\ &= \frac{1}{temp_A} \cdot k_1^2 \cdot H_B \\ &= \frac{1}{temp_A} \cdot \overrightarrow{k_1^1} \cdot \overrightarrow{sk_{AR}}^\top \cdot H_B \\ &= \frac{1}{temp_A} \cdot temp_A \cdot \overrightarrow{sk_{AL}} \cdot \overrightarrow{sk_{AR}}^\top \cdot H_B \\ &= \overrightarrow{sk_{AL}} \cdot \overrightarrow{sk_{AR}}^\top \cdot H_B \\ &= sk_A \cdot H_B \\ &= sk_A \cdot h_B \cdot G \\ &= h_B \cdot PK_A. \end{aligned}$$

Second, K_2 computed by A and B are identical since

$$\begin{aligned} K_2 &= \frac{1}{temp_B} \cdot K_2^3 \\ &= \frac{1}{temp_B} \cdot k_2^2 \cdot H_A \\ &= \frac{1}{temp_B} \cdot \overrightarrow{k_2^1} \cdot \overrightarrow{sk_{BR}}^\top \cdot H_A \\ &= \frac{1}{temp_B} \cdot temp_B \cdot \overrightarrow{sk_{BL}} \cdot \overrightarrow{sk_{BR}}^\top \cdot H_A \\ &= \overrightarrow{sk_{BL}} \cdot \overrightarrow{sk_{BR}}^\top \cdot H_A \\ &= sk_B \cdot H_A \\ &= sk_B \cdot h_A \cdot G \\ &= h_A \cdot PK_B. \end{aligned}$$

Finally, K_3 computed by A and B are identical since

$$\begin{aligned} K_3 &= h_A \cdot H_B \\ &= h_A \cdot h_B \cdot G \\ &= h_B \cdot h_A \cdot G \\ &= h_B \cdot H_A. \end{aligned}$$

6 Security Proof

This section proves the λ-CAFL-eCK security of Protocol π_2 given that $n > 20$.

6.1 eCK-Security of Protocol π_1

Here we claim the eCK security of Protocol π_1 in Theorem 4 and provide a proof sketch. The detailed proof is given in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TDSC.2020.2967703.

Theorem 4. *Protocol π_1 is eCK-secure under the ECGDH assumption if H_1 and H_2 are modeled by independent random oracles.*

Proof. Let $\mathcal{M}$ be a PPT adversary against Protocol π_1 that runs in time $\leq t$, involves $\leq N_P$ honest parties and actives $\leq N_S$ sessions.

First, to prove the eCK security of Protocol π_1, we need to prove that the advantage of $\mathcal{M}$ in the eCK security game (denoted as $\mathrm{Adv}_{\pi_1}^{\mathrm{eCK}}(\mathcal{M})$) is negligible.

Second, to prove $\mathrm{Adv}_{\pi_1}^{\mathrm{eCK}}(\mathcal{M})$ is negligible, we construct a ECGDH solver $\mathcal{S}$ using $\mathcal{M}$ as a subroutine and prove that the advantage of $\mathcal{S}$ in solving the ECGDH problem is

$$\mathrm{Adv}^{\mathrm{ECGDH}}(\mathcal{S}) \geq \frac{1}{2} \cdot \min\left\{\frac{2}{{N_S}^2}, \frac{1}{N_P \cdot N_S}\right\} \cdot \mathrm{Adv}_{\pi_1}^{\mathrm{eCK}}(\mathcal{M}).$$

The construction of $\mathcal{S}$ is as follows. $\mathcal{S}$ executes the eCK security game with $\mathcal{M}$ and modifies the data returned by the honest parties in such a way that if $\mathcal{M}$ wins the eCK experiment, then $\mathcal{S}$ can reveal the solution to the ECGDH problem.

Finally, under the ECGDH assumption, $\mathrm{Adv}_{\mathrm{ECGDH}}(\mathcal{S})$ is negligible. Therefore, $\mathrm{Adv}_{\pi_1}^{\mathrm{eCK}}(\mathcal{M})$ is negligible and Protocol π_1 has eCK security. □

6.2 λ-CAFL-eCK Security of Protocol π_2

Theorem 5. *Given that the underlying protocol π_1 is eCK-secure and $n > 20$, Protocol π_2 is λ-CAFL-eCK-secure with $\lambda = (0.15n\log q, 0.15n\log q)$.*

Proof. First, since in Protocol π_2 $m = 1$, the condition $n > 20$ guarantees that $n > 20 \cdot m$, $n \geq m/3$ and $n > 16$. Therefore, according to Theorems 1 and 2, the leakage-resilient storage $\Lambda_{Z_q^*}^{n,1}$ is $(2\lambda, \epsilon)$-secure leakage-resilient and the refreshing protocol $\mathrm{Refresh}_{Z_q^*}^{n,1}$ is (l, λ, ϵ') for $l \in N$, negligible ϵ and ϵ' and $\lambda = (0.15n\log q, 0.15n\log q)$.

Second, since the underlying protocol π_1 is eCK-secure, the underlying leakage-resilient storage scheme $\Lambda_{Z_q^*}^{n,1}$ is $(2\lambda, \epsilon)$-secure leakage-resilient and the underlying refreshing protocol $\mathrm{Refresh}_{Z_q^*}^{n,1}$ is (l, λ, ϵ')-secure leakage-resilient,

IEEE Proof

the advantage of a PPT adversary $\mathcal{A}$ against π_2 in the λ-CAFL-eCK secure game is

$$\mathrm{Adv}_{\pi_2}^{\lambda-\mathrm{CAFL-eCK}}(\mathcal{A}) \leq N_P \cdot (\mathrm{Adv}_{\pi_1}^{\mathrm{eCK}}(\mathcal{A}) + \epsilon'),$$

according to Theorem 3.

Therefore, Protocol π_2 is λ-CAFL-eCK-secure with $\lambda = (0.15n \log q, 0.15n \log q)$. □

6.3 Leakage Tolerance of Protocol π_2

In Protocol π_2, the length of the private key sk_A is $\log q$ bits; and its two parts sk_{AL} and sk_{AR} are both of size $n \log q$ bits. When $n > 20$, according to Theorem 5, the leakage parameter is $\lambda = (0.15n \log q, 0.15n \log q)$ which means the leakages of sk_A and sk_B are up to $0.15n \log q$ bits respectively. The leakage tolerance is $\frac{0.15n \log q}{n \log q} \times 100\% = 15\%$ for both sk_{AL} and sk_{AR}. It means Protocol π_2 can tolerate 15 percent leakage from two parts of privates keys in every protocol session. The overall leakage is unbounded since continuous leakage is allowed.

7 Light-Weighting Protocol π_1 and π_2

This section introduces a method which transfers computations from one party to its partner in an AKE protocol. We have designed several lightweight AKE protocols using this method [56], [57], [58]. In application scenarios where the two communicating parties have great disparity in computational power, the method will remarkably reduce the burden on the weak side, and thereby improve the overall performance.

We first introduce the lightweight construction of Protocol π_1 and π_2. Then we discuss the security concerns and propose countermeasures.

7.1 Light-Weight Versions of Protocol π_1 and π_2

7.1.1 Light-Weighting of Protocol π_1

Suppose the initiator A is a computationally limited party and the responder B is a powerful one. A and B have the common public parameters (E, G, n, p, H) where (E, G, n, p) have the same meaning as in Protocol π_1 and $H : E \to Z_n$ is a hash function. The key exchange procedure in the light-weight version of Protocol π_1 is presented as follows.

1. A generates a random integer r_A, computes $u_A = r_A + sk_A$, and sends u_A to B.
2. B generates a random integer r_B, computes $u_B = r_B + sk_B$ and $U_B = u_B \cdot G$, and sends U_B to A.
3. A computes the shared key as follows:

$$K_{temp} = r_A \cdot (U_B - PK_B)$$

$$k = H(K_{temp}),$$

B computes the shared key as follows:

$$U_A = u_A \cdot G$$

$$K_{temp} = r_B \cdot (U_A - PK_A)$$

$$k = H(K_{temp}).$$

The light-weight version of Protocol π_1 reduces the computational burden on A by transferring an elliptic curve scalar multiplication from A to B. With the same method, we can reduce the burden on B for scenarios that the responder is much less powerful than the initiator.

7.1.2 Light-Weighting of Protocol π_2

Still suppose A is much less powerful than B. A and B have the common public parameters (E, G, n, p, H) where (E, G, n, p) have the same meaning as in Protocol π_2 and $H : E \to Z_n$ is a hash function. The key exchange procedure in the light-weight version of Protocol π_2 is presented as follows.

- A chooses a random value R_A, sets

$$\overrightarrow{v_A} = (r_A, sk_{AL_1}, \ldots, sk_{AL_n})$$

$$\overrightarrow{w_A} = (1, sk_{AR_1}, \ldots, sk_{AR_n}),$$

and computes

$$u_A = \overrightarrow{v_A} \cdot \overrightarrow{w_A}^\top.$$

Then A sends u_A to B.

- B chooses a random value r_B, sets

$$\overrightarrow{v_B} = (r_B, sk_{BL_1}, \ldots, sk_{BL_n})$$

$$\overrightarrow{w_B} = (1, sk_{BR_1}, \ldots, sk_{BR_n}),$$

and computes

$$u_B = \overrightarrow{v_B} \cdot \overrightarrow{w_B}^\top$$

$$U_B = u_B \cdot G.$$

Then B sends U_B to A.

- A computes the shared key k as follows

$$K_{temp} = r_A \cdot (U_B - PK_B)$$

$$k = H(K_{temp}),$$

B computes the shared key K as follows

$$U_A = u_A \cdot G$$

$$K_{temp} = r_B \cdot (U_A - PK_A)$$

$$k = H(K_{temp}).$$

7.2 Discussion

The reducing of computations on one of the parties will inevitably weaken its security. For example, in the light-weight version of Protocol π_1, after alleviate the computations on A, an attacker C can impersonate B and establish a shared session key with A as follows

- A generates random integer r_A, computes $u_A = r_A + sk_A$, and sends u_A to B.
- B generates random integer r_B, computes $u_B = r_B + sk_B$ and $U_B = u_B \cdot G$, and sends U_B to A.

  At this step, C intercepts the message from B to A and replaces U_B with $U_C = r_C \cdot G + PK_B$ for some random value r_C generated by C

IEEE Proof

TABLE 1
Evaluation: Numbers of Scalar Multiplications

	π_1	π_2	light-weight π_1	light-weight π_2
A	4	5	1	1
B	4	5	3	3
Overall	8	10	4	4

- A computes the shared key as follows:

$$K_{temp} = r_A \cdot (U_C - PK_B)$$

$$k = H(K_{temp}),$$

C computes the shared key as follows:

$$U_A = u_A \cdot G$$

$$K_{temp} = r_C \cdot (U_A - PK_A)$$

$$k = H(K_{temp}).$$

At the end of key exchange, A and C compute the same session key $k = H(K_{temp}) = H(r_A \cdot r_C \cdot G)$.

To remove the above attack, a countermeasure is to add an authentication mechanism such as message authentication codes or digital signatures for A authenticating B; this will increase computations a bit. Another countermeasure is to hide PK_B from attackers; this method does not increase computations but might be inconvenient in practice for some scenarios.

8 Performance

This section evaluates the performance in terms of computation for Protocol π_1, π_2 and their light-weight versions. We first theoretically evaluate the computations by counting the numbers of time-consuming operations; then we carry out a set of experiments to test the performance in practice.

8.1 Evaluation

We count the number of elliptic curve scalar multiplications of each protocol. The results are summarized in Table 1.

According to the table, the two light-weight protocols require less scalar multiplications on A and B compared with Protocol π_1 and π_2. Meanwhile, they require less scalar multiplication on A than on B. We can draw the following two conclusions:

- the lightweight versions have better performance in terms of computations, compared with Protocol π_1 and π_2; and
- the lightweight versions are more friendly to A.

In the next subsection we use a set of experiments to verify the above conclusions and to show to what extent the light-weight versions have improved the performance compared with Protocol π_1 and π_2.

TABLE 2
Experimental Environment: Software

Item	Implementation Details
Programming Language	Python 2.7
Communication	Socket programming with TCP
Elliptic Curve	FIPS P-192, P-256, P-384 and P-521
Hash Function	MD5

8.2 Experiments

8.2.1 Setup

We realize prototypes of Protocol π_1, π_2 and their light-weight versions using Python programming language. The hash functions are realized through Message-Digest Algorithm (MD5). The communication is realized through socket programming with Transmission Control Protocol (TCP).

The experimental environment is explained in Tables 2 and 3. In the experiments, we run the prototypes on four recommended curves in Federal Information Processing Standards (FIPS) [59], [60]: P-192, P-256, P-384 and P-521. The two communicating parties A and B are simulated by two virtual machines with the same configuration run on the same laptop (Table 3).

8.2.2 Results and Analysis

In the experiments, we run the prototypes for ten times between two virtual machines. The average runtime is analyzed in Fig. 1.

According to Fig. 1, for all the four elliptic curves, Protocol π_2 is the most time-consuming protocol. The two lightweight versions have less computing time on A than on B; and they have much less overall computing time than both Protocol π_1 and π_2. The experimental results accord with the theoretical evaluation in Table 1.

9 Use Case

This section demonstrates how to apply the proposed protocols in Bluetooth. The original protocol in Bluetooth 5.0 is set as benchmark and compared with the leakage-resilient ones.

9.1 Overview

Bluetooth is a significant wireless communication technique in the edge networks. It connects smart devices in Edge AI applications such as smart building, smart city, smart industrial and so on. ECDH-based AKE protocols are supported in the latest version Bluetooth specification 5.0. The protocols are called Secure Simple Pairing protocols in the specification. They basically includes five phases as follows.

TABLE 3
Experimental Environment: Hardware

Device	Operating System	Base Memory	Storage	CPU
A	Ubuntu 16.04.3 (64-bit)	1,024 MB	10 GB	1 CPU
B	Ubuntu 16.04.3 (64-bit)	1,024 MB	10 GB	1 CPU
Laptop	Windows 10 (64-bit)	8 GB	256 G	Intel(R) Core(TM) i5-8250U @ 1.60 GHz 1.80 GHz

IEEE Proof

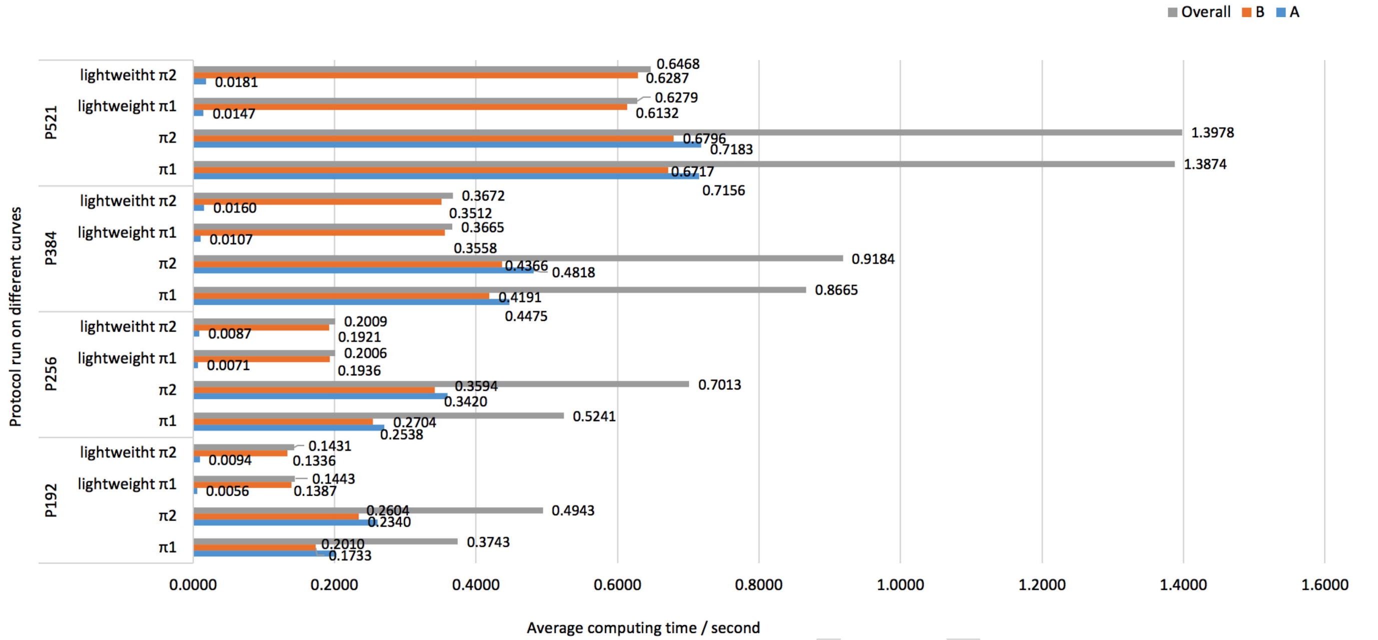


Fig. 1. Average computing time of Protocol π_1, π_2 and their lightweight versions on P-192, P-256, P-384 and P-521.

- Phase 1: Public Key Exchange. The devices generate their ECDH public and private keys and exchange the public keys.
- Phase 2: Authentication Stage 1. The devices select and exchange random values, and authenticate the exchanged data in Phase 1 and 2.
- Phase 3: Authentication Stage 2. The devices compute the shared key (DHKey) and check if they have the same DHKey.
- Phase 4: Link Key Calculation. The devices derive the link key from the DHKey.
- Phase 5: Link Manager Protocol Authentication and Encryption. This phase includes authentication and generation of the encryption key.

The first four phases constitute a basic ECDH-based AKE protocol that does not resist side-channel attacks. In the following two subsections we demonstrate how to apply our LRAKE protocols in Bluetooth 5.0.

9.2 Application of Protocol π_2

Suppose the initiator A and the responder B are two Bluetooth-connected devices with similar computational power. We apply Protocol π_2 in the Secure Pairing procedures as follows.

- Phase 1. This phase applies the initialization procedure of Protocol π_2. After the generation of ECDH public and private keys, each private key is encoded into two 21-dimension vectors. The vectors are securely stored and the private keys are destroyed. Then A and B exchange the public keys.
- Phase 2. This phase applies the key exchange and refreshing procedures of Protocol π_2, except that the last step of key exchange procedure is not executed.
- Phase 3. This phase applies the last step of key exchange procedure in Protocol π_2 to compute the DHKey.
- Phase 4 and Phase 5 are the same as those in Bluetooth 5.0.

Compared with the Secure Simple Pairing in Bluetooth 5.0, in each protocol run the above procedures can tolerate up to 15 percent leakage from each 21-dimension vector.

9.3 Application of Light-Weighting Protocol π_2

Now, suppose the initiator A is a computationally limited sensor and the responder B is a gateway that is more

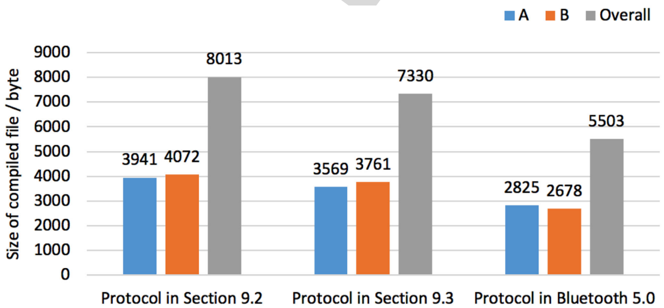


Fig. 2. Size of compiled files.

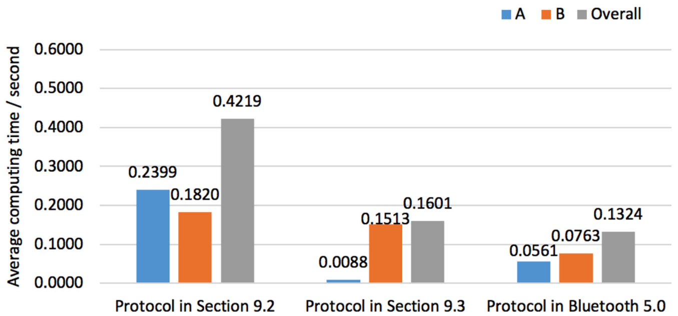


Fig. 3. Average computing time of Protocols in Section 9.2, Section 9.3 and Bluetooth 5.0 on P-256.

TABLE 4
Comparison With Secure Simple Pairing in Bluetooth 5.0

AKA Protocol	Security	Storage Requirement	Average Computing Time
Protocol in Section 9.2	Resisting side-channel attacks	Size of compiled files: 8,013 bytes	0.4219 seconds
Protocol in Section 9.3	Resisting side-channel attacks	Size of compiled files: 7,330 bytes	0.1601 seconds
Protocol in Bluetooth 5.0	Being vulnerable to side-channel attacks	Size of complied files: 5,503 bytes	0.1324 seconds

powerful than A. We apply the lightweight version of Protocol π_2 in the Secure Pairing procedures as follows.

- Phase 1. This phase is the same as Phase 1 in Section 9.2.
- Phase 2. This phase is the same as Phase 2 in Section 9.2 excepted that the key exchange procedure adopted here is the one of lightweight Protocol π_2.
- Phase 3. This phase applies the last step of key exchange procedure in lightweight Protocol π_2 to compute the DHKey.
- Phase 4 and Phase 5 are the same as those in Bluetooth 5.0.

Compared with the Secure Simple Pairing in Bluetooth 5.0, the above procedures possess not only leakage-resilient feature but also better performance. In addition, the security concerns discussed in Section 7.2 is addressed by the authentication measures provided in Bluetooth 5.0.

9.4 Comparison

We realize prototypes of the Secure Simple Paring protocol in Bluetooth 5.0, its leakage-resilient version in Section 9.2 and light-weighting version in Section 9.3. The software and hardware environments are the same as Tables 2 and 3 in Section 8.2.1.

Performance of the three protocols are compared in Figs. 2, 3 and Table 4. Fig. 2 compares the size of compile files. This evaluates the storage requirements of each protocol. In Fig. 3, the average runtime of each protocol with P-256 are compared. The leakage-resilient protocol in Section 9.2 has the largest overall runtime. The increase of runtime is reasonable since this protocol has the strongest security. The lightweight protocol in Section 9.3 has similar overall runtime as the original protocol in Bluetooth 5.0. Both can be good alternative AKE protocols in future versions of Bluetooth. A more comprehensive comparison is summarized in Table 4.

10 Conclusion

In this paper we presented an LRAKE protocol that is proved secure under the CAFLR-eCK model. To improve its performance, particularly in the edge environments where limited devices are wide deployed, a lightweight construction was presented to shift some computations from the limited party to its more powerful communicating partner. The proposed protocols will help to enhance the security for Edge AI.

To evaluate the performance and study the usability of the proposed protocols, prototypes were realized and a set of experiments were carried out. Implementation details were also presented. Moreover, a use case for Bluetooth 5.0 was illustrated. The theoretical design and implementation details will provide a guidance to future applications.

The Dziembowski-Faust leakage-resilient storage and refereshing method used in this paper is a bit complicated, though it can achieve high security level. In our future work, we plan to study leakage-resilient AKA protocols constructed by other methods.

Acknowledgments

This work was supported by the National Natural Science 1017 Foundation of China under Grant 61672289, Key Program Special Fund in XJTLU under Grant KSF-E-54, Jiangsu Planned 1018 Projects for Postdoctoral Research Funds under Grant 1019 2018K005A, National Natural Science Foundation of China 1020 under Grant 61701418, and Innovation Projects of The Next 1021 Generation Internet Technology under Grant NGII20170301.

References

[1] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[2] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 546–556, Sep./Oct. 2015.

[3] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[4] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," 2016, *arXiv:1611.03814*.

[5] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *Proc. IEEE Symp. Security Privacy*, 2017, pp. 3–18.

[6] S. Yeom, I. Giacomelli, M. Fredrikson, and S. Jha, "Privacy risk in machine learning: Analyzing the connection to overfitting," in *Proc. IEEE 31st Comput. Security Found. Symp.*, 2018, pp. 268–282.

[7] G. D. L. Torre, P. Rad, and K. K. R. Choo, "Driverless vehicle security: Challenges and future research opportunities," *Future Gener. Comput. Syst.*, 2018.

[8] J. Zeng, C. Li, and L. J. Zhang, "A face recognition system based on cloud computing and AI edge for IoT," in *Proc. Int. Conf. Edge Comput.*, 2018, pp. 91–98.

[9] G. Zhu, D. Liu, Y. Du, C. You, J. Zhang, and K. Huang, "Towards an intelligent edge: Wireless communication meets machine learning," 2018, *arXiv: 1809.00343*.

[10] W. Zhang, Z. Zhang, S. Zeadally, H. C. Chao, and V. Leung, "MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4216–4224, Jul. 2019.

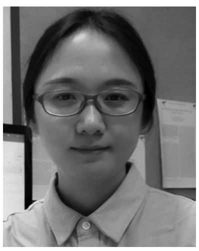
[11] Y. Shin, H. C. Kim, D. Kwon, J. H. Jeong, and J. Hur, "Unveiling hardware-based data prefetcher, a hidden source of information leakage," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 131–145.

[12] A. I. Mohammad and S. Ren, "Ohm's law in data centers: A voltage side channel for timing power attacks," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 146–162.

[13] G. Camurati, S. Poeplau, M. Muench, T. Hayes, and A. Francillon, "Screaming channels: When electromagnetic side channels meet radio transceivers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 163–177.

IEEE Proof

[14] L. J. V. Bulck, F. Piessens, and R. Strackx, "Nemesis: Studying microarchitectural timing leaks in rudimentary CPU interrupt," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 178–195.
[15] H. Naghibijouybari, A. Neupane, Z. Qian, and N. Abu-Ghazaleh, "Rendered insecure: GPU side channel attacks are practical," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 2139–2153.
[16] G. Maisuradze and C. Rossow, "ret2spec: Speculative execution using return stack buffers," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2018, pp. 2109–2122.
[17] P. Frigo, C. Giuffrida, H. Bos, and K. Razavi, "Grand pwning unit: Accelerating microarchitectural attacks with the GPU," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 357–372.
[18] J. Monaco, "Keylogging side channels," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 420–437.
[19] M. Zhao and G. E. Suh, "FPGA-based remote power side-channel attacks," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 229–244.
[20] D. Gruss *et al.*, "Another flip in the wall of rowhammer defenses," in *Proc. IEEE Symp. Security Privacy*, 2018, pp. 245–261.
[21] S. V. Schaik, C. Giuffrida, H. Bos, and K. Razavi, "Malicious management unit: Why stopping cache attacks in software is harder than you think," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 937–954.
[22] B. Gras, K. Razavi, H. Bos, and C. Giuffrida, "Translation leak-aside buffer: Defeating cache side-channel protections with TLB attacks," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 955–972.
[23] M. Lipp *et al.*, "Meltdown: Reading kernel memory from user space," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 973–990.
[24] V. Bulck *et al.*, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *Proc. 27th USENIX Security Symp.*, 2018, pp. 991–1008.
[25] O. Bronchain, J. M. Hendrickx, C. Massart, A. Olshevsky, and F. X. Standaert, "Leakage certification revisited: Bounding model errors in side-channel security evaluations," in *Proc. Annu. Int. Cryptology Conf.*, 2019, pp. 713–737.
[26] G. Cherubin, K. Chatzikokolakis, and C. Palamidessi, "F-BLEAU: Fast black-box leakage estimation," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 835–852.
[27] D. Genkin, M. Pattani, R. Schuster, and E. Tromer, "Synesthesia: Detecting screen content via remote acoustic side channels," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 853–869.
[28] A. C. Aldaya, B. B. Brumley, S. ul Hassan, C. P. Garcia, and N. Tuveri, "Port contention for fun and profit," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 870–887.
[29] M. Yan, R. Sprabery, B. Gopireddy, C. Fletcher, R. Campbell, and J. Torrellas, "Attack directories, not caches: Side channel attacks in a non-inclusive world," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 888–904.
[30] A. Kwong, W. Xu, and K. Fu, "Hard drive of hearing: Disks that eavesdrop with a synthesized microphone," in *Proc. IEEE Symp. Security Privacy*, 2019, pp. 905–919.
[31] C. Canella *et al.*, "Fallout: Leaking data on meltdown-resistant CPUs," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 769–784.
[32] K. Ryan, "Hardware-backed heist: Extracting ECDSA keys from qualcomm's TrustZone," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 181–194.
[33] A. Bhattacharyya *et al.*, "SMoTherSpectre: Exploiting speculative execution through port contention," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 785–800.
[34] P. C. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *Proc. Annu. Int. Cryptol Conf.*, 1996, pp. 104–113.
[35] T. Messerges, E. Dabbish, and R. Sloan, "Examining smart-card security under the threat of power analysis attacks," *IEEE Trans. Comput.*, vol. 51, no. 5, pp. 541–552, May 2002.
[36] M. Hutter, S. Mangard, and M. Feldhofer, "Power and EM attacks on passive 13.56 MHz RFID devices," in *Proc. Int. Workshop Cryptographic Hardware Embedded Syst.*, 2007, pp. 320–333.
[37] S. H. Shin, K. Kobara, and H. Imai, "Leakage-resilient authenticated key establishment protocols," in *Proc. Int. Conf. Theory Appl. Cryptol Inf. Security*, 2003, pp. 155–172.
[38] O. Ruan, M. Zhang, and J. Chen, "Leakage-resilient password-based authenticated key exchange," in *Proc. Int. Conf. Algorithms Archit Parallel Process.*, 2017, pp. 285–296.
[39] O. Ruan, J. Chen, and M. Zhang, "Provably leakage-resilient password-based authenticated key exchange in the standard model," *IEEE Access*, vol. 5, pp. 26832–26841, 2017.
[40] O. Ruan, Y. Zhang, M. Zhang, J. Zhou, and L. Harn, "After-the-fact leakage-resilient identity-based authenticated key exchange," *IEEE Syst. J.*, vol. 12, no. 2, pp. 2017–2026, Jun. 2018.
[41] R. Chen, Y. Mu, G. Yang, W. Susilo, and F. Guo, "Strongly leakage-resilient authenticated key exchange," in *Proc. Cryptographers' Track RSA Conf.*, 2016, pp. 19–36.
[42] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.
[43] Internet Engineering Task Force (IETF), "The Transport Layer Security (TLS) protocol version 1.3," 2018.
[44] Bluetooth SIG proprietary, "Bluetooth core specification version 5.0," 2016. [Online]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/
[45] IEEE computer society, "IEEE Standard 802.15.6: Wireless body area networks," 2012. [Online]. Available: https://standards.ieee.org/standard/802_15_6-2012.html
[46] B. LaMacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *Proc. Int. Conf. Provable Security*, 2007, pp. 1–16.
[47] D. Moriyama and T. Okamoto, "Leakage resilient eCK-secure key exchange protocol without random oracles," in *Proc. 6th ACM Symp. Inf. Comput. Commun. Security*, 2011, pp. 441–447.
[48] J. Alawatugoda, "On the leakage-resilient key exchange," *J. Math. Cryptol*, vol. 11, pp. 215–269, 2020.
[49] J. Alawatugoda, D. Stebila, and C. Boyd, "Modelling after-the-fact leakage for key exchange," in *Proc. 9th ACM Symp. Inf. Comput. Commun. Security*, 2014, pp. 207–216.
[50] J. Alawatugoda, C. Boyd, and D. Stebila, "Continuous after-the-fact leakage-resilient key exchange," in *Proc. Australas. Conf. Inf. Security Privacy*, 2014, pp. 258–273.
[51] J. Alawatugoda, C. Boyd, and D. Stebila, "Continuous after-the-fact leakage-resilient eCK-secure key exchange," in *Proc. IMA Int. Conf. Cryptography Coding*, 2015, pp. 277–294.
[52] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. Berlin, Germany: Springer, 2006.
[53] X. Chen, F. Zhang, W. Susilo, H. Tian, J. Li, and K. Kim, "Identity-based chameleon hashing and signatures without key exposure," *Inf. Sci.*, vol. 265, pp. 198–210, 2014.
[54] S. Dziembowski and S. Faust, "Leakage-resilient cryptography from the inner-product extractor," in *Proc. Int. Conf. Theory Appl. Cryptol Inf. Security*, 2011, pp. 702–721.
[55] Y. Yacobi and Z. Shmuely, "On key distribution systems," in *Proc. Conf. Theory Appl. Cryptol*, 1989, pp. 344–355.
[56] J. Zhang, X. Huang, P. Craig, A. Marshall, and D. Liu, "An improved protocol for the password authenticated association of IEEE 802.15. 6 standard that alleviates computational burden on the node," *Symmetry*, vol. 8, no. 11, 2016, Art. no. 131.
[57] J. Zhang, N. Xue, and X. Huang, "A secure system for pervasive social network-based healthcare," *IEEE Access*, vol. 4, pp. 9239–9250, 2016.
[58] J. Zhang, X. Huang, W. Wang, and Y. Yue, "Unbalancing pairing-free identity-based authenticated key exchange protocols," *IEEE Internet Things J.*, vol. 6, no. 1, pp. 878–890, Feb. 2019.
[59] National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS), "Recommended elliptic curves for federal government use," 1999. [Online]. Available: http://csrc.nist.gov/encryption/
[60] National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS), "Digital Signature Standard (DSS)," 2013. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf



Jie Zhang received the MS degree from Nanjing Normal University, Nanjing, China, in 2013, and the PhD degree from the University of Liverpool, Liverpool, United Kingdom, in 2018. Her current research interests include public key cryptography, key management, and security in the Internet of Things.

IEEE Proof

Futai Zhang received the bachelor's and master's degrees in mathematics from Shaanxi Normal University, China, and the PhD degrees from Xidian University, China. His main research interests include cryptography and cyberspace security.

Xin Huang received the PhD degree from the University of Oxford, Oxford, United Kingdom. He is currently currently a professor with the Taiyuan University of Technology, China; he is also working as a postdoctoral researcher with Soochow University and Suzhou L.J.T Intelligent Technology Co., Ltd., Suzhou, China. His research interests include network security, Internet of Things, and blockchain.

Xin Liu is working toward the master's degree in the School of Computer Science and Technology, Nanjing Normal University, Nanjing, China. Her research interests include key management and threshold signature.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/csdl.

IEEE Proof