LLRA: A Lightweight Leakage-Resilient Authentication Key Exchange Scheme for Smart Meters

Ran Cheng, Yukun Yang, Zhengzhuo Zhang, Xiaoxin Sun, Xin Huang*, Xiaohua Wu, and Liangbin Zhao

***Abstract*—The authentication between smart meters and Neighbourhood Area Communication Network (NAN) gateways is a critical issue due to various types of side channel attacks. To address this, a new protocol called Lightweight Leakage-Resilient Authentication Key Exchange (LLRA) was proposed for smart meters. LLRA incorporates continuous after-the-fact leakage-resilient property. Furthermore, extended BPR (CLR-eBPR) security model and extended GNY (CLR-eGNY) logic were introduced. Through provable security analysis, logic analysis, and practical performance evaluations, we demonstrated that LLRA effectively mitigates side-channel attacks while maintaining minimum authentication delay and energy consumption for the computing devices used in smart meters and NAN gateways. Compared to benchmark protocols, the authentication delay of LLRA was decreased by at least 14%; in addition, the energy consumption of LLRA was decreased by at least 7%.**

***Index Terms*—Leakage-resilient, Side-channel attacks, Authentication key exchange**

I. Introduction

SMART grids have been introduced as a solution to provide electricity efficiently and reliably, facilitating sustainable development. Smart meters are one of the most crucial components in smart grids [1]. Smart meters are digital devices that manage and control energy consumption. However, ensuring the security of information transmission between smart meters and Neighbourhood Area Communication Network (NAN) gateways is a significant concern [2], [3].

In order to ensure secure information transmission, smart meters need to use authentication key exchange (AKE) protocols. A great number of AKE protocols [3]–[8] were proposed for smart meters. However, side-channel attacks [9]–[15] against smart meters pose a challenge to the security of these AKE protocols. For example, an attacker can extract long-term private information from the non-volatile memory of a smart meter through side channel attacks [16]. So far several leakage-resilient AKE (LRAKE) protocols [17]–[20] have been proposed to resist side-channel attacks. However, these LRAKE protocols have high authentication delay and energy consumption when applied to smart meters. Lightweight AKE protocols that are specifically designed to mitigate side-channel attacks against smart meters have not been well studied.

Ran Cheng, Yukun Yang, Zhengzhuo Zhang, Xiaoxin Sun, and Xin Huang are with the College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology, Taiyuan 030024, China.

Xiaohua Wu, and Liangbin Zhao are with energy management department, Chinatowercom Co., Shanxi Branch, Taiyuan 030024, China.

In this paper, we propose a lightweight leakage-resilient authentication key exchange scheme (LLRA) for smart meters. This scheme enhances the security of smart meters, and guarantees security even when there is an arbitrarily large and continuous leakage of the long-term secret key. The main contributions are summarized as follows.

- First, we propose a new scheme called Lightweight Leakage-Resilient Authentication Key Exchange (LLRA) for smart meters. It can resist side-channel attacks on smart meters and also provide ID privacy. Furthermore, based on the comparison results in the use case, it has the shortest authentication delay and energy consumption compared to benchmarks. When CPU frequencies of both parties are equal, the authentication delay of LLRA was decreased by at least 14%, and the energy consumption of LLRA was decreased by at least 7%.
- Second, we create three additional rules based on jurisdiction rules of GNY logic. The new jurisdiction rules add leakage-resilient property. We used extended GNY (CLR-eGNY) logic to analyze the freshness, recognizability and possession of LLRA.
- Third, we also create extended BPR (CLR-eBPR) security model based on BPR security model. $Send$ query of adversary was modified to incorporate the adversary's ability to leak private information and messages caused by side-channel attacks. We used CLR-eBPR security model to analyze LLRA-secure.
- Fourth, LLRA has a high level of security. The provable security analysis, logic analysis and automatic formal verification using ProVerif confirm that LLRA successfully meets security requirements.
- Fifth, performance of LLRA has been evaluated in Section VI. The experimental results show that LLRA reduces the running time and computational time compared to benchmarks. When CPU frequencies of both parties are equal, the running time of LLRA was decreased by at least 14%, and the total computational time of LLRA was decreased by at least 22%.

The remaining parts of this paper are organized as follows: Section II presents an overview of related work; Section III introduces some mathematical foundations of this paper, system architecture and leakage-resilient storage scheme; Section IV presents the security problem of Zhang *et al.*'s protocol and detailed design of our scheme; Section V and VI present the security analysis and performance evaluation of the proposed

new scheme respectively; Section VII presents the practical application of the proposed mechanism. Finally, the conclusion and future work are presented in Section VIII.

II. Related Work

This section reviews the relevant research on authentication schemes for smart meters, leakage-resilient AKE protocols, and security analysis of leakage-resilient property.

A. AKE Protocols

In recent years, numerous authentication key exchange (AKE) schemes were proposed for smart meters. Mahmood *et al.* [4] presented a lightweight authentication scheme that utilizes the public key infrastructure (PKI). Srinivas *et al.* [5] proposed an anonymous user-authenticated key agreement protocol that restricts access solely to authorize users. In addition, Khan *et al.* [6] designed Password-Based Anonymous Lightweight Key Agreement Framework (PALK), while Chaudhry *et al.* [7] addressed its vulnerabilities through a new scheme (iPALE). Lian *et al.* [21] proposed a secure symmetric password authentication key agreement protocol. Chaudhry *et al.* [8] proposed a lightweight authentication scheme (LAS-SG) using elliptic curve cryptography (ECC) and symmetric key encryption. In addition, Chai *et al.* [3] developed a user-authenticated and key agreement scheme suitable for devices with limited computing capabilities. However, most AKE schemes do not consider side-channel attacks.

B. Leakage-Resilient AKE and PAKE protocols

Alawatugoda *et al.* presented leakage-resilient authentication key exchange (LRAKE) protocol [22], which required refreshing long-term secret keys before each session using the inner-product extractor method. However, this method had a heavy computational load. To solve this performance problem, Wu *et al.* [23] used the multiplicative blinding technique to divide each long-term secret key into two components and refresh them before each session. Ruan *et al.* [19] proposed leakage-resilient PAKE protocol that split the password into two parts. Zhang *et al.* [18] pointed out that [19] did not consider the security and communication standards of ECDH-based schemes, and proposed ECDH-based lightweight LRAKE protocol that was designed to be suitable for computationally limited devices. However, lightweight LRAKE [18] could not resist spoofing attack and could not provide mutual authentication. In 2021, Peng *et al.* [24] introduced an efficient LRAKE-IoT protocol suitable for resources-limited clients. Both protocol proposed by Wu *et al.* [23] and LRAKE-IoT [24] used bilinear pairing, which were inefficient.

C. Security Analysis of Leakage-Resilient Property

Moriyama and Okamoto [25] firstly proposed LR-eCK model to analyze leakage-resilient security property. Alawatugoda *et al.* [17] presented the Generic After-the-fact Leakage-eCK ((·)AFL-eCK) model. These models allow adversaries to retrieve portions of long-term and/or short-term secret keys while ensuring that the short-term key remains uncompromised. Alawatugoda *et al.*'s CLR-eCK model [26] permits adversaries to obtain partial content of long-term/short-term secret keys even after a session key is established. These models are based on extended Canetti-Krawczyk (eCK) security model; however, how Bellare-Pointcheval-Rogaway (BPR) security model can be extended to analyze leakage-resilient security property of AKE protocols has not been discussed.

In addition to provable security analysis mentioned above, logic is also a widely used security analysis method. Burrows-Abdi-Needham (BAN) [27] and Gong-Needham-Yahalom (GNY) [28] are two fundamental ones. However, they did not take side-channel attacks into account.

Table I: Notations and Descriptions.

Notation	Description
$E(\mathbb{F}_p)$	Elliptic curve group over finite field $\mathbb{F}_p$
P	A base point of $E(\mathbb{F}_p)$
q	The order of G
$\mathbb{Z}_q^*$	Prime finite field
k	Security parameter
U_A	Smart meter
S_B	NAN gateway
ID_U	Identity of participant U, $ID_U \in \{0,1\}^k$, $U \in \{A, B\}$
pw_U	Password of U, $U \in \{A, B\}$
y_A, y_B	Ephemeral private keys
r_A, r_B	Long-term private keys
SK_A, SK_B	Session key
msg_j	The jth message in the protocol
T_A, T_B	Timestamp
S_A, S_B	Salt
$Enc_k(\cdot)$	Symmetric encryption algorithm using key k
$Dec_k(\cdot)$	Symmetric decryption algorithm using key k
$H(\cdot)$	Hash function, $H : \{0,1\}^* \rightarrow \{0,1\}^{p(k)}$
$H_1(\cdot)$	Hash function, $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^*$
$H_2(\cdot)$	$\bigoplus$-linear hash function, $H_2 : \mathbb{Z}_q^* \rightarrow \{0,1\}^k$, $H_2(M \oplus M') = h(M) \oplus h(M')$
$H_3(\cdot)$	Hash function, $H_3 : E(\mathbb{F}_p) \rightarrow \mathbb{Z}_q^*$
$\oplus$	Bitwise XOR operation

D. Summary

Lightweight AKE protocols especially designed to mitigate side-channel attacks against smart meters have not been well studied. In addition, how BPR security model can be extended to analyze leakage-resilient security property of PAKE protocols has not been discussed. Furthermore, how logic can be extended to analyze leakage-resilient AKE protocols also has not been discussed.

III. Preliminary

This section introduces foundations for LLRA. Notations are explained in Table I.

A. Mathematical Foundations

1) Elliptic Curve Group and Hard Problems

Let $F(p)$ denotes an elliptic curve group containing all points (x, y) defined by the following equation plus the point at infinite [29]: $y^2 = x^3 + ax + b \bmod p$, with $a, b \in F(p)$ and $4a^3 + 27b^2 \neq 0$. The following hard problems are assumed to be hard over G of order n with a generator P [30].

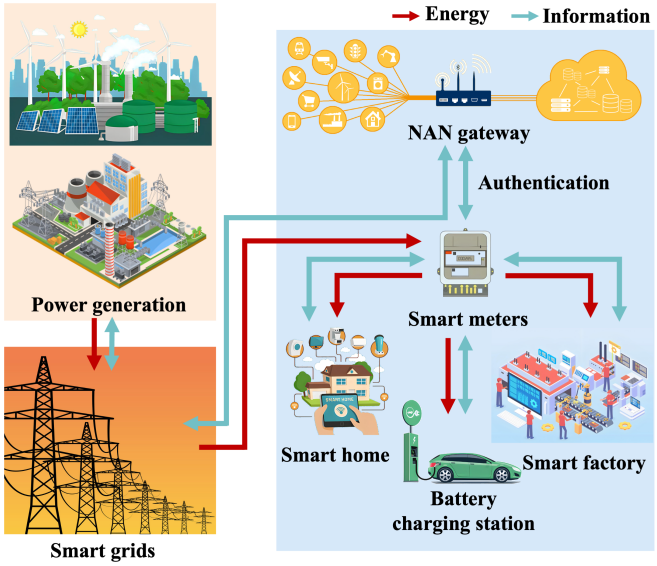


Figure 1: System Architecture.

Definition 1 (Elliptic Curve Discrete Logarithm Problem (ECDLP))**.** *Given* $(P, G, x{\cdot}P)$*, output* x*, where* $x \in \mathbb{Z}_q^*, x{\cdot}P \in G$*.*

Definition 2 (Elliptic Curve Computational Diffie-Hellman Problem (ECCDHP))**.** *Given* $(P, G, x \cdot P, y \cdot P)$*, output* $xy \cdot P$*, where* $x, y \in \mathbb{Z}_q^*$ *and* $x \cdot P, y \cdot P \in G$*.*

B. System Architecture and Side-Channel Attacks

The typical architecture is shown in Figure 1. Smart meters are deployed in residential and industrial areas (e.g., factories and homes). Data is transmitted from the smart meters to NAN gateways and subsequently to the main service provider. Trust establishment between smart meters and NAN gateways is crucial throughout this process. According to [9]–[15], smart meters are vulnerable to side-channel attacks.

- Microarchitectural side-channel attacks [10]: Attacks on the microarchitecture of smart meters become a practical threat to privacy. An adversary may deploy a malicious program on smart meters, yet lacks control over the scheduling of the attacking process.
- Physical side-channel attacks [11]: The attacker must have physical access to smart meters in order to collect physical signals (e.g., power consumption) during execution and spy the behaviors of smart meters.
- Network side-channel attacks [12], [13]: An attacker can use network application characteristics (e.g, response information, packet patterns and sizes) as side channels to attack network services protected by encryption protocols used by smart meters.
- Transient execution attacks [14]: Side-channel techniques are employed as a means to extract confidential information. The attacker focuses on accessing all data within the protected memory areas of smart meters.
- Invasive attacks [15]: These active attacks have the potential to compromise cryptographic ciphers and disrupt the data stored on smart meters.

C. Leakage-Resilient Storage

This part reviews the definitions of leakage-resilient storage according to Dziembowski *et al.* [31].

Definition 3 (Leakage-resilient storage scheme)**.** *The storage scheme* $\Lambda^{n,m}_{\mathbb{Z}_q^*} = (Encode^{n,m}_{\mathbb{Z}_q^*}, Decode^{n,m}_{\mathbb{Z}_q^*})$ *stores elements* $r \in \left(\mathbb{Z}_q^*\right)^m$ *for any* $m, n \in \mathbb{N}$*:*

- $Encode^{n,m}_{\mathbb{Z}_q^*}(r) : r_L \xleftarrow{\$} \left(\mathbb{Z}_q^*\right)^n \setminus \{(0^n)\}$, $r_R \leftarrow \left(\mathbb{Z}_q^*\right)^{n\times m}$, $r_L \cdot r_R = r$ *and outputs* (r_L, r_R)
- $Decode^{n,m}_{\mathbb{Z}_q^*}(r_L, r_R)$ *: outputs* r*, where* $r = r_L \cdot r_R$

Definition 4 (Refreshing protocol of leakage-resilient storage)**.** *Let* $Refresh^{n,m}_{\mathbb{Z}_q^*}(r_L, r_R)$ *denotes a refreshing protocol:*

- *Input:* (r_L, r_R), $r_L \in \left(\mathbb{Z}_q^*\right)^n$, $r_R \in \left(\mathbb{Z}_q^*\right)^{n\times m}$
- *Output:* (r'_L, r'_R)

Where $Decode^{n,m}_{\mathbb{Z}_q^*}(r_L, r_R) = Decode^{n,m}_{\mathbb{Z}_q^*}(r'_L, r'_R)$

Definition 5 ($(\boldsymbol{\lambda}_\Lambda, \epsilon)$-secure leakage-resilient storage scheme)**.** *Assuming that* $m < n/20$*, we define* $\Lambda^{n,m}_{Z_q^*} = (Encode^{n,m}_{\mathbb{Z}_q^*}, Decode^{n,m}_{\mathbb{Z}_q^*})$ *as being* $(2\boldsymbol{\lambda}_\Lambda, \epsilon)$*-secure leakage-resilient, the leakage from* $Encode^{n,m}_{\mathbb{Z}_q^*}(r_A)$ *and* $Encode^{n,m}_{\mathbb{Z}_q^*}(r_B)$ *is statistically* ϵ*-close. And* $\boldsymbol{\lambda}_\Lambda = (0.15 \cdot n \log q, 0.15 \cdot n \log q)$*.*

Definition 6 ($(l, \boldsymbol{\lambda}_\Lambda, \epsilon')$-secure leakage-resilience of a refreshing protocol)**.** *Given that* $n \geq m/3$*,* $n \geq 16$*,* $l \in N$*, for a* $(2\boldsymbol{\lambda}_\Lambda, \epsilon)$*-secure leakage-resilient storage scheme* $\Lambda^{n,m}_{Z_q^*} = (Encode^{n,m}_{\mathbb{Z}_q^*}, Decode^{n,m}_{\mathbb{Z}_q^*})$*, a refresh protocol* $Refresh^{n,m}_{\mathbb{Z}_q^*}(r_L, r_R)$ *is* $(l, \boldsymbol{\lambda}_\Lambda, \epsilon')$*-secure leakage-resilient, the leakages from* $Refresh^{n,m}_{\mathbb{Z}_q^*}(r_{AL}, r_{AR})$ *and* $Refresh^{n,m}_{\mathbb{Z}_q^*}(r_{BL}, r_{BR})$ *are statistically* ϵ'*-close. And* $\boldsymbol{\lambda}_\Lambda = (0.15 \cdot n \log q, 0.15 \cdot n \log q)$*.*

D. Threat Model

According to Figure 1, we assume that the attacker can eavesdrop and intercept messages transmitted in the bidirectional wireless communication channel between the smart meter and the NAN gateway. The attacker can also replay messages recorded from previous sessions; inject forged messages; and spoof identities and important parameters. Especially, the attacker can exploit side-channel attacks to access sensitive information in smart meters and the NAN gateway.

E. Security Goals

Security requirements suggested by the National Institute Standards Technology (NIST) [32] are summarised as follow:

- *Authentication and session key security:* The proposed protocol allows only registered smart meters and NAN gateways to authenticate each other. A session key should be established between the smart meter and the NAN gateway at the end of the proposed protocol.
- *Confidentiality:* Sensitive information exchanged through public channels must not be disclosed to illegal parties.
- *Message integrity:* Upon receiving a message, the legal participant can verify whether the message has been tampered with or not.

This article has been accepted for publication in IEEE Transactions on Smart Grid. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TSG.2024.3435548

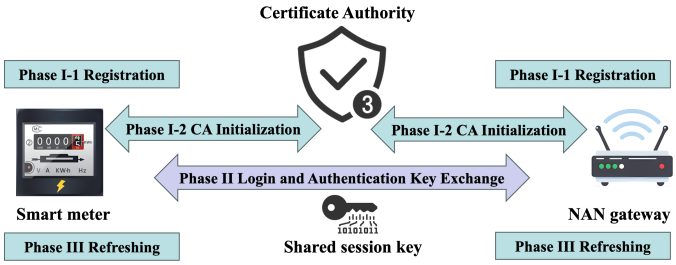


Figure 2: General Overview of LLRA Protocol.

A	B
Initialization:	
runs $Encode_{Z_q^*}^{n,1}$ to encode sk_A	runs $Encode_{Z_q^*}^{n,1}$ to encode sk_B
$\overrightarrow{sk_{AL}} = (sk_{AL_1}, \ldots, sk_{AL_n})$	$\overrightarrow{sk_{BL}} = (sk_{BL_1}, \ldots, sk_{BL_n})$
$\overrightarrow{sk_{AR}} = (sk_{AR_1}, \ldots, sk_{AR_n})$	$\overrightarrow{sk_{BR}} = (sk_{BR_1}, \ldots, sk_{BR_n})$
stores $\overrightarrow{sk_{AL}}$ and $\overrightarrow{sk_{AR}}$.	stores $\overrightarrow{sk_{BL}}$ and $\overrightarrow{sk_{BR}}$.
destroys sk_A.	destroys sk_B.
Key Exchange:	
Chooses a random value $r_A \leftarrow \mathbb{Z}_q^*$	Chooses a random value $r_B \leftarrow \mathbb{Z}_q^*$
Computes	Computes
$\overrightarrow{v_A} = (r_A, sk_{AL_1}, \ldots, sk_{AL_n})$	$\overrightarrow{v_B} = (r_B, sk_{BL_1}, \ldots, sk_{BL_n})$
$\overrightarrow{w_A} = (1, sk_{AR_1}, \ldots, sk_{AR_n})$	$\overrightarrow{w_B} = (1, sk_{BR_1}, \ldots, sk_{BR_n})$
Computes	Computes
$u_A = \overrightarrow{v_A} \cdot \overrightarrow{w_A}^\top$	$u_B = \overrightarrow{v_B} \cdot \overrightarrow{w_B}^\top$
$\xrightarrow{msg_1=\{u_A\}}$	$U_B = u_B \cdot G.$
	$\xleftarrow{msg_2=\{U_B\}}$
Computes	Computes
$K_{temp} = r_A \cdot (U_B - PK_B)$	$U_A = u_A \cdot G$
$k = H(K_{temp})$	$K_{temp} = r_B \cdot (U_A - PK_A)$
	$k = H(K_{temp})$

Figure 3: Zhang *et al.*'s Protocol [18].

- *Perfect forward secrecy:* The information transmitted in previous runs should not impact the security of subsequent protocol executions.
- *Anonymity:* During the authentication process, it is impossible for attackers to uncover identities of smart meters.
- *Leakage-resilient security:* The protocol should effectively resist side-channel attacks.
- *Resilience against common attacks:* The system should avoid replay attacks, impersonation attacks, man-in-the-middle attacks, Denial of Service (DoS) attacks, key compromise impersonation attacks, and off-line password guessing attacks.

IV. Proposed Scheme

In this section, we analyze the security problem of Zhang *et al.*'s protocol Lightweight LRAKE [18] and introduce our leakage-resilient AKE (LLRA) protocol. Figure 2 is an overview of LLRA protocol.

A. Review of Lightweight LRAKE

This part introduces security problems of Zhang *et al.*'s protocol Lightweight LRAKE [18] that is shown in Figure 3.

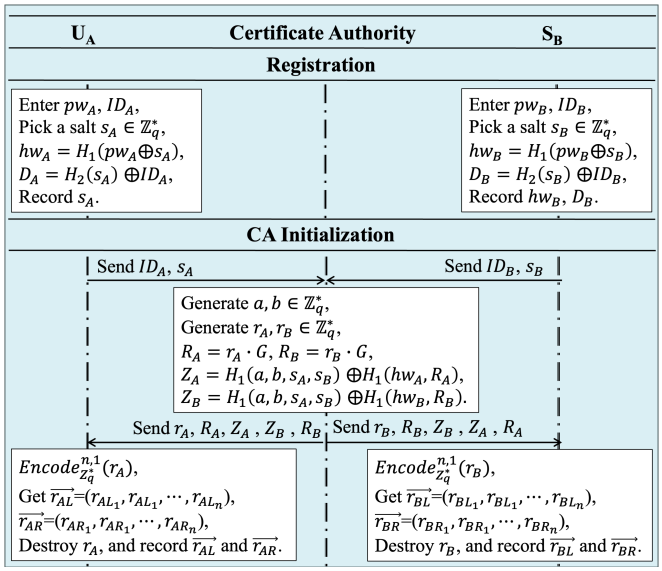


Figure 4: Initialization of LLRA.

Lightweight LRAKE was an ECC-based authentication key exchange scheme proposed to resist side-channel attacks. However, Lightweight LRAKE suffers from impersonation attack. For example, an adversary $\mathcal{E}$ can impersonate B and establish a shared session key with A. $\mathcal{E}$ intercepts the message from B to A, and replaces U_B with $U_E = r_E \cdot G + PK_B$ for some random value r_E generated by $\mathcal{E}$. At the end of key exchange, A computes the session key $k = H(K_{temp}) = H(r_A \cdot (U_\mathcal{E} - PK_B)) = H(r_A \cdot r_\mathcal{E} \cdot G)$, and $\mathcal{E}$ computes the same session key $k = H(K_{temp}) = H(r_\mathcal{E} \cdot (U_A - PK_A)) = H(r_A \cdot r_\mathcal{E} \cdot G)$. $\mathcal{E}$ can also initiate man-in-the-middle attack to Lightweight LRAKE. In addition, Lightweight LRAKE lacks of mutual authentication. Thus, although Lightweight LRAKE provides leakage-resilient security property and avoids side-channel attacks, it is essential to avoid these attacks when it is used in smart meters due to the security goals described in Section III-E.

B. LLRA Protocol

In this subsection, we present the details of our proposed scheme. Similar to Lightweight LRAKE, LLRA consists of three phases: initialization, login and authentication key exchange, and refreshing.

1) Initialization

During this phase, assuming messages are delivered to each other over a secure channel. As shown in Figure 4, the process of initialization is given as follow.

1. **Step 1: Registration**

U_A and S_B are allowed to take pre-computation.

- U_A *Registration*: U_A enters a password pw_A and an identity ID_A into the device U_A. U_A picks a salt $s_A \in \mathbb{Z}_q^*$ randomly to randomize pw_A and ID_A. U_A then computes $hw_A = H_1(pw_A \oplus s_A)$ and $D_A = H_2(s_A) \oplus ID_A$. Then, U_A records s_A.
- S_B *Registration*: S_B enters a password pw_B and an identity ID_B into the device S_B. S_B picks a salt $s_B \in \mathbb{Z}_q^*$

randomly to randomize pw_B and ID_B. S_B then computes $hw_B = H_1(pw_B \oplus s_B)$ and $D_B = H_2(s_B) \oplus ID_B$. Then, S_B records a password file which includes hw_B and D_B.

2. **Step 2: CA Initialization**

Certificate Authority (CA) selects random numbers $a, b \in \mathbb{Z}_q^*$. U_A sends ID_A and s_A to CA. CA verifies the uniqueness of ID_A and associates it with U_A. If (U_A, ID_A) already exists, U_A must repeat Step 1. Otherwise, CA generates a pair of public/private key for U_A: it selects a random number $r_A \in \mathbb{Z}_q^*$ and calculates $R_A = r_A \cdot G$, $Z_A = H_1(a, b, s_A, s_B) \oplus H_1(hw_A, R_A)$. Similarly, S_B registers with the service provider using its unique identifier ID_B and s_B through a secure channel. CA generates a pair of public-private key for S_B: it selects a random number $r_B \in \mathbb{Z}_q^*$ and calculates $R_B = r_B \cdot G$, $Z_B = H_1(a, b, s_A, s_B) \oplus H_1(hw_B, R_B)$. CA sends $\{r_A, R_A, Z_A, Z_B, R_B\}$ to U_A, and $\{r_B, R_B, Z_B, Z_A, R_A\}$ to S_B. U_A runs the encode algorithm $Encode_{Z_q^*}^{n,1}$ to encode r_A into two n-dimensional vectors $\overrightarrow{r_{AL}} = (r_{AL_1}, \ldots, r_{AL_n})$ and $\overrightarrow{r_{AR}} = (r_{AR_1}, \ldots, r_{AR_n})$. Then U_A stores $\overrightarrow{r_{AL}}$ and $\overrightarrow{r_{AR}}$ independently and destroys r_A. S_B runs the encode algorithm $Encode_{Z_q^*}^{n,1}$ to encode r_B into two n-dimensional vectors $\overrightarrow{r_{BL}} = (r_{BL_1}, \ldots, r_{BL_n})$ and $\overrightarrow{r_{BR}} = (r_{BR_1}, \ldots, r_{BR_n})$. Then S_B stores $\overrightarrow{r_{BL}}$ and $\overrightarrow{r_{BR}}$ independently and destroys r_B.

2) Login In and Authentication Key Exchange Phase

The protocol can achieve mutual authentication and generate a shared session key. An overview of our LLRA can be seen in Figure 5.

1. Login in: U_A enters ID_A and password pw_A, U_A computes $hw_A = H_1(pw_A \oplus s_A)$ and $D_A = H_2(s_A) \oplus ID_A$.
2. $(NewSession, T_A, S_B)$: U_A selects $y_A \leftarrow \mathbb{Z}_q^*$. It sets $\overrightarrow{v_A} = (y_A, r_{AL_1}, \ldots, r_{AL_n})$ and $\overrightarrow{w_A} = (1, r_{AR_1}, \ldots, r_{AR_n})$. It computes $u_A = \overrightarrow{v_A} \cdot \overrightarrow{w_A}^\top$, $X_A = H_2(hw_A \oplus D_A \oplus u_A \oplus y_A)$, $Y_A = (u_A + y_A) \cdot G$, $K_A = Z_A \oplus H_1(hw_A, R_A) \oplus X_A$, $Auth = Enc_{K_A}(y_A \oplus T_A)$ and $C_A = H(T_A, y_A, Auth)$. Then U_A sends $(X_A, Y_A, T_A, C_A, Auth)$ to S_B, where T_A denotes the timestamp.
3. $(NewSession, T_B, U_A)$: S_B selects $y_B \leftarrow \mathbb{Z}_q^*$. It sets $\overrightarrow{v_B} = (y_B, r_{BL_1}, \ldots, r_{BL_n})$ and $\overrightarrow{w_B} = (1, r_{BR_1}, \ldots, r_{BR_n})$. It computes $u_B = \overrightarrow{v_B} \cdot \overrightarrow{w_B}^\top$, $X_B = H_2(hw_B \oplus D_B \oplus u_B \oplus y_B)$ and $Y_B = (u_B + y_B) \cdot G$. After receiving $(X_A, Y_A, T_A, C_A, Auth)$ from U_A, S_B checks timestamp T_A, and aborts if not fresh. It then computes $K_A^* = Z_B \oplus H_1(hw_B, R_B) \oplus X_A$, $y_A^* = Dec_{K_A^*}(Auth) \oplus T_A$ and $C_A^* = H(T_A, y_A^*, Auth)$. And S_B checks $C_A^* \stackrel{?}{=} C_A$. If it fails, then aborts; else, it computes $V_B = (u_B + y_B) \cdot Y_A$, $sk_B = X_A \oplus X_B \oplus H_3(V_B)$ and $E_B = Enc_{sk_B}(T_B, hw_B, y_B, X_A, V_B, T_A)$. Finally, S_B sends (X_B, Y_B, T_B, E_B).
4. After receiving (X_B, Y_B, T_B, E_B) from S_B, U_A first checks if the timestamp T_B is fresh. Next, it computes $V_A = (u_A + y_A) \cdot Y_B$, $sk_A = X_A \oplus X_B \oplus H_3(V_A)$. Then, it obtains $(T_B, hw_B^*, y_B^*, X_A^*, V_B^*, T_A)$ by decrypting the ciphertext E_B. Then U_A verifies if T_B is valid, and checks $X_A^* \stackrel{?}{=} X_A$ and $V_B^* \stackrel{?}{=} V_A$. If it fails, then aborts; else, U_A computes $E_A = Enc_{sk_A}(T_B, hw_A, y_A, u_A, X_B, V_A)$. Then, it sends E_A to S_B. Finally, It sets $sid_A = (Z_A, Y_A, Z_B, Y_B)$ and computes session key $SK_A = H(sk_A, sid_A)$.
5. After receiving E_A from U_A, S_B obtains $(T_B, hw'_A, y'_A, u'_A, X'_B, V'_A)$ by decrypting the ciphertext E_A. It verifies if T_A is valid, and checks $X'_B \stackrel{?}{=} X_B$ and $V'_A \stackrel{?}{=} V_B$. If it fails, then aborts; else, it sets $sid_B = (Z_A, Y_A, Z_B, Y_B)$ and computes session key $SK_B = H(sk_B, sid_B)$.

U_A	S_B
Authentication Key Exchange:	
On input (Newsession, T_A, S_B)	On input (Newsession, T_B, U_A)
$y_A \leftarrow \mathbb{Z}_q^*$	$y_B \leftarrow \mathbb{Z}_q^*$
Set	Set
$\overrightarrow{v_A} = (y_A, r_{AL_1}, \ldots, r_{AL_n})$	$\overrightarrow{v_B} = (y_B, r_{BL_1}, \ldots, r_{BL_n})$
$\overrightarrow{w_A} = (1, r_{AR_1}, \ldots, r_{AR_n})$	$\overrightarrow{w_B} = (1, r_{BR_1}, \ldots, r_{BR_n})$
Compute	Compute
$u_A = \overrightarrow{v_A} \cdot \overrightarrow{w_A}^\top$	$u_B = \overrightarrow{v_B} \cdot \overrightarrow{w_B}^\top$
$X_A = H_2(hw_A \oplus D_A \oplus u_A \oplus y_A)$	$X_B = H_2(hw_B \oplus D_B \oplus u_B \oplus y_B)$
$Y_A = (u_A + y_A) \cdot G$	$Y_B = (u_B + y_B) \cdot G$
$K_A = Z_A \oplus H_1(hw_A, R_A) \oplus X_A$	
$Auth = Enc_{K_A}(y_A \oplus T_A)$	
$C_A = H(T_A, y_A, Auth)$	
$\xrightarrow{X_A, Y_A, T_A, C_A, Auth}$	
	If T_A is fresh, compute
	$K_A^* = Z_B \oplus H_1(hw_B, R_B) \oplus X_A$
	$y_A^* = Dec_{K_A^*}(Auth) \oplus T_A$
	$C_A^* = H(T_A, y_A^*, Auth)$
	Verify if $C_A^* = C_A$
	$V_B = (u_B + y_B) \cdot Y_A$
	$sk_B = X_A \oplus X_B \oplus H_3(V_B)$
	$E_B =$
	$Enc_{sk_B}(T_B, hw_B, y_B, X_A, V_B, T_A)$
	$\xleftarrow{X_B, Y_B, T_B, E_B}$
If T_B is fresh, compute	
$V_A = (u_A + y_A) \cdot Y_B$	
$sk_A = X_A \oplus X_B \oplus H_3(V_A)$	
$Dec_{sk_A}(E_B) =$	
$(T_B, hw_B^*, y_B^*, X_A^*, V_B^*, T_A)$	
If T_B is valid, verify if	
$X_A^* = X_A$ and $V_B^* = V_A$	
Compute	
$E_A =$	
$Enc_{sk_A}(T_B, hw_A, y_A, u_A, X_B, V_A)$	
$\xrightarrow{E_A}$	
$sid_A = (Z_A, Y_A, Z_B, Y_B)$	$Dec_{sk_B}(E_A) =$
$SK_A = H(sk_A, sid_A)$	$(T_B, hw'_A, y'_A, u'_A, X'_B, V'_A)$
	If T_A is valid, verify if
	$X'_B = X_B$ and $V'_A = V_B$
	Compute
	$sid_B = (Z_A, Y_A, Z_B, Y_B)$
	$SK_B = H(sk_B, sid_B)$

Figure 5: Authentication Key Exchange of LLRA Protocol.

Remark: U_A and S_B establish the same session key in LLRA, i.e. $sk_A = sk_B$.

$$V_A = (u_A + y_A) \cdot Y_B = Y_A \cdot (u_B + y_B) = V_B$$
$$= (u_B + y_B)(u_A + y_A) \cdot G$$

$$sk_A = X_A \oplus X_B \oplus H_3(V_A) = X_A \oplus X_B \oplus H_3(V_B) = sk_B$$

3) Refreshing

As defined in Definition 4, U_A runs the refresh protocol $Refresh_{Z_q^*}^{n,1}$ to refresh $(\overrightarrow{r_{AL}}, \overrightarrow{r_{AR}})$. Let $\left(\overrightarrow{r'_{AL}}, \overrightarrow{r'_{AR}}\right) \leftarrow Refresh_{Z_q^*}^{n,1}(\overrightarrow{r_{AL}}, \overrightarrow{r_{AR}})$. S_B also runs the refresh protocol $Refresh_{Z_q^*}^{n,1}$ to refresh $(\overrightarrow{r_{BL}}, \overrightarrow{r_{BR}})$.

V. Security Analysis

This section introduces the formal security analysis of LLRA using CLR-eBPR security model and CLR-eGNY logic. Also, LLRA is verified using ProVerif.

A. CLR-eBPR Security Model

In this subsection, we introduce an extended BPR (CLR-eBPR) security model based on BPR security model [33].

1) Partner

There are multiple instances of each principal P during the protocol execution. We denote the i-th instance of P as an oracle, represented by π_p^i. π_p^i should hold a particular session key (SK), session id (sid), and partner id (pid).

Definition 7 (Partner)**.** *If oracles π_s^i holding (SK_s, sid_s, pid_s) and π_n^j holding (SK_n, sid_n, pid_n) have both accepted sid_s and SK_s, and the following conditions hold:*

- *s acts as the initiator while n acts as the responder, or vice versa; $sid_s = sid_n$, $SK_s = SK_n$, $pid_s = n$ and $pid_n = s$,*
- *Except π_s^i and π_n^j, no other oracle in the security game accepts a session ID equal to sid_s.*

2) Adversarial Queries

The continuous leakage [22] is modeled using a tuple leakage function $f = (f_{Li}, f_{Ri})$ and a leakage parameter $r = (r_L, r_R)$. In the security game, a probabilistic polynomial time (PPT) adversary $\mathcal{A}$ interacts with the instance oracles by queries below:

- $Execute(\pi_s^i, \pi_n^j)$: This oracle returns the transcript of all messages sent over the network to $\mathcal{A}$.
- $Send(\pi_s^i, \pi_n^j, msg, f)$: This query returns the next message with the leakage boundary $f(r)$. $\left|f_{Li}\left(r_L^{\pi^i}\right)\right| \leq r_L$ and $\left|f_{Ri}\left(r_R^{\pi_s^i}\right)\right| \leq r_R$.
- $Reveal(\pi_s^i)$: This query returns a session key $\pi_s^i.sk$.
- $Corrupt(\pi_s^i)$: This query returns the long-term secrets of the instance π_s^i to $\mathcal{A}$.
- $Test(\pi_s^i)$: This query returns $null$ if π_s^i do not have session key freshness (The BPR security model gives the definition of session key freshness). Only a fresh instance oracle can be $Test$-queried, which avoids the adversary trivially win the security game. Otherwise, an unbiased coin b is tossed. This query returns $\pi_s^i.sk$ if $b = 0$ or a random value with the same length of sk if $b = 1$. If $\mathcal{A}$'s output b' is equal to b, then we say that $\mathcal{A}$ wins the security game.

3) Semantic Security

We define $\Pr[b' = b]$ as the probability of adversary $\mathcal{A}$ wins the security game. Thus, the advantage of $\mathcal{A}$ against protocol Π is

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{LLRA}(\lambda) = |2\Pr[b' = b] - 1|.$$

Definition 8 (LLRA-Secure)**.** *A protocol Π is LLRA-secure if there is a negligible function $\varepsilon(\lambda)$ such that for all PPT adversaries $\mathcal{A}$ in the LLRA model,*

$$\mathbf{Adv}_{\Pi,\mathcal{A}}^{LLRA}(\lambda) \leq \varepsilon(\lambda).$$

B. Formal Security Proof

We use CLR-eBPR security model defined in Section V-A to prove the security of LLRA.

Theorem 1. *Suppose there is a PPT adversary $\mathcal{A}$ against LLRA protocol. If $H_1(\cdot), H_2(\cdot)$ and $H_3(\cdot)$ are modeled as random oracles and the ECDLP and ECCDHP are hard under the subgroup generated by G on $E(\mathbb{F}_p)$, LLRA protocol is LLRA-secure.*

For any AKE adversary $\mathcal{A}$ against LLRA that runs in time at most t, involves at most n_p honest parties and activates at most k sessions, we show that there exists a ECCDH solver $\mathcal{S}$ and a ECDLP solver $\mathcal{T}$, such that

$$\begin{aligned}\mathbf{Adv}_{\mathcal{A}}^{LLRA}(k) &\leq 2n_p \cdot \epsilon' + \frac{(n_e + n_s)^2}{n_p} + \frac{n_h^2}{2^l} \\ &+ 2\left(\frac{n_s + n_h}{|X|} + \frac{n_e + n_h}{|X|}\right) + \frac{4(n_s - 1)}{|X|} \\ &+ 2(n_s + n_c)\mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T}) + 2n_c\mathbf{Adv}_{\mathcal{A}}^{ECCDH}(\mathcal{S})\end{aligned}$$

where $\mathcal{S}$ runs in time $O(tk)$ and $\mathcal{T}$ runs in time $O(t)$. n_s, n_c, n_e and n_h denote the number of $Send$, $Corrupt$, $Execute$, H hash oracles queried by $\mathcal{A}$, respectively. And l, $|X|$ denote the bit length of H's outputs and the cardinality of the dictionary X respectively.

Proof. We present the formal proof using a sequence of games from the real security game G_0 to the random game G_9. The games run in time at most t, and involve at most n_p honest parties. We also define an event $\xi_i (0 \leq i \leq 9)$ as $\mathcal{A}$ winning game G_i by breaching the semantic security of LLRA. Additionally, we assume $\mathcal{S}$ is the solver of ECCDHP and $\mathcal{T}$ is the solver of ECDLP.

Game G_0: This game corresponds to a real attack scenario. Based on the security definition, we obtain:

$$\mathbf{Adv}_{\mathcal{A}}^{LLRA}(k) = \left|2\Pr\left[b' = b\right] - 1\right|$$

Game G_1: This game simulates the hash oracle H. The execution of the $Reveal$, $Send$, $Corrupt$, and $Test$ queries in this game is equivalent to executing an actual attack. Thus, we have:

$$\text{Prob}\left[\xi_1\right] = \text{Prob}\left[\xi_0\right]$$

Game G_2: In this game, a collision occurs based on the birthday attack. In the content simulation, the probability of collisions is at most $\begin{pmatrix} n_e + n_s \\ 2 \end{pmatrix} \cdot \frac{1}{n_p}$. In the hash oracle

simulation, the probability of collisions is at most $\begin{pmatrix} n_h \\ 2 \end{pmatrix} \cdot \frac{1}{2^l}$. Thus, we have:

$$\begin{aligned} |\mathrm{Prob}\left[\xi_2\right] - \mathrm{Prob}\left[\xi_1\right]| &\leq \begin{pmatrix} n_e + n_s \\ 2 \end{pmatrix} \cdot \frac{1}{n_p} + \begin{pmatrix} n_h \\ 2 \end{pmatrix} \cdot \frac{1}{2^l} \\ &= \frac{(n_e + n_s)(n_e + n_s - 1)}{2n_p} + \frac{n_h(n_h - 1)}{2^{l+1}} \\ &\leq \frac{(n_e + n_s)^2}{2n_p} + \frac{n_h^2}{2^{l+1}} \end{aligned}$$

Game G_3**:** In this game, we analyze the leakage-resilience property in Definition 6. We set $m = 1$ and $n > 20$; this setting guarantees that $n > 20 \cdot m$, $n \geq m/3$ and $n > 16$. The leakage-resilient storage $\Lambda_{Z_q^*}^{n,1}$ is $(2\lambda, \epsilon)$-secure leakage-resilient and the refreshing protocol $Refresh_{Z_q^*}^{n,1}$ is (l, λ, ϵ'), $l \in N$, ϵ and ϵ' is negligible, and $\lambda = (0.15n \log q, 0.15n \log q)$. Thus, we have:

$$|\mathrm{Prob}\left[\xi_3\right] - \mathrm{Prob}\left[\xi_2\right]| \leq n_p \cdot \epsilon'$$

Game G_4**:** In this game, $\mathcal{A}$ replaces Y_A with the random value x_{Y_A}. In order to distinguish x_{Y_A} from Y_A, $\mathcal{A}$ queries n_s times with a probability of $\mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T})$. Then, we have:

$$|\mathrm{Prob}\left[\xi_4\right] - \mathrm{Prob}\left[\xi_3\right]| \leq n_s \mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T})$$

Game G_5**:** In this game, the ciphertext E_B is replaced with the encryption output of hw_B'. If $\mathcal{A}$ can distinguish game G_5 from game G_4, then the semantic security of authenticated encryption will be compromised in at most $n_e + n_h$ instances, each with a probability of $\frac{1}{|X|}$. Thus, we have:

$$|\mathrm{Prob}\left[\xi_5\right] - \mathrm{Prob}\left[\xi_4\right]| \leq \frac{n_e + n_h}{|X|}$$

Game G_6**:** In this game, the ciphertext E_A is replaced with the encryption result of hw_A. If E_A is a valid ciphertext, the session key is set to be identical to that of π_n^j; otherwise, the session key is uniformly chosen from the elements in dictionary $|X|$. In total, there are $n_s + n_h$ events, each with a probability of $\frac{1}{|X|}$. Therefore, we have:

$$|\mathrm{Prob}\left[\xi_6\right] - \mathrm{Prob}\left[\xi_5\right]| \leq \frac{n_s + n_h}{|X|}$$

Game G_7**:** In this game, we show that the adversary $\mathcal{A}$ can distinguish a real session key from a random number if the following situations occur.

- Case 1: $\mathcal{A}$ makes $Corrupt(\pi_s^i)$ query but no $Send(\pi_n^j, \pi_s^i, \cdot, \cdot)$ query. $\mathcal{A}$ can obtain identity information by issuing n_c $Corrupt$ queries on π_s^i. Meanwhile, $\mathcal{A}$ finds it difficult to retrieve y_A from X_A and Y_A. Thus, if $\mathcal{A}$ can correctly compute the value sk_A with advantage $\mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T}) + \mathbf{Adv}_{\mathcal{A}}^{ECCDH}(\mathcal{S})$, $\mathcal{A}$ can forge E_A with the same advantage.
- Case 2: $\mathcal{A}$ makes $Corrupt(\pi_s^i)$ query and $Send(\pi_n^j, \pi_s^i, \cdot, \cdot)$ query. $\mathcal{A}$ obtains identity information of π_s^i by issuing a $Corrupt$ query. $\mathcal{A}$ can set x_{y_A} and x_{u_A} and request a valid message msg_2' using the $Send(\pi_n^j, \pi_s^i, msg_1', \cdot)$ query. Then $\mathcal{A}$ compute the value $x_{sk_A} = X_A \oplus X_B \oplus H_3(Y_B \cdot (x_{u_A} + x_{y_A}))$. If $\mathcal{A}$ executes $n_s - 1$ times $Send$ queries to guess $ID_{\pi_n^j}$, the probability that $\mathcal{A}$ produces a valid E_A from x_{sk_A} is bounded by $\frac{1}{|X|}$.

Therefore, we have:

$$\begin{aligned} &|\mathrm{Prob}\left[\xi_7\right] - \mathrm{Prob}\left[\xi_6\right]| \\ &\leq \frac{n_s - 1}{|X|} + n_c(\mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T}) + \mathbf{Adv}_{\mathcal{A}}^{ECCDH}(\mathcal{S})) \end{aligned}$$

Game G_8**:** Similar to the Case 2 of game G_7, we can get:

$$|\mathrm{Prob}\left[\xi_8\right] - \mathrm{Prob}\left[\xi_7\right]| \leq \frac{n_s - 1}{|X|}$$

Game G_9**:** This game serves as a bridging step where the advantage of $\mathcal{A}$ in guessing b is completely eliminated. Therefore, we obtain: $\mathrm{Prob}\left[\xi_9\right] = \mathrm{Prob}\left[\xi_8\right] = \frac{1}{2}$. Thus, we have:

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}}^{LLRA}(k) &\leq 2n_p \cdot \epsilon' + \frac{(n_e + n_s)^2}{n_p} + \frac{n_h^2}{2^l} \\ &+ 2(\frac{n_s + n_h}{|X|} + \frac{n_e + n_h}{|X|}) + \frac{4(n_s - 1)}{|X|} \\ &+ 2(n_s + n_c)\mathbf{Adv}_{\mathcal{A}}^{ECDLP}(\mathcal{T}) + 2n_c\mathbf{Adv}_{\mathcal{A}}^{ECCDH}(\mathcal{S}) \end{aligned}$$

The detailed formal security proof is provided in GitHub[1]. □

C. CLR-eGNY Logic Analysis

According to definition 5, 6 in Dziembowski *et al.* [31], we create a new GNY expression.

Based on previous studies, $P \mid\equiv \Lambda(r)$ (P believes, or is entitled to believe, that private key r is *leakage-resilient*) is generalized as follow. $\Lambda_{Z_q^*}^{n,m} = (Encode_{\mathbb{Z}_q^*}^{n,m}, Decode_{\mathbb{Z}_q^*}^{n,m})$ is a $(2\boldsymbol{\lambda}_\Lambda, \epsilon)$-secure leakage-resilient storage scheme, $Refresh_{\mathbb{Z}_q^*}^{n,m}$ is $(l, \boldsymbol{\lambda}_\Lambda, \epsilon')$-secure leakage-resilient for $\Lambda_{Z_q^*}^{n,m}$. P possesses $Encode_{\mathbb{Z}_q^*}^{n,m}(r)$, P possesses session key SK, P possesses $Refresh_{\mathbb{Z}_q^*}^{n,m}(r_L, r_R)$. There is a leakage function $f = (f_1, f_2)$ and $f_1(r_L) = f_2(r_R) = 0.15n \log q$ bits. P leaks a maximum of $f_1(r_L)$ and $f_2(r_R)$ bits of long-term private key.

$$\frac{\begin{array}{l} P \ni Encode_{\mathbb{Z}_q^*}^{n,m}(r), P \mid\sim (f_1(r_L), f_2(r_R)), \\ P \ni SK, P \ni \mathrm{Refresh}_{\mathbb{Z}_q^*}^{n,m}(r_L, r_R) \end{array}}{P \mid\equiv \Lambda(r)}$$

We create three additional rules based on jurisdiction rules of GNY logic, as shown in the following.

Suppose that for principal P, all of the following conditions hold: (1) P believes that Q is an authority on some statement $C(r_P, r_Q)$; (2) P believes that Q believes in $C(r_P, r_Q)$; (3) P believes that r_P is leakage-resilient; (4) Q believes that r_Q is leakage-resilient. Then P ought to believe in $C(r_P, r_Q)$ as well.

$$\frac{\begin{array}{l} P \mid\equiv Q \mid\Longrightarrow C(r_P, r_Q), P \mid\equiv Q \mid\equiv C(r_P, r_Q), \\ P \mid\equiv \Lambda(r_P), Q \mid\equiv \Lambda(r_Q) \end{array}}{P \mid\equiv C(r_P, r_Q)} (LRJ1)$$

Suppose that for principal P, all of the following conditions hold: (1) P believes that Q is honest and competent; (2) P receives a message $X \rightsquigarrow C(r_P, r_Q)$ which P believes Q conveyed; (3) P believes that X is fresh; (4) P believes that r_P

[1]https://github.com/qazcheng/LLRA

[2]https://bblanche.gitlabpages.inria.fr/proverif/

Table II: Automated Formal Verification Using ProVerif.

Descriptions	Check Code	Result
Secrecy of the session key	query attacker (ska) query attacker (ska)	Query not attacker(ska[]) is true Query not attacker(skb[]) is true
Secrecy of ID	noninterf IDa noninterf IDb	Non-interference IDa is true Non-interference IDb is true
Secrecy of pw	weaksecret PW	Weak secret PW is true
Mutual authentication	query inj-event (EndUserA)==>inj-event(BeginServerB).	Query inj-event(EndUserA)==> inj-event(BeginServerB) is true

is leakage-resilient; (5) Q believes that r_Q is leakage-resilient. Then P ought to believe that Q really believes $C(r_P, r_Q)$.

$$\frac{P \mid\equiv Q \mid\Longrightarrow Q \mid\equiv *, P \mid\equiv \mathrm{Q} \mid\sim (X \rightsquigarrow C(r_P, r_Q)), \quad P \mid\equiv \sharp X, P \mid\equiv \Lambda(r_P), Q \mid\equiv \Lambda(r_Q)}{P \mid\equiv Q \mid\equiv C(r_P, r_Q)} (LRJ2)$$

Suppose that for principal P, all of the following conditions hold: (1) P believes that Q is honest and competent; (2) P believes that Q believes that Q believes in $C(r_P, r_Q)$; (3) P believes that r_P is leakage-resilient; (4) Q believes that r_Q is leakage-resilient. Then P ought to believe that Q really believes $C(r_P, r_Q)$.

$$\frac{P \mid\equiv Q \mid\Longrightarrow Q \mid\equiv *, P \mid\equiv Q \mid\equiv Q \mid\equiv C(r_P, r_Q), \quad P \mid\equiv \Lambda(r_P), Q \mid\equiv \Lambda(r_Q)}{P \mid\equiv Q \mid\equiv C(r_P, r_Q)} (LRJ3)$$

We analyze LLRA's security in extended GNY logic (CLR-eGNY) as follow.

1) **Description**

LLRA is rewritten as follows.

$Msg_1 : S_B \triangleleft *X_A, *Y_A, *T_A, *C_A, *Auth$;

$Msg_2 : U_A \triangleleft *X_B, *Y_B, *T_B, *E_B$;

$Msg_3 : S_B \triangleleft *E_A$.

2) **Goal**

The shared session keys between U_A and U_B shall achieve the following goals: **Goal 1**: $U_A \mid\equiv \sharp SK$; **Goal 2**: $U_A \mid\equiv \phi SK$; **Goal 3**: $U_A \mid\equiv U_B \ni SK$; **Goal 4**: $U_B \mid\equiv \sharp SK$; **Goal 5**: $U_B \mid\equiv \phi SK$; **Goal 6**: $U_B \mid\equiv U_A \ni SK$.

3) **Initial Assumptions**

Referring to registration phrase of LLRA, we have several initialization assumptions:

A1: $U_A \mid\equiv \sharp y_A$; A2: $U_A \mid\equiv \phi y_A$;

A3: $U_A \ni y_A, ID_A, hw_A, D_A, \overrightarrow{r_{AL}}, \overrightarrow{r_{AR}}, G, Z_A, Z_B$;

A4: $S_B \mid\equiv \sharp y_B$; A5: $S_B \mid\equiv \phi y_B$;

A6: $S_B \ni y_B, ID_B, hw_B, D_B, \overrightarrow{r_{BL}}, \overrightarrow{r_{BR}}, G, Z_A, Z_B$;

A7: $U_A \mid\equiv S_B \ni Z_A, Z_B$; A8: $S_B \mid\equiv U_A \ni Z_A, Z_B$;

A9: $U_A \mid\equiv U_A \overset{pw}{\leftrightarrow} S_B$; A10: $S_B \mid\equiv S_B \overset{pw}{\leftrightarrow} U_A$;

A11: $U_A \mid\equiv S_B \mid\Longrightarrow S_B \mid\equiv *$; A12: $S_B \mid\equiv U_A \mid\Longrightarrow U_A \mid\equiv *$;

A13: $U_A \mid\equiv S_B \mid\Longrightarrow S_B \ni (u_B, y_B)$;

A14: $S_B \mid\equiv U_A \mid\Longrightarrow U_A \ni (u_A, y_A)$;

A15: $U_A \mid\equiv \Lambda(r_A)$; A16: $S_B \mid\equiv \Lambda(r_B)$.

4) **Proof**

Then, we start the formal proof of **Goal 1**, **Goal 2** and **Goal 3** in GNY logic.

Goal 1: According to A1, A3 and the rules F1, P2, P4 and F10, we can get that U_A believes that $(sk_A||sid_A)$ is fresh. Similarly, we can prove the Goal 4.

Goal 2: According to A1 and the rules R1 and R10, we can get that U_A believes that $((u_A + y_A) \cdot G)$ is recognizable. Similarly, we can prove the Goal 5.

Goal 3: According to A1 and the rule F10, we can get that S_A believes that X_A is fresh, and $X_A = H_2(hw_A \oplus D_A \oplus u_A \oplus y_A)$.

According to the rule F1, we can get that U_A believes that sk_A is fresh, and $sk_A = X_B \oplus X_A \oplus H_3(V_A)$.

According to A2 and the rule R5, we can get that U_A believes that X_A is recognizable, and $X_A = H_2(hw_A \oplus D_A \oplus u_A \oplus y_A)$.

According to the rule R1, we can get that U_A believes that sk_A is recognizable.

According to A9 and the rule I1, we can get that U_A believes that S_B once conveyed $(T_B, hw_B, y_B, X_A, V_B, T_A)$, and U_A owns sk_B.

According to the rule I7, we can get that U_A believes that S_B once conveyed V_B.

According to the rule F1, we can get that U_A believes that V_B is fresh, and $V_B = (u_B + y_B) \cdot Y_A$.

According to our new GNY expression, we can get that $U_A \mid\equiv \Lambda(r_A)$.

According to A11 and the rule LRJ2, we can get that U_A believes that S_B believes that S_B possesses u_B.

According to A13 and the rule LRJ1, we can get that U_A believes that S_B possesses u_B.

According to the rationality rule from postulate P3, we can get that $U_A \mid\equiv S_B \ni y_B$.

According to the rationality rule from postulate P2, we can get that $U_A \mid\equiv S_B \ni (u_B + y_B) \cdot G$, so we can obtain that $U_A \mid\equiv S_B \ni Y_B$.

According to the rule I6, we can get that $U_A \mid\equiv S_B \ni V_B$.

According to the rationality rule from postulate P5, we can get that $U_A \mid\equiv S_B \ni Y_A$.

According to the rationality rule from postulate P2, we can get that $U_A \mid\equiv S_B \ni Z_A||Z_B||Y_A||Y_B$, so we can obtain that $U_A \mid\equiv S_B \ni sid_B$.

According to the rationality rule from postulate P2 and P4, we can get that $U_A \mid\equiv S_B \ni H(sk_B||sid_B)$, so we can obtain that $U_A \mid\equiv S_B \ni SK_B$.

The detailed CLR-eGNY proof for other goals is provided in GitHub[1]. A1, F1, and other rules used above can be found in [28].

D. Automatic Formal Verification Using ProVerif

As shown in Table II, we use an automated tool ProVerif[2] cryptographic protocol verifier to verify reachability properties. We prove that LLRA provides secrecy of the session key and resistance to off-line guessing attack, etc. The ProVerif code is given in GitHub[1].

E. Informal Security Analysis

In the Table III, IV, we analyze general security features of the proposed scheme.

- **Perfect forward secrecy:** Suppose that A's long-term private values are compromised. The adversary $\mathcal{M}$ cannot compute session key sk_A or sk_B without knowing the

Table III: Security Properties.

Schemes	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
Lightweight LRAKE [18]	×	-	×	×	×	√	×	×	-
ULSS [34]	×	-	√	√	×	-	×	×	-
DM2018-1 [35]	√	-	√	√	×-	√	√	-	
DM2018-2 [36]	√	-	√	√	×	-	√	√	-
SM2 [37]	√	√	√	√	√	×	√	√	√
PSLA [3]	×	√	√	√	√	×	√	√	√
CRISP [38]	×	√	√	×	√	×	√	√	√
LRAKE [23]	√	-	×	×	√	√	×	√	√
LRAKE-IoT [24]	√	-	×	√	√	√	√	√	√
LYZ [21]	√	√	√	√	√	×	×	√	√
LLRA	√	√	√	√	√	√	√	√	√

Note: F_1-perfect forward secrecy; F_2-resistance to off-line password guessing attack; F_3-resistance to replay attack; F_4-mutual authentication; F_5-resistance to key compromise impersonation attack; F_6-leakage-resilient (resistance to side-channel attack), F_7-resistance to man-in-the-middle attack; F_8-resistance to impersonation attack; F_9-resistance to DoS attack.

Table IV: Security Models.

Schemes	S_1	S_2	S_3
Lightweight LRAKE [18]	CLR-eCK	-	-
ULSS [34]	-	-	ProVerif
DM2018-1 [35]	-	-	ProVerif
DM2018-2 [36]	Random Oracle	-	ProVerif
SM2 [37]	-	-	-
PSLA [3]	modified Bellare-Rogaway (mBR) [39]	GNY	-
CRISP [38]	Universally Composable (UC) framework [40]	-	-
LRAKE [23]	eCK	-	-
LRAKE-IoT [24]	CLR-eCK	-	-
LYZ [21]	BPR	-	-
LLRA	CLR-eBPR	CLR-eGNY	ProVerif

Note: S_1: formal security analysis using provable security model; S_2: formal logic verification; S_3: automated formal verification.

Table V: Comparison of Security Model.

Model		**Description**
Formal Security Model	LR-eCK [25]	Add new query ($StaticKeyLeakage(\cdot,\cdot)$) for adversary to the eCK security model.
	(·)AFL-eCK [17]	Add leakage query (use a leakage function f embedded with the $Send$ query) to the eCK security model.
	CLR-eCK [26]	Based on the eCK security model, CLR-eCK allows continuous access to arbitrarily large amounts of long-term secret keys by adversary, which is a bounded leakage of protocols.
	CLR-eBPR	Add a leakage function f embedded with the $Send$ query to the BPR security model. And CLR-eBPR enforces the restriction that the amount of leakage per observation is bounded.
Logic	GNY [28]	No rules related to leakage-resilient property
	BAN [27]	No rules related to leakage-resilient property
	CLR-eGNY	Leakage-resilient security property is considered in the Jurisdiction Rule of GNY logic.

ephemeral private key y_A or y_B that was randomly chosen for each session.

- **Resistance to off-line password guessing attack:** In LLRA $X_A = H_2(hw_A \oplus D_A \oplus u_A \oplus y_A)$, the adversary $\mathcal{A}$ is unaware of the ephemeral secret y_A. Even if $\mathcal{A}$ is able to obtain X_A, hw_A is still protected. Therefore, it is computationally infeasible for $\mathcal{A}$ to guess the password pw_A.
- **Resistance to replay attack:** The freshness of the timestamps (T_A, T_B) ensures that the proposed protocol is immune to replay attacks.
- **Mutual authentication:** S_B extracts y_A^* and computes C_A^* to check the validity of C_A. S_B computes E_B and sends it to U_A to prove its legitimacy. U_A decrypts E_B and obtains $X_A^*$and V_B^*. Then U_A checks $X_A^* = X_A$ and $V_B^* = V_A$. Then U_A computes E_A and sends it to S_B to complete the mutual authentication.
- **Resistance to key compromise impersonation attack:** Assuming that U_A's hw_A and D_A are compromised. Because $\mathcal{A}$ does not know the ephemeral secret y_B of S_B. $\mathcal{A}$ cannot impersonate S_B to participate in the key exchange.
- **Resistance to man-in-the-middle attack:** The adversary $\mathcal{A}$ pretends to be the communicating party but cannot pass the verification of the following formulae $C_A^* \stackrel{?}{=} C_A$, $X_A^* \stackrel{?}{=} X_A$ and $V_B^* \stackrel{?}{=} V_A$. Thus, the proposed scheme can resist man-in-the-middle attack.
- **Resistance to impersonation attack:** $\mathcal{A}$ needs to obtain the long-term secret ($\overrightarrow{r_{AL}}$, $\overrightarrow{r_{AR}}$ or $\overrightarrow{r_{BL}}$, $\overrightarrow{r_{BR}}$) to pass the verification ($C_A^* = C_A$, $X_A^* = X_A$ and $V_B^* = V_A$). However, the attacker finds it difficult to obtain the long-term secret when the refreshing protocol is used. As a result, our scheme can resist impersonation attacks.
- **Resistance to DoS attack:** In each session, U_A and S_B verify the timestamp T_A and T_B upon receiving a message. If the verification fails, the subsequent steps are

Table VI: Execution Time of Different Cryptographic Operations on Different Devices.

Notations	Descriptions	Laptop (ms)	Raspberry Pi (ms)	Output Length (bits)
T_{bp}	Bilinear pairing	4.122	53.179	512
T_{sm}	scalar multiplication in G	1.832	32.563	512
T_{pa}	point addition in G	0.053	0.629	512
T_{exp}	Modular exponentiation in G	1.435	34.834	512
T_h	(Keyed) One-way hash	0.003	0.041	256
T_{sed}	Encryption and decryption	0.054	0.301	256

halted. Thus, our scheme can resist DoS attacks.

F. Security Analysis of Leakage-Resilient Property

In this part, we analyze how LLRA can resist side-channel attacks. This also means that LLRA can provide leakage-resilient property. Assuming that attacker $\mathcal{A}$ obtains the transmitted message via $Send$-query and a few bits of $\overrightarrow{r_{AL}}$ or $\overrightarrow{r_{AR}}$ leaked using the leakage function $f = (f_{Li}, f_{Ri})$ with boundary $|f_{Li}(\overrightarrow{r_{AL}})| \leq r_L$ and $|f_{Ri}(\overrightarrow{r_{AR}})| \leq r_R$. U_A used the following methodologies to resist side-channel attacks. U_A does not directly store the long-term private key r_A. With the $Encode_{\mathbb{Z}_q^*}^{n,m}(r_A)$ function, r_A is split into n-dimensional vectors $\overrightarrow{r_{AL}} = (r_{AL_1}, \ldots, r_{AL_n})$ and $\overrightarrow{r_{AR}} = (r_{AR_1}, \ldots, r_{AR_n})$. Then U_A stores $\overrightarrow{r_{AL}}$ and $\overrightarrow{r_{AR}}$ independently and destroys r_A. In the authentication key exchange phase, the ephemeral secret y_A is used to regenerate the two n-dimensional vectors $\overrightarrow{v_B} = (y_B, r_{BL_1}, \ldots, r_{BL_n})$ and $\overrightarrow{w_B} = (1, r_{BR_1}, \ldots, r_{BR_n})$. Finally, U_A and S_B generate a shared session key, and the storage of $\overrightarrow{r_{AL}}$ and $\overrightarrow{r_{AR}}$ is updated using the $Refresh_{\mathbb{Z}_q^*}^{n,m}(\overrightarrow{r_{AL}}, \overrightarrow{r_{AR}})$ in Definition 4. S_B performs the same steps.

As shown in Table V, we compare the CLR-eBPR security model and CLR-eGNY logic to typical security models and logics.

G. Summary

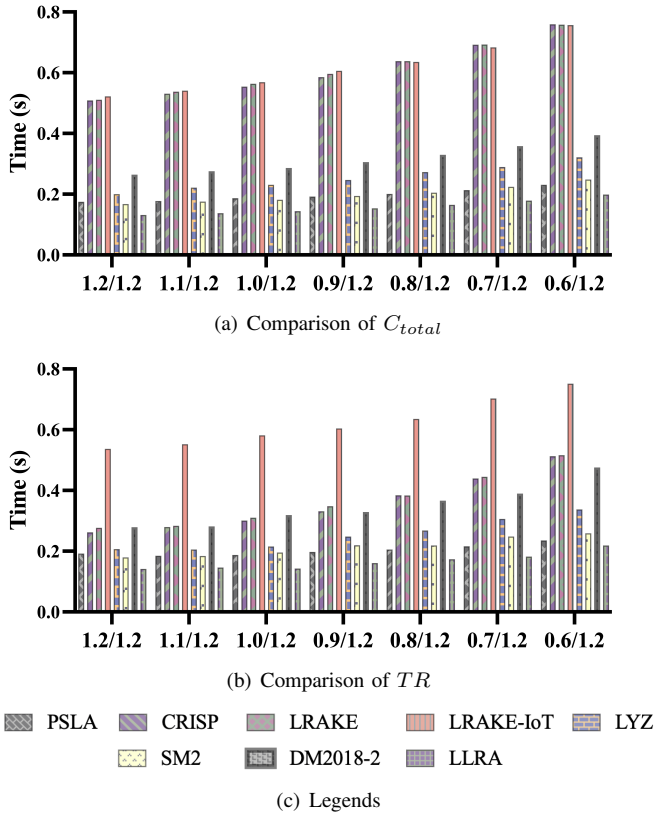
The security analysis shows that LLRA can meet the security goals: providing resistance to side-channel attacks, authenticity, session key security, and perfect forward secrecy, as well as being able to resist other common attacks.

VI. Performance Analysis

This section studies the performance of the proposed scheme LLRA and benchmarks. Benchmarks include PSLA [3], CRISP [38], LRAKE [23], LRAKE-IoT [24], LYZ [21] and Shangyong Mima2 (SM2) [37] from State Cryptography Administration of China.

A. Experiment Set Up

We conducted experiments using two Raspberry Pis (OS: Raspbian GNU/Linux 10, CPU: ARM Cortex-A53, 2GB memory, 16GB storage). $RaspA$ denotes the Raspberry Pi representing U_A, $RaspB$ denotes the Raspberry Pi representing S_B. C_{total} denotes the total computational time. TR represents the running time of the protocol. C_{comm} denotes the communication cost. We ran each of the experiments ten times, and took the average value as final result. We measured the running time of various cryptographic operations on the laptop and Raspberry Pi, as shown in Table VI.



(a) Comparison of C_{total}

(b) Comparison of TR

(c) Legends

Figure 6: Comparison of C_{total} and TR in Experiment II.

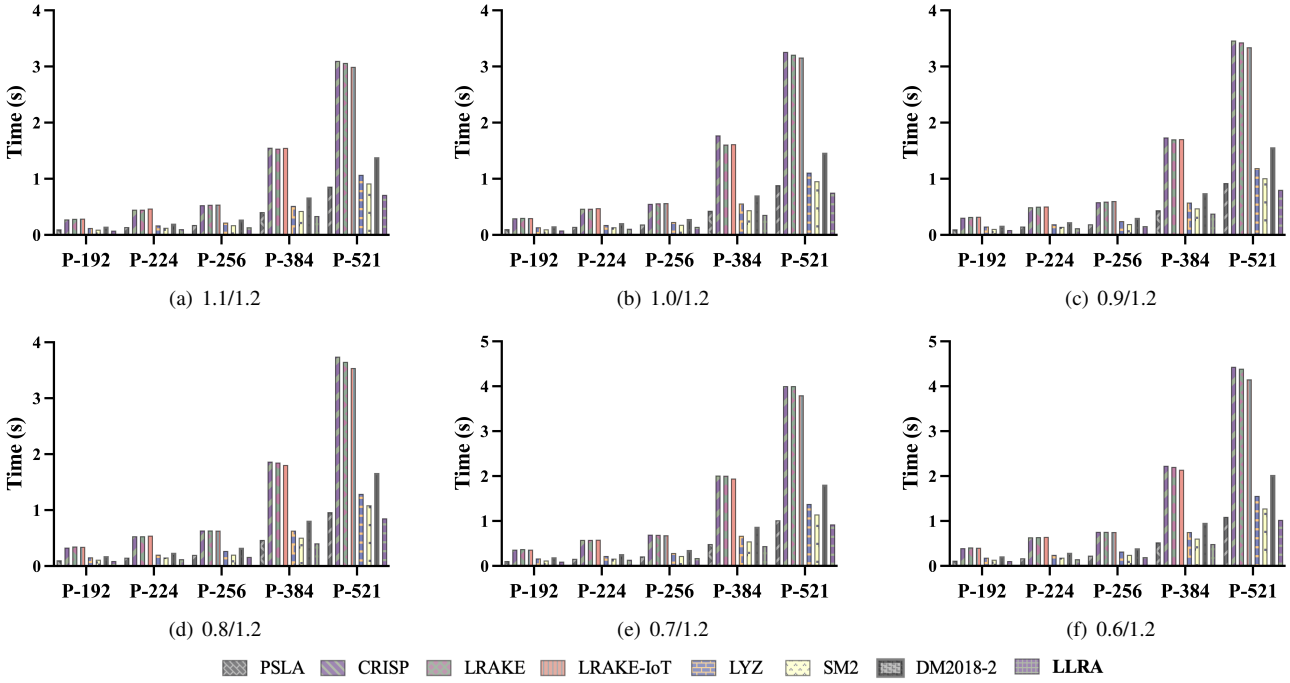
B. Experiment I

As shown in Table VII, we compared the number of cryptographic operations required by LLRA and benchmark protocols under elliptic curve P-256. CPU frequency of both Raspberry Pis are 1.2GHz. C_{total} of ULSS [34] $C_{total}^{ULSS} = (2T_{sm} + 4T_h) + (2T_sm + 6T_h) \approx 0.06653 + 0.06680 = 0.13333s$. Although ULSS has several interesting features like ultra-lightweight design, it is vulnerable to different attacks that is, Man-in-the-middle and impersonation attacks; and it had 4 communication rounds. C_{total} of DM2018-1 [35] $C_{total}^{DM2018-1} = (4T_{sm} + 4T_h) + (4T_{sm} + 4T_h) \approx 0.12669 + 0.13193 = 0.25862s$; C_{total} of DM2018-2 [36] $C_{total}^{DM2018-2} = (4T_{sm} + 5T_h) + (4T_{sm} + 5T_h) \approx 0.13206 + 0.13231 = 0.26437s$. Both DM2018-1 and DM2018-2 have had 4 ECC scalar multiplication by each entites, with similar results for $C_{total}^{DM2018-1}$ and $C_{total}^{DM2018-2}$. C_{total} of SM2 [37] $C_{total}^{SM2} = (2.5T_{sm} + 2T_h) + (2.5T_{sm} + 2T_h) \approx 0.08350 + 0.08433 = 0.16783s$; C_{total} of PSLA [3] $C_{total}^{PSLA} = (4T_{sm} + 5T_h) + (4T_{sm} + 5T_h) \approx 0.05147 + 0.12333 = 0.17480s$. Both PSLA and SM2 have had 5 ECC scalar multiplication of total compuataional cost, with similar result for C_{total}^{SM2} and C_{total}^{PSLA}. In addition, we have $C_{total}^{CRISP} = (3T_{exp} + T_{MTP} + 3T_{bp}) + (3T_{exp} + T_{MTP} + 3T_{bp}) \approx 0.25320 + 0.25600 = 0.50920s$,

Table VII: Computational and Communication Costs.

Schemes	$\mathbf{U_A}$	$\mathbf{S_B}$	$\mathbf{C_{comm}}$ (bits)	Rounds	$\mathbf{C_{total}}$	TR
ULSS [34]	$2T_{sm}+4T_h$ 0.06653s	$2T_{sm}+6T_h$ 0.06680s	1024	4	0.13333s	0.07034s
DM2018-1 [35]	$4T_{sm}+4T_h$ 0.12669s	$4T_{sm}+4T_h$ 0.13193s	2688	2	0.25862s	0.28751s
DM2018-2 [36]	$4T_{sm}+5T_h$ 0.13206s	$4T_{sm}+5T_h$ 0.13231s	1792	3	0.26437s	0.27903s
SM2 [37]	$2.5T_{sm}+2T_h$ 0.08350s	$2.5T_{sm}+2T_h$ 0.08433s	1344	3	0.16783s	0.17954s
PSLA [3]	$1.5T_{sm}+5T_h$ 0.05147s	$3.5T_{sm}+5T_h$ 0.12333s	2112	3	0.17480s	0.19244s
CRISP [38]	$3T_{exp}+T_{MTP}+3T_{bp}$ 0.25320s	$3T_{exp}+T_{MTP}+3T_{bp}$ 0.25600s	3136	4	0.50920s	0.26256s
LRAKE [23]	$4T_{exp}+2T_{bp}$ 0.25707s	$4T_{exp}+2T_{bp}$ 0.25384s	1024	2	0.51091s	0.27733s
LRAKE-IoT [24]	$7T_{exp}+2T_h$ 0.22400s	$6T_{exp}+2T_h+2T_{bp}$ 0.29810s	2560	3	0.52210s	0.53688s
LYZ [21]	$3T_{exp}+7T_h+2T_{sed}$ 0.10296s	$3T_{exp}+7T_h+2T_{sed}$ 0.09736s	1728	3	0.20032s	0.20721s
LLRA	$2T_{sm}+5T_h+3T_{sed}$ 0.06567s	$2T_{sm}+5T_h+3T_{sed}$ 0.06578s	2624	3	0.13145s	0.14155s

Notes: T_{bp}-Bilinear pairing; T_{sm}-ECC scalar multiplication; T_{pa}-ECC point addition; T_{exp}-Modular exponentiation; T_h-(keyed) one-way hash; T_{MTP}-Map-to-point hash.



(a) 1.1/1.2 (b) 1.0/1.2 (c) 0.9/1.2

(d) 0.8/1.2 (e) 0.7/1.2 (f) 0.6/1.2

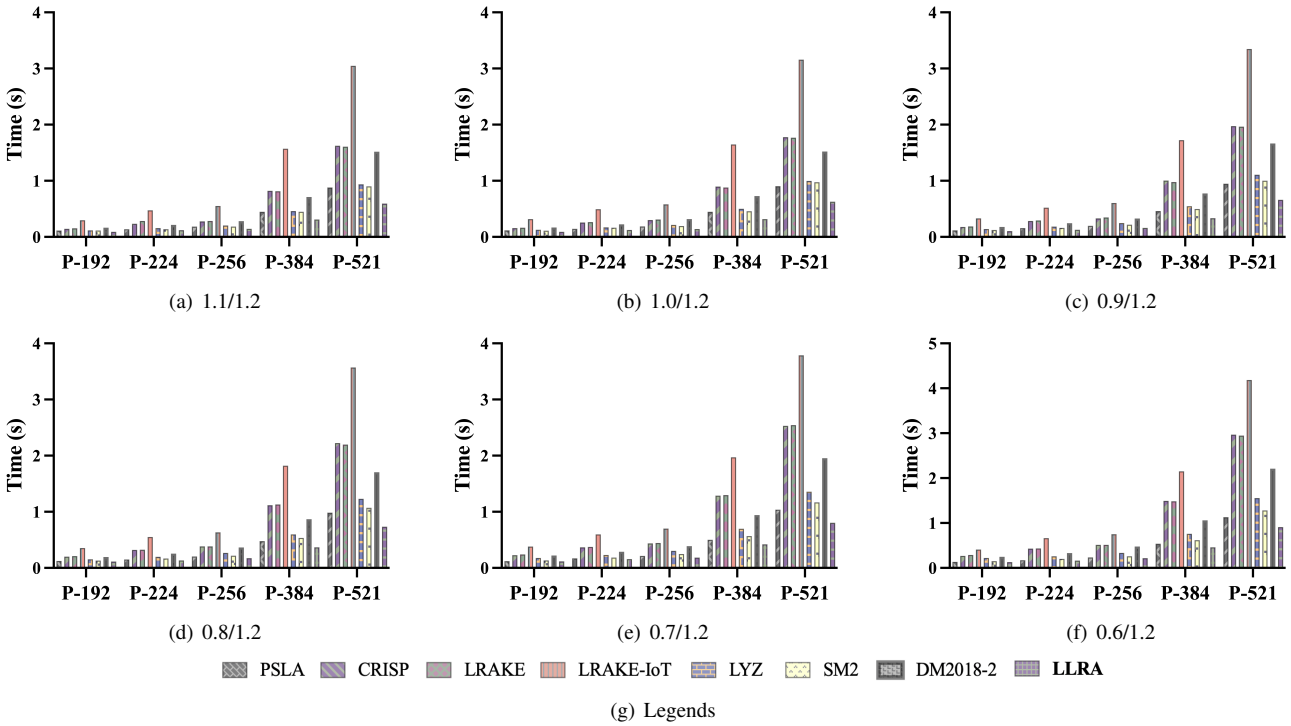
(g) Legends

Figure 7: Comparison of C_{total} in Experiment III.

$C_{total}^{LRAKE} = (4T_{exp}+2T_{bp})+(4T_{exp}+2T_{bp}) \approx 0.25707 + 0.25384 = 0.51091s$, $C_{total}^{LRAKE-IoT} = (7T_{exp}+2T_h)+(6T_{exp}+2T_h+2T_{bp}) \approx 0.22400+0.29810 = 0.52210s$, $C_{total}^{LYZ} = (3T_{exp}+7T_h+2T_{sed})+(3T_{exp}+7T_h+2T_{sed}) \approx 0.10296+0.09736 = 0.20032s$ and $C_{total}^{LLRA} = (2T_{sm}+5T_h+3T_{sed})+(2T_{sm}+5T_h+3T_{sed}) \approx 0.06567+0.06578 = 0.13145s$. It can be seen that LLRA had the shortest total computational time (C_{total}) and running time (TR).

C. Experiment II

We measured C_{total} and TR of LLRA authentication stage between two Raspberry Pis under elliptic curve P-256. F_A and F_B denote the CPU frequency of $RaspA$ and CPU frequency of $RaspB$ respectively. We tested the following combinations F_A/F_B: 1.2/1.2, 1.1/1.2, 1.0/1.2, 0.9/1.2, 0.8/1.2, 0.7/1.2 and 0.6/1.2. As shown in Figure 6, LLRA had the shortest C_{total} and TR.



(a) 1.1/1.2 (b) 1.0/1.2 (c) 0.9/1.2

(d) 0.8/1.2 (e) 0.7/1.2 (f) 0.6/1.2

(g) Legends

Figure 8: Comparison of TR in Experiment III.

D. Experiment III

Based on the experimental setting of Experiment II, we conducted further tests under different elliptic curves P-192, P-224, P-256, P-384, P-521. As shown in Figure 7 and 8, LLRA had the shortest C_{total} under the five elliptic curves when F_A/F_B =1.2/1.2 and 1.1/1.2. LLRA had the shortest C_{total} under four elliptic curves (P-192, P-224, P-256, P-384) when F_A/F_B =1.0/1.2, 0.9/1.2, 0.8/1.2, 0.7/1.2, and 0.6/1.2. In the remaining cases, C_{total} of PSLA is the smallest. In addition, LLRA had the shortest TR in all cases. Besides, we had the following findings. (1) The computational time of scalar multiplication is significantly longer than the computational time of hashing, AES encryption/decryption, and XOR. (2) In the case of elliptic curve (P-521), the computational time of scalar multiplication is longer than that using P-192, P-224, P-256, P-384. (3) U_A in LLRA involves an extra scalar multiplication compared to U_A in PSLA; thus, when F_A/F_B decreases, the computational time of U_A in LLRA increases faster than that in PSLA.

E. Experiment IV

In this experiment, we further widen the CPU frequency gap between both parties. We conducted this experiment using Raspberry Pi $RaspA$ and virtual machine VM (OS: Ubuntu, CPU: i5-8250U). F_{VM} denotes the CPU frequency of VM. We tested the following combinations F_A/F_{VM}: 1.2/12 and 0.6/12. $VS1$ denotes the ratio of C_{total} between two schemes. $VS2$ denotes the ratio of TR between tow schemes. As shown in Figure 9, we had the following findings. (1) The smaller the value of F_A/F_B (or F_A/F_{VM}), the greater the advantage of PSLA. (2) When there is a significant difference in CPU frequency between the communication parties, the higher the order of the elliptic curve, the greater the advantage of PSLA.

F. Summary

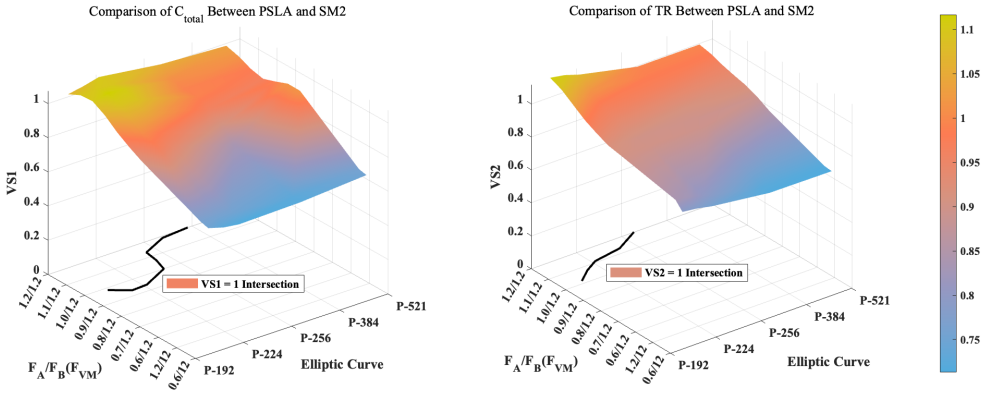
The experimental results clearly demonstrated that LLRA can effectively reduce the computational time and running time. Compared to benchmark protocols, the running time of LLRA was decreased by at least 14%. When the computing capabilities of both parties are comparable, the total computational time of LLRA was decreased by at least 33%, making it more efficient and more suitable for smart meters.

VII. Use Case

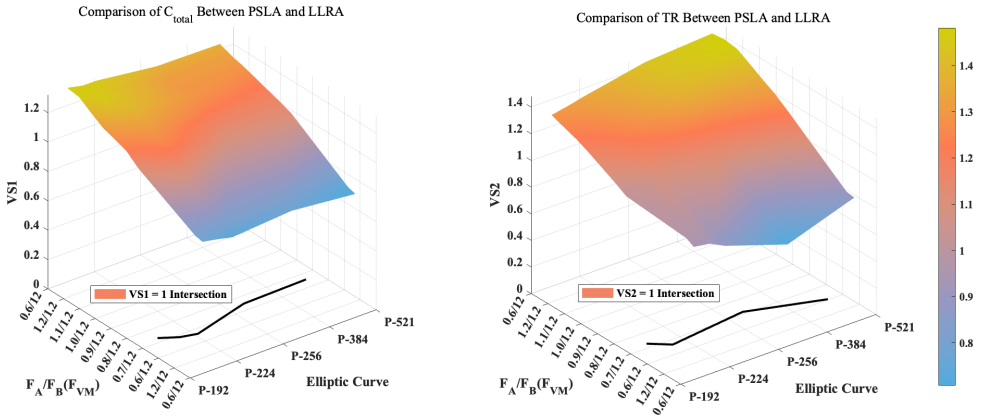
In this section, we will explore the applications and advantages of LLRA.

A. Scenario Description

In Taiyuan, Shanxi Province, China, the 246 charging stations are provided by the Taiyuan Branch of China Tower Co., Ltd. Each charging pile is equipped with a smart meter. According to the application requirements, the smart meters of the charging piles must undergo authentication and upload electricity consumption data to the NAN gateways on a scheduled basis—daily, weekly, or monthly. Subsequently, this data is transmitted to the Taiyuan Branch of China Tower Co., Ltd. The NAN gateway will also receive feedback from the service provider and then forward it to the smart meters.



(a) Comparison of C_{total} and TR between PSLA and SM2. In this case, we set $VS1 = C_{total}^{PSLA}/C_{total}^{SM2}$ and $VS2 = TR^{PSLA}/TR^{SM2}$.



(b) Comparison of C_{total} and TR between PSLA and LLRA. In this case, we set $VS1 = C_{total}^{PSLA}/C_{total}^{LLRA}$ and $VS2 = TR^{PSLA}/TR^{LLRA}$.

Figure 9: Comparison of C_{total} and TR in Experiment IV.

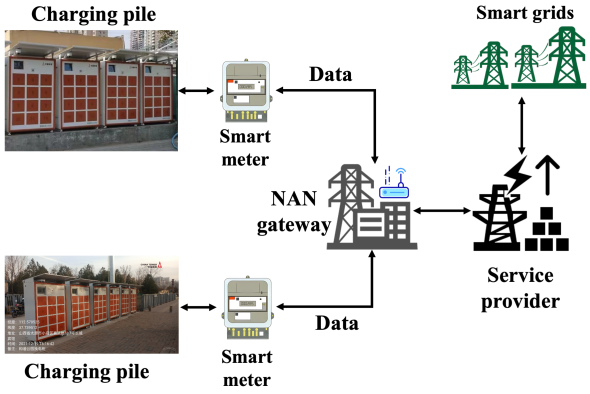


Figure 10: Use Case of LLRA.

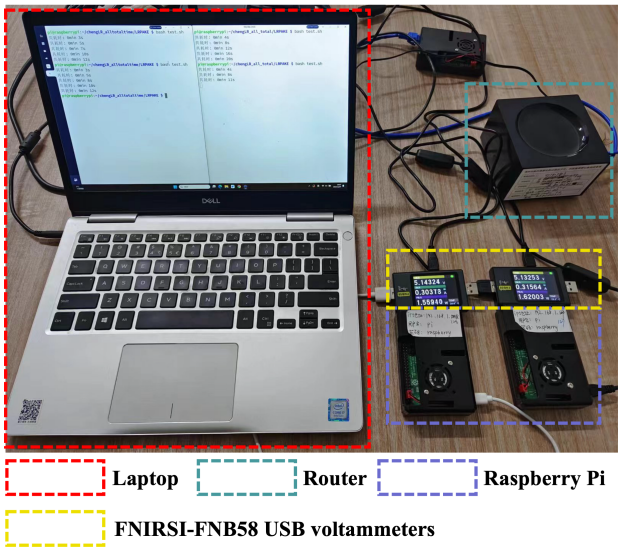


Figure 11: Hardware Platform of Simulation Experiment in Use Case.

B. Advantages of Using LLRA

Based on data from China Tower Co., Ltd. Shanxi Branch, it has been observed that, a charging station is connected to 12 smart meters. We analyze the advantages of LLRA for smart meters. As shown in Table VIII, LLRA provides higher level of security, which was analysed in Section V. We

Table VIII: Comparison of Protocols for Smart Meters.

Scenario requirements	Description	Protocol						
		PSLA [3]	CRISP [38]	LRAKE [23]	LRAKE-IoT [24]	LYZ [21]	SM2 [37]	**Ours**
High level of security	Provable security analysis, logic analysis, automatic formal verification and meets security properties	×	×	×	×	×	×	√
Minimizing the authentication delay	The total authentication delay (s) $T_d = n_{BCS} \cdot TR$, where $n_{BCS} = 12$.	2.5854	3.6585	3.3659	7.1220	3.1707	2.3902	2.2439
Minimizing the energy consumption	The energy consumption (J)	2.4930	3.1614	3.1443	4.0629	2.5561	2.1724	2.0102

Notes: 1. We compute the authentication delay by using the result of Experiment I in section VI;
2. In energy consumption experiments, we use the same experiments environment as Experiment III, and set $F_A = F_B = 0.6GHz$. We use 2 FNIRSI-FNB58 USB voltammeters to test the energy consumption of $RsapA$ and $RaspB$ respectively. The process of the experiment can be found at this link[1]. When Raspberry Pi is in idle time, $RaspA$ consumes 1.6556J per second, $RaspB$ consumes 1.6014J per second. The elliptic curve used for testing is P-256.

conducted simulation experiments using devices described in Section VI. As shown in Figure 11, the Raspberry Pi $RaspA$ on the left simulates a smart meter, while the Raspberry Pi $RaspB$ on the right simulates the NAN gateway. In addition, we use FNIRSI-FNB58 USB voltammeters to measure the power consumption of Raspberry Pis. We use elliptic curve cryptosystem (P-256) based on FIPS 186-3 [41] of NIST; and SHA-256 is chosen as the secure hash function. The codes was provided in GitHub[1]. Given a fixed time period of approximately 30,000 ms, $30000/183.55 \approx 161$ charging piles could be authenticated using LLRA, whereas only 151 using SM2 [37], and 141 using PSLA [3]. During peak hours, compared to LLRA protocol, the benchmark protocols, due to its higher authentication latency, prevents NAN gateways from authenticating more smart meters. Therefore, during peak electricity consumption periods, the concurrent operation of LLRA has the potential to alleviate congestion. The energy consumed by 12 charging piles to run LLRA could support $12 \times 2.0102/2.1724 \approx 11$ charging piles running SM2 [37] or 9 charging piles running PSLA [3]. Therefore, the benchmarks will consume more energy compared to LLRA protocol.

C. Summary

The analysis of the use case demonstrates that compared to benchmarks, LLRA offers higher level of security, reduces authentication latency and energy consumption. Compared to benchmark protocols, the authentication delay of LLRA was decreased by at least 14%; in addition, the energy consumption of LLRA was decreased by at least 7%.

VIII. Conclusion

In this paper, we proposed a new scheme called Lightweight Leakage-Resilient Authentication Key Exchange (LLRA) for smart meters. The design of this scheme considered long-term private key leakage due to side-channel attacks. Provable security analysis using CLR-eBR security model, CLR-eGNY logic analysis and automatic verification using ProVerif prove the security of LLRA. Experimental results demonstrate that LLRA achieves the design goals and has significant performance advantages over benchmark solutions in terms of authentication delay and energy consumption.

Given the latest advances in quantum computing, anti-quantum security schemes are needed; and several lattice-based public key encryption and key exchange schemes have been proposed [42]–[45]. In the future, we will explore lattice-based techniques to enhance the security of LLRA. In addition, we will seek to explore the applications of LLRA in other industrial IoT scenarios. Additionally, we will analyze the security threats to LLRA in these scenarios and strive to enhance its security and efficiency.

Acknowledgment

This work was supported in part by the Shanxi Scholarship Council of China under Grant 2021-038; in part by the Special Project for Guiding the Transformation of Scientific and Technological Achievements of Shanxi Province under Grant 202204021301043; and in part by the Fundamental Research Project of Shanxi Province under Grant 20210302123130.

References

[1] M. Mustapa, M. Y. Niamat, A. P. Deb Nath, and M. Alam, "Hardware-oriented authentication for advanced metering infrastructure," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1261–1270, 2018.

[2] A. Mondal, S. Misra, and M. S. Obaidat, "Distributed home energy management system with storage in smart grid using game theory," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1857–1866, 2017.

[3] S. Chai, H. Yin, B. Xing, Z. Li, Y. Guo, D. Zhang, X. Zhang, D. He, J. Zhang, X. Yu, W. Wang, and X. Huang, "Provably secure and lightweight authentication key agreement scheme for smart meters," *IEEE Transactions on Smart Grid*, vol. 14, no. 5, pp. 3816–3827, 2023.

[4] K. Mahmood, S. A. Chaudhry, H. Naqvi, T. Shon, and H. F. Ahmad, "A lightweight message authentication scheme for smart grid communications in power sector," *Computers & Electrical Engineering*, vol. 52, pp. 114–124, 2016.

[5] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial internet of things," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1133–1146, 2020.

[6] A. A. Khan, V. Kumar, M. Ahmad, S. Rana, and D. Mishra, "Palk: Password-based anonymous lightweight key agreement framework for smart grid," *International Journal of Electrical Power & Energy Systems*, vol. 121, p. 106121, 2020.

[7] S. A. Chaudhry, "Correcting "palk: Password-based anonymous lightweight key agreement framework for smart grid"," *International Journal of Electrical Power & Energy Systems*, vol. 125, p. 106529, 2021.

[8] S. A. Chaudhry, K. Yahya, S. Garg, G. Kaddoum, M. M. Hassan, and Y. B. Zikria, "Las-sg: An elliptic curve-based lightweight authentication scheme for smart grid environments," *IEEE Transactions on Industrial Informatics*, vol. 19, no. 2, pp. 1504–1511, 2023.

[9] X. Lou, T. Zhang, J. Jiang, and Y. Zhang, "A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography," *ACM Computing Surveys (CSUR)*, vol. 54, no. 6, pp. 1–37, 2021.

[10] M. Green, L. Rodrigues-Lima, A. Zankl, G. Irazoqui, J. Heyszl, and T. Eisenbarth, "Autolock: Why cache attacks on arm are harder than you think," in *26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 1075–1091.

[11] D. Genkin, I. Pipman, and E. Tromer, "Get your hands off my laptop: physical side-channel key-extraction attacks on pcs: Extended version," *Journal of Cryptographic Engineering*, vol. 5, pp. 95–112, 2015.

[12] D. Bleichenbacher, "Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs# 1," in *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*. Springer, 1998, pp. 1–12.

[13] S. Vaudenay, "Security flaws induced by cbc padding—applications to ssl, ipsec, wtls..." in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2002, pp. 534–545.

[14] C. Trippel, D. Lustig, and M. Martonosi, "Checkmate: Automated synthesis of hardware exploits and security litmus tests," in *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2018, pp. 947–960.

[15] N. El Mrabet, J. J. Fournier, L. Goubin, and R. Lashermes, "A survey of fault attacks in pairing based cryptography," *Cryptography and Communications*, vol. 7, pp. 185–205, 2015.

[16] V. O. Nyangaresi and Z. Mohammad, "Privacy preservation protocol for smart grid networks," in *2021 International Telecommunications Conference (ITC-Egypt)*, 2021, pp. 1–4.

[17] J. Alawatugoda, D. Stebila, and C. Boyd, "Modelling after-the-fact leakage for key exchange," in *Proceedings of the 9th ACM symposium on Information, computer and communications security*, 2014, pp. 207–216.

[18] J. Zhang, F. Zhang, X. Huang, and X. Liu, "Leakage-resilient authenticated key exchange for edge artificial intelligence," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 6, pp. 2835–2847, 2021.

[19] O. Ruan, M. Zhang, and J. Chen, "Leakage-resilient password-based authenticated key exchange," in *Algorithms and Architectures for Parallel Processing: 17th International Conference, ICA3PP 2017, Helsinki, Finland, August 21-23, 2017, Proceedings 17*. Springer, 2017, pp. 285–296.

[20] O. Ruan, J. Chen, and M. Zhang, "Provably leakage-resilient password-based authenticated key exchange in the standard model," *IEEE Access*, vol. 5, pp. 26 832–26 841, 2017.

[21] H. Lian, Y. Yang, and Y. Zhao, "Efficient and strong symmetric password authenticated key exchange with identity privacy for iot," *IEEE Internet of Things Journal*, vol. 10, no. 6, pp. 4725–4734, 2023.

[22] J. Alawatugoda, D. Stebila, and C. Boyd, "Continuous after-the-fact leakage-resilient eck-secure key exchange," in *IMA international conference on cryptography and coding*. Springer, 2015, pp. 277–294.

[23] J.-D. Wu, Y.-M. Tseng, and S.-S. Huang, "Efficient leakage-resilient authenticated key agreement protocol in the continual leakage eck model," *IEEE Access*, vol. 6, pp. 17 130–17 142, 2018.

[24] A.-L. Peng, Y.-M. Tseng, and S.-S. Huang, "An efficient leakage-resilient authenticated key exchange protocol suitable for iot devices," *IEEE Systems Journal*, vol. 15, no. 4, pp. 5343–5354, 2021.

[25] D. Moriyama and T. Okamoto, "Leakage resilient eck-secure key exchange protocol without random oracles," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, 2011, pp. 441–447.

[26] J. Alawatugoda, C. Boyd, and D. Stebila, "Continuous after-the-fact leakage-resilient key exchange," in *Information Security and Privacy: 19th Australasian Conference, ACISP 2014, Wollongong, NSW, Australia, July 7-9, 2014. Proceedings 19*. Springer, 2014, pp. 258–273.

[27] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems (TOCS)*, vol. 8, no. 1, pp. 18–36, 1990.

[28] L. Gong, R. M. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols." in *IEEE Symposium on Security and Privacy*, vol. 1990. Citeseer, 1990, pp. 234–248.

[29] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[30] D. Hankerson, S. Vanstone, and A. Menezes, *Guide to Elliptic Curve Cryptography*, ser. Springer Professional Computing. Springer New York, NY, 2004.

[31] S. Dziembowski and S. Faust, "Leakage-resilient cryptography from the inner-product extractor," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2011, pp. 702–721.

[32] A. Gopstein, C. Nguyen, C. O'Fallon, N. Hastings, D. Wollman *et al.*, *NIST framework and roadmap for smart grid interoperability standards, release 4.0*. Department of Commerce. National Institute of Standards and Technology . . . , 2021.

[33] M. Bellare, D. Pointcheval, and P. Rogaway, "Authenticated key exchange secure against dictionary attacks," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2000, pp. 139–155.

[34] D. Abbasinezhad-Mood and M. Nikooghadam, "An ultra-lightweight and secure scheme for communications of smart meters and neighborhood gateways by utilization of an arm cortex-m microcontroller," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 6194–6205, 2018.

[35] D. Abbasinezhad-Mood and M. Nikooghadam, "Design and hardware implementation of a security-enhanced elliptic curve cryptography based lightweight authentication scheme for smart grid communications," *Future Generation Computer Systems*, vol. 84, pp. 47–57, 2018.

[36] Abbasinezhad-Mood, Dariush and Nikooghadam, Morteza, "An anonymous ecc-based self-certified key distribution scheme for the smart grid," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 10, pp. 7996–8004, 2018.

[37] State Cryptography Administration of China, "Public key cryptographic algorithm sm2 based on elliptic curves," http://www.oscca.gov.cn/UpFile/2010122214822692.pdf, 2010.

[38] M. Naor, S. Paz, and E. Ronen, "Crisp: Compromise resilient identity-based symmetric pake." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 529, 2020.

[39] C. Kudla and K. G. Paterson, "Modular security proofs for key agreement protocols," in *Advances in Cryptology - ASIACRYPT 2005*. Springer, 2005, pp. 549–565.

[40] R. Canetti, "Universally composable security: a new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001, pp. 136–145.

[41] T. A. Hall, "The fips 186-3 digital signature algorithm validation system (dsavs)," 2010.

[42] C. Cheng, Y. Qin, R. Lu, T. Jiang, and T. Takagi, "Batten down the hatches: Securing neighborhood area networks of smart grid in the quantum era," *IEEE Transactions on Smart Grid*, vol. 10, no. 6, pp. 6386–6395, 2019.

[43] J. Qian, Z. Cao, M. Lu, X. Chen, J. Shen, and J. Liu, "The secure lattice-based data aggregation scheme in residential networks for smart grid," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 2153–2164, 2022.

[44] S. Zhang, Y. Zhang, and B. Wang, "Antiquantum privacy protection scheme in advanced metering infrastructure of smart grid based on consortium blockchain and rlwe," *IEEE Systems Journal*, vol. 17, no. 2, pp. 3036–3046, 2023.

[45] K. Fan, Y. Ren, Y. Bai, G. Wei, K. Zhang, H. Li, and Y. Yang, "Fault-tolerant and collusion-resistant lattice-based multidimensional privacy-preserving data aggregation in edge-based smart grid," *IEEE Internet of Things Journal*, vol. 11, no. 6, pp. 9487–9504, 2024.

Ran Cheng Ran Cheng received her M.S. degree in Software Engineering from Taiyuan University of Technology in 2024. Her main research interests include cryptography and information security, certificateless public key cryptography and applied cryptography.

Yukun Yang Yukun Yang is currently pursuing the M.S. degree with College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology. His current research interests include public key cryptography, information security, and Internet of Things.

Zhengzhuo Zhang Zhengzhuo Zhang is currently pursuing the M.S. degree with College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology. His current research interests include public key cryptography, logical proof, and security analysis.

Xiaoxin Sun Xiaoxin Sun is currently pursuing the M.S. degree with College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology. Her current research interests include public key cryptography, information security, and Internet of Things.

Xin Huang Xin Huang received his Ph.D. degree from University of Oxford. He is currently a professor with Taiyuan University of Technology, China. His research interests include network security, blockchain, and Internet of Things.

Xiaohua Wu Xiaohua Wu received his Ph.D. degree from Univeristy of Chinese Academy of Science. He is currently the director of energy management department at Chinatowercom Co., Shanxi Branch. His research interests include low speed electric vehicle battery exchange, electric vehicle charging, electric bicycle charging and standby power supply.

Liangbin Zhao Liangbin Zhao received his Master's degree from Northeastern University. He is currently an engineer of energy management department at Chinatowercom Co., Shanxi Branch. His research interests include Charging and swapping technology for electric bicycles, BMS Battery Intelligent Management.