Journal of Southeast University (English Edition) Vol. 41, No. 3, pp. 392-400 Sept. 2025 ISSN 1003-7985

Quantum-resistant dynamic authenticated group key agreement scheme for the Internet of Things

JIANG Rui, XU Tengyu

(School of Cyber Science and Engineering, Southeast University, Nanjing 211189, China)

Abstract: With the recent advances in quantum computing, the key agreement algorithm based on traditional cryptography theory, which is applied to the Internet of Things (IoT) scenario, will no longer be secure due to the possibility of information leakage. In this paper, we propose a anti-quantum dynamic authenticated group key agreement scheme (AQDA-GKA) according to the ring-learning with errors (RLWE) problem, which is suitable for IoT environments. First, the proposed AQDA-GKA scheme can implement a group key agreement against quantum computing attacks by leveraging an RLWE-based key agreement mechanism. Second, this scheme can achieve dynamic node management, ensuring that any node can freely join or exit the current group. Third, we formally prove that the proposed scheme can resist quantum computing attacks as well as collusion attacks. Finally, the performance and security analysis reveals that the proposed AQDA-GKA scheme is secure and effective.

Key words: group key agreement; lattice-based cryptography; dynamic authentication; collusion attack resistance; Internet of Things

DOI: 10. 3969/j. issn. 1003-7985. 2025. 03. 015

Driven by the rapid progress and technological advances in the Internet of Things (IoT), the interconnection of smart devices has increased significantly in recent years. IoT devices often require sensitive data to be exchanged while operating in an open, dynamic, and potentially vulnerable environment. However, the inception of quantum computers has put traditional group key agreement algorithms at risk of being compromised. Furthermore, data security in the IoT environment is facing major challenges[1]. This has facilitated the need for further research on quantum computing attack resistance group key agreement (GKA) schemes for IoT.

GKA is a traditional cryptographic primitive[2]. Ever since the concept of key exchange was first introduced by Diffie-Hellman[3], there has been continuous research on key exchange in both academia and industry. Burmester et al. [4] proposed a two-round GKA protocol with the ability to overcome issues such as increased communication and computation costs with the rise in the number of communication rounds and participants. However, it could not incorporate authentication mechanisms, and its operation was based on the assumption of a trusted channel. Yang et al. [5] solved this issue by integrating digital signatures. In recent years, several studies have optimized GKA for specific applications. Abdussami et al. [6] introduced a provably secure and lightweight authenticated key agreement protocol for use in the modern healthcare industry. Cheng et al. [7] proposed a dynamic authentication GKA for 5G networks. Cao et al. [8] introduced a vehicular network GKA protocol, but it relied on a key generation center and lacked dynamic user departure support.

The introduction of Shor's algorithm [9] has compromised the security of traditional GKA schemes based on the discrete logarithm problem. Currently, lattice-based cryptosystems are considered the most viable for future cryptographic advancements. Ding et al. [10] proposed the first-ever lattice-based GKA protocol, which incorporated the concept of the DH protocol, and they constructed a quantum-resistant key exchange based on the LWE problem. Peikert [11] developed an efficient passive secure key exchange protocol according to the RLWE problem and introduced an error coordination mechanism. However, their approach was vulnerable to man-in-the-middle attacks. Bos et al. [12] proposed Frodo, an LWE-based post-quantum authenticated key agreement protocol with forward secrecy. However, this protocol incurred high computational and communication costs. Hougaard et al. [13] designed a tree-based RLWE GKA scheme to balance communication and memory efficiency but lacked dynamic user management. Choi et al. [14] designed an RLWE-based authenticated GKA but with linear communication complexity. Wang et al. [15] proposed a two-round LWE-based protocol with dynamic identity authentication, supporting user join but not removal. Cheng et al. [16] proposed an anonymous identity

Received 2025-01-08, **Revised** 2025-03-21.

Biography: Jiang Rui (1968—), male, doctor, professor, R. Jiang @seu. edu. cn.

Foundation items: The Project Supported by the National Engineering Research Center of Classified Protection and Safeguard Technology for Cybersecurity (No. C23640-XD-07), the Open Foundation of Key Laboratory of Cyberspace Security of Ministry of Education of China and Henan Key Laboratory of Network Cryptography (No. KLCS20240301).

Citation: JIANG Rui, XU Tengyu. Quantum-resistant dynamic authenticated group key agreement scheme for the Internet of Things[J]. Journal of Southeast University (English Edition), 2025, 41(3): 392-400. DOI: 10. 3969/j. issn. 1003-7985. 2025. 03. 015.

authentication and group key distribution scheme using quantum random numbers. To the best of the authors' knowledge, despite these advancements, achieving a balance between efficiency, dynamic user management, and post-quantum security is a persisting challenge.

1 Preliminaries

In this section, we introduce definitions based on some assumptions.

1.1 Lattice and Gaussian distribution

Definition 1 (lattice)[17] Let $\{\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n\}$ be a set of n linearly independent vectors, where $\boldsymbol{a}_i \in \mathbf{Z}^n$. The lattice Λ generated by the basis is $\Lambda = \mathcal{L}(\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_n) = \left\{\sum_{i=1}^{n} s_i \boldsymbol{a}_i | s_i \in \mathbf{Z}\right\}$.

Definition 2 (discrete Gaussian distribution)[17] For vectors $\boldsymbol{c} \in \mathbf{Z}^n$ and $\sigma \in R$, the Gaussian function is defined as $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{X}) = \exp(-\pi \| \boldsymbol{X} - \boldsymbol{c} \|^2 / \sigma^2)$. The discrete Gaussian distribution over a lattice Λ is defined as $D_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{X}) = \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{X}) / \sum_{\boldsymbol{X} \in \Lambda} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{X})$.

1.2 Ring learning with errors problem

Definition 3[18] Let $f(x) = x^n + 1$, where n is a power of 2. Let $R = \mathbf{Z}[x]/\langle f(x) \rangle$ be the ring of integer polynomials modulo $f(x)$, where the elements of R are the polynomials of degree less than n. Let $q = 1 \bmod 2n$, $R_q = \mathbf{Z}_q[x]/\langle f(x) \rangle$, where the polynomial coefficients are from $\{0, 1, \cdots, q-1\}$.

Definition 4 (ring learning with errors)[18] A sample from RLWE distribution, $A_{s,\Phi}$, is generated by randomly choosing $v \in R_q, e \in \Phi$, and outputting $(v, c = sv + e)$. The RLWE search problem denotes how to recover the secret value s using the equation.

Lemma 1 (hardness of R-LWE problem)[18] Let $\alpha \in (0, 1)$ be a real and q be a prime such that $\alpha q > 2\sqrt{n}$. Assuming that there exists an efficient algorithm that can solve the RLWE problem with Φ_α, there exists an efficient quantum algorithm for solving the worst case of SVP and CVP problems.

1.3 Generic key reconciliation mechanism

The key reconciliation mechanism comprises two algorithms—recMsg and recKey.

Definition 5 (recMsg and recKey)[11] Input $v \in R_q$, the modular rounding function is defined as $\lfloor \cdot \rceil_2$: $v \mapsto \lfloor 2v/q \rceil \bmod 2$, and the cross rounding function is defined as $\langle \cdot \rangle_2$: $v \mapsto \lfloor 4v/q \rceil \bmod 2$. recMsg$(v) \to (\text{rec}, k)$. Input $v \in R_q$, output $k = \lfloor v \rceil_2$, and signal is rec $= \langle v \rangle_2$.

Define $I_0 = \{0, 1, \cdots, \lfloor q/4 \rceil - 1\}$, $I_1 = \{-\lfloor q/4 \rceil, \cdots, -1\}$, and the set $E = \left[-\frac{q}{8}, \frac{q}{8}\right) \cap \mathbf{Z}$, recKey$(w, \text{rec})$ is defined as follows:

$$\text{recKey}(w, \text{rec}) := \begin{cases} 0 & \text{if } w \in I_{\text{rec}} + E \pmod q \\ 1 & \text{otherwise} \end{cases}$$

where $w \in R_q$, rec $\in R_q$.

Lemma 2[11] If $|v - w| < q/8$, then recKey $(w, \langle v \rangle_2) = \lfloor v \rceil_2$.

1.4 Lattice-based signature

Definition 6[19] The preimage sampleable trapdoor function is given by two algorithms—TrapGen and SamplePre.

The TrapGen$(q, 1^n)$ algorithm takes q, n as input and outputs $(g_i(\cdot), T_i)$, where $G_i \in \mathbf{Z}_q^{n \times m}$, $g_i(e) = G_i e \bmod q$, and T_i is a basis of $\Lambda^{\perp}(G_i)$.

The SamplePre(T_i, y) algorithm takes T_i and $y \in R_n$ as input and outputs $x = t + v$, where $x \in D_{\mathbf{Z}^m,s}$, $t \in \mathbf{Z}^m$, $G_i t \bmod q = y$, $v \in D_{\Lambda^{\perp}(G_i),s,-t}$.

Definition 7 (signature algorithm)[19] The signature algorithms primarily include SigKeyGen, Sign, and Verify algorithms.

The SignKeyGen$(q, 1^n)$ algorithm takes q, n as input and outputs $(g_i(\cdot), T_i)$. The public key is $\text{pk}_i = g_i(e)$, and the private key is $\text{sk}_i = T_i$.

The Sign$(\text{sk}_i, \text{msg})$ algorithm takes the private key sk_i and msg $\in \mathbf{Z}_q^n$ as input and outputs $\sigma_{\text{msg}} \in D_{\mathbf{Z}^m,s}$, where msg $= g_i(\sigma_{\text{msg}}) = G_i \sigma_{\text{msg}} \bmod q$.

The Verify$(\text{pk}_i, \text{msg}, \sigma_{\text{msg}})$ algorithm takes pk_i, msg $\in \mathbf{Z}_q^n$, $\sigma_{\text{msg}} \in D_{\mathbf{Z}^m,s}$ as input. Then, the algorithm calculates $g_i(\sigma_{\text{msg}}) = G_i \sigma_{\text{msg}} \bmod q$, if $\sigma_{\text{msg}} \in D_{\mathbf{Z}^m,s}$ and $g_i(\sigma_{\text{msg}}) = \text{msg}$, the signature verification is successful, else it outputs $\perp$.

2 Construction of AQDA-GKA Scheme

2.1 System model

The system model of the proposed AQDA-GKA scheme, illustrated in Fig. 1, consists of different groups of IoT devices. The IoT devices within the group are re-

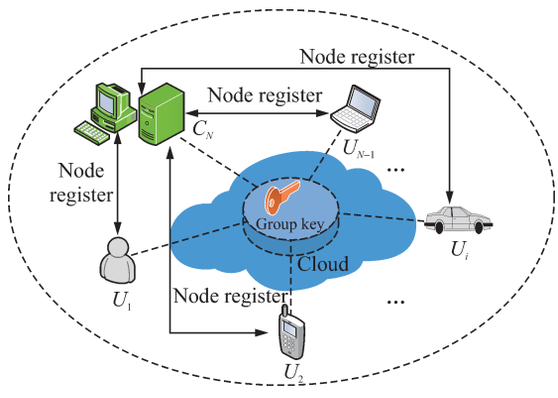

Fig. 1 System model of our AQDA-GKA scheme

ferred to as nodes. The controller node, which is denoted as C_N, possesses the highest computational capability. The other nodes are denoted as $\{U_1, U_2, \cdots, U_{N-1}\}$. All nodes negotiate the group session key.

2.2 Threat model

In the proposed AQDA-GKA scheme, let U be the set of participants. Any subset of U can establish a group key, and all nodes are semi-trusted nodes, which means that they will correctly perform all operations required in the scheme. However, they may passively collect and analyze the information they have access to.

Send. The adversary sends a message to an instance, which then processes it and returns a response.

Execute. The adversary passively eavesdrops on protocol execution and records the results.

Join$(\boldsymbol{G}, \boldsymbol{G}_1)$. The adversary observes the execution of the Join algorithm when a new group member $\boldsymbol{G}_1$ is added.

Exit$(\boldsymbol{G}, \boldsymbol{G}_2)$. The adversary observes the execution of the Exit algorithm when a member $\boldsymbol{G}_2$ leaves the group.

Corrupt(U_i). This query outputs the long-term key of U_i.

Test. This adversary attempts to distinguish between a genuine group key and a random key.

The adversary outputs a guess b'. Then, the advantage of adversary A against our AQDA-GKA scheme is defined as $\mathrm{Adv}_{A,\mathrm{P}} = |2\Pr[b' = b] - 1|$.

3 Proposed AQDA-GKA Scheme

3.1 System initialization

The system initialization phase involves three main steps—Setup, keyGen, and Register.

Setup(1^{λ}). In this step, the system generates the public parameters $\{q, n, l, \Phi_\alpha\}$. The system uses unique ID_i to represent the identity of node U_i, selects integers n and l, chooses a large prime q, and sets the Gaussian distribution $\Phi_\alpha, \alpha \in (0, 1)$ so that $\alpha q > 2\sqrt{n}$, where the number field is a ring $R_q = \mathbf{Z}_q[x]/\langle x^n + 1\rangle$. Furthermore, the system chooses four hash functions where $H_0{:}\{0,1\}^* \times R_q \times \{0,1\}^* \to \mathbf{Z}_q^n$, $H_1{:}\{0,1\}^* \times R_q \times \{0,1\}^l \times R_q \times \{0,1\}^* \to \mathbf{Z}_q^n$, $H_2{:}\{0,1\}^* \to \{0,1\}^l$, $H_3{:}\{0,1\}^* \to \mathbf{Z}_q^n$.

KeyGen(q, n). In this step, the group node executes the SignKeyGen$(q, 1^n)$ algorithm and outputs $(g_i(\cdot), T_i)$, where the node signature public key is $\mathrm{pk}_i = g_i(e)$, and the private key is $\mathrm{sk}_i = T_i$.

Register(ID_i, r_i). In this step, the group nodes select U_N as controller C_N, who possesses the highest computational capability. Then, each node randomly selects a string $r_i \in \{0,1\}^l$ and sends their ID_i along with r_i to C_N.

3.2 Group key generation

In this phase, the group nodes $\{U_i\}_{1 \leqslant i \leqslant N-1}$ and C_N execute the KeyAgreement algorithm and negotiate the group session key. The detailed process is illustrated in Fig. 2.

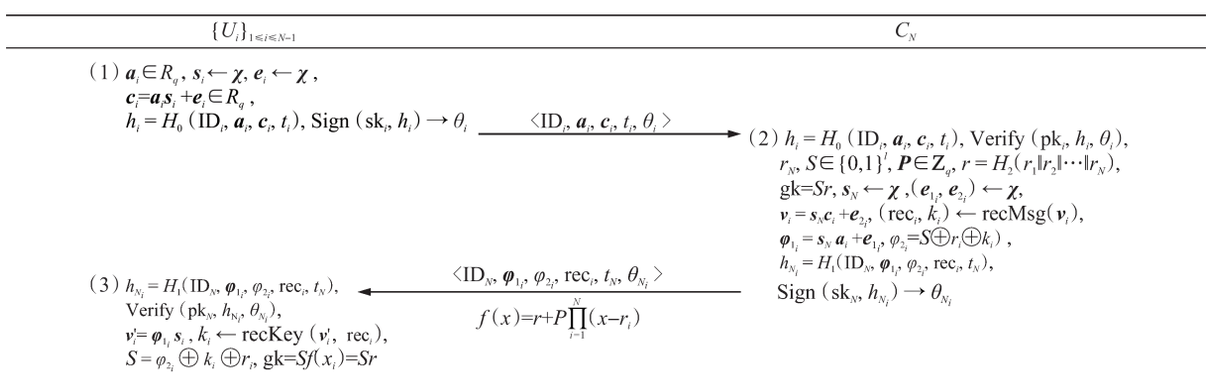

Fig. 2 KeyAgreement algorithm

In the first step, U_i randomly generates $\boldsymbol{a}_i \in R_q$, $\boldsymbol{e}_i \leftarrow \chi, \boldsymbol{s}_i \leftarrow \chi$ and calculates $\boldsymbol{c}_i = \boldsymbol{a}_i\boldsymbol{s}_i + \boldsymbol{e}_i$, where $\boldsymbol{s}_i$ serves as the node's long-term session key. Subsequently, U_i chooses a timestamp t_i, computes $h_i = H_0(\mathrm{ID}_i, \boldsymbol{a}_i, \boldsymbol{c}_i, t_i)$, and signs the h_i with their signature private key sk_i using the Sign(sk_i, h_i) algorithm to generate θ_t Finally, U_i broadcasts the information $\langle \mathrm{ID}_i, \boldsymbol{a}_i, \boldsymbol{c}_i, t_i, \theta_i \rangle$.

In the second step, upon receiving the message from U_i, C_N first verifies the correctness of the t_i and signature θ_i in the message. Further, C_N computes $h_i = H_0(\mathrm{ID}_i, \boldsymbol{a}_i, \boldsymbol{c}_i, t_i)$ and then runs the Verify$(\mathrm{pk}_i, h_i, \theta_i)$ algorithm with node signature public key pk_i. If the signature verification is successful, C_N accepts the message; otherwise, C_N outputs $\perp$.

Upon successful receipt of the message, C_N selects string $S, r_N \in \{0,1\}^l$ and calculates $r = H_2(r_1 \| r_2 \| \cdots \| r_N)$. Then, C_N calculates the group session key $\mathrm{gk} = Sr$. After that, C_N generates a column vector $\boldsymbol{s}_N \leftarrow \chi$, chooses $N - 1$ pairs of different vectors $(\boldsymbol{e}_{1_i}, \boldsymbol{e}_{2_i}) \leftarrow \chi$, and com-

putes $\boldsymbol{v}_i = \boldsymbol{s}_N\boldsymbol{c}_i + \boldsymbol{e}_{2_i}, \boldsymbol{\varphi}_{1_i} = \boldsymbol{s}_N\boldsymbol{a}_i + \boldsymbol{e}_{1_i}$. Then, C_N runs the general key agreement algorithm $\mathrm{recMsg}(\boldsymbol{v}_i)$ to calculate the signal $\mathrm{rec}_i = \langle \boldsymbol{v}_i \rangle_2$, $k_i = \lfloor \boldsymbol{v}_i \rceil_2$. Finally, C_N encrypts group session key material S with $\varphi_{2_i} = S \oplus r_i \oplus k_i$, where r_i is the secret value registered by U_i in the Register phase.

Finally, C_N selects the timestamp t_N, computes $h_{N_i} = H_1(\mathrm{ID}_N, \boldsymbol{\varphi}_{1_i}, \varphi_{2_i}, \mathrm{rec}_i, t_N)$, and runs the sign algorithm $\mathrm{Sign}(\mathrm{sk}_N, h_{N_i})$ to generate θ_{N_i}. Further, C_N designs a group key management function $f(x) = r + P\prod_{i=1}^{N}(x - r_i)$, where $P \in \mathbf{Z}_q$. Subsequently, C_N sends the message $\langle \mathrm{ID}_N, \boldsymbol{\varphi}_{1_i}, \varphi_{2_i}, \mathrm{rec}_i, t_N, \theta_{N_i} \rangle$ and broadcasts $f(x)$.

In the third step, U_i first verifies the correctness of t_N and θ_{N_i} in the message. Then, U_i calculates $h_{N_i} = H_1(\mathrm{ID}_N, \boldsymbol{\varphi}_{1_i}, \varphi_{2_i}, \mathrm{rec}_i, t_N)$ and runs the $\mathrm{Verify}(\mathrm{pk}_N, h_{N_i}, \theta_{N_i})$ algorithm with the node signature public key pk_N. If $\theta_{N_i} \in D_{\mathbf{Z}^m, s}$ and $g_N(\theta_{N_i}) = h_{N_i}$, U_i accepts the message, otherwise, it outputs $\perp$.

To recover the group session key material S from φ_{2_i}, U_i calculates $\boldsymbol{v}'_i = \boldsymbol{\varphi}_{1_i}\boldsymbol{s}_i$, where $\boldsymbol{s}_i$ is the node's long-term session key. Then, U_i calculate $k_i = \mathrm{recKey}(\boldsymbol{v}'_i, \mathrm{rec}_i)$ according to Definition 5 and finally recovers $S = \varphi_{2_i} \oplus k_i \oplus r_i$, where r_i is the secret value registered by U_i in the Register phase.

Finally, U_i evaluates the group session key $\mathrm{gk} = Sf(x_i) = S\left[r + P\prod_{i=1}^{N}(x_i - r_i)\right] = Sr, x_i = r_i$.

3.3 Efficient dynamic group key management

In the group key management phase, we designed two algorithms—Join and Exit.

3.3.1 Nodes dynamical join

When a group of nodes $\boldsymbol{G}_1$ wants to join group $\boldsymbol{G}$ to form a new group $\boldsymbol{G}'$, the system will execute the Join algorithm. A detailed explanation of this process is illustrated in Fig. 3.

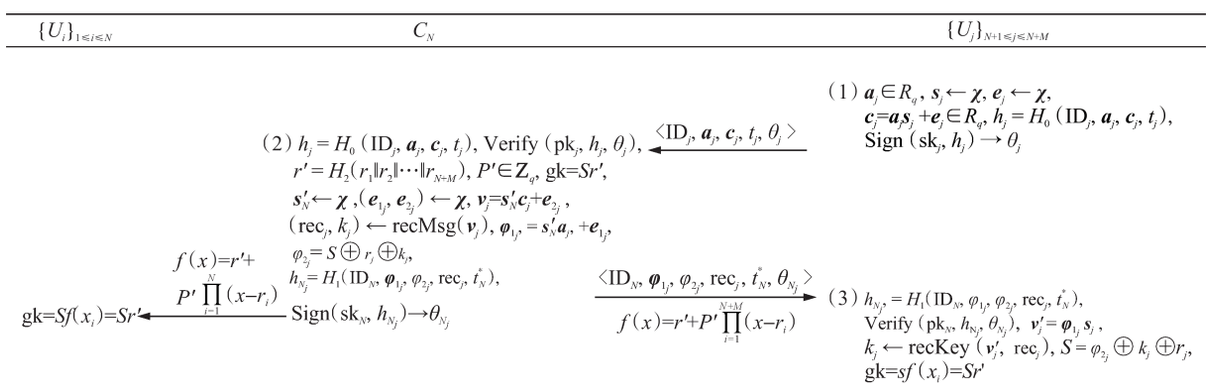

Fig. 3 Join algorithm

$\mathrm{Join}(\boldsymbol{s}, \boldsymbol{G}, \boldsymbol{G}_1)$. Assuming that there is a group of nodes $\boldsymbol{G}_1\{U_j\}_{N+1 \le j \le N+M}$ that wish to join an existing session group $\boldsymbol{G}\{U_i\}_{1 \le i \le N}$, they need to execute the Register and Join algorithms. This process forms a new group $\boldsymbol{G}' = \boldsymbol{G} \cup \boldsymbol{G}_1$ through a round of information exchange, obtaining a new session key gk′, with the main computational resources still mainly provided by node C_N.

In the first step, nodes $\{U_j\}_{N+1 \le j \le N+M}$, which wish to join group $\{U_i\}_{1 \le i \le N}$, generate their own long-term session key $\boldsymbol{s}_j$ and calculate $\boldsymbol{c}_j = \boldsymbol{a}_j\boldsymbol{s}_j + \boldsymbol{e}_j$, where $\boldsymbol{a}_j \in R_q, \boldsymbol{e}_j \leftarrow \chi$. Then, U_j selects t_j and computes $h_j = H_0(\mathrm{ID}_j, \boldsymbol{a}_j, \boldsymbol{c}_j, t_j)$. Finally, U_j signs the h_j with signature private key sk_j to generate θ_j and broadcasts the information $\langle \mathrm{ID}_j, \boldsymbol{a}_j, \boldsymbol{c}_j, t_j, \theta_j \rangle$.

In the second step, C_N first verifies the timestamp and signature, and it then recalculates the group session key $\mathrm{gk} = Sr'$, $r' = H_2(r_1 \| r_2 \| \cdots \| r_{N+M})$. After that, C_N generates a column vector $\boldsymbol{s}'_N$, chooses M pairs of different vectors $(\boldsymbol{e}_{1_j}, \boldsymbol{e}_{2_j})$, and computes $\boldsymbol{v}_j = \boldsymbol{s}'_N\boldsymbol{c}_j + \boldsymbol{e}_{2_j}, \boldsymbol{\varphi}_{1_j} = \boldsymbol{s}'_N\boldsymbol{a}_j + \boldsymbol{e}_{1_j}$. Then, C_N runs the $\mathrm{recMsg}(\boldsymbol{v}_j)$ algorithm to calculate $\mathrm{rec}_j = \langle \boldsymbol{v}_j \rangle_2$, $k_j = \lfloor \boldsymbol{v}_j \rceil_2$ and encrypts S with $\varphi_{2_j} = S \oplus r_j \oplus k_j$, where r_j is the secret value registered by U_j in the Register phase. Subsequently, C_N chooses the timestamp, computes $h_{N_j} = H_1(\mathrm{ID}_N, \varphi_{1_j}, \varphi_{2_j}, \mathrm{rec}_j, t^*_N)$, and runs the sign algorithm $\mathrm{Sign}(\mathrm{sk}_N, h_{N_j})$ to generate θ_{N_j}. Then, C_N redesigns the group key management function as $f(x) = r' + P'\prod_{i=1}^{N+M}(x - r_i)$. Finally, C_N sends $\langle \mathrm{ID}_N, \varphi_{1_j}, \varphi_{2_j}, \mathrm{rec}_j, t^*_N, \theta_{N_j} \rangle$ and $f(x)$ to nodes.

In the third step, U_j verifies the timestamp and signature first. Then, it calculates $\boldsymbol{v}'_j = \boldsymbol{\varphi}_{1_j}\boldsymbol{s}_j, k_j = \mathrm{recKey}(\boldsymbol{v}'_j, \mathrm{rec}_j)$ and recovers $S = \varphi_{2_j} \oplus k_j \oplus r_j$. Finally,

nodes $\{U_i\}_{1 \leqslant i \leqslant N+M}$ apply the group key management function $f(x)$ to evaluate the group session key $\mathrm{gk}' = Sf(x_i) = S\left[r' + P' \prod_{i=1}^{M+N}(x_i - r_i)\right] = Sr', x_i = r_i$.

3.3.2 Nodes dynamical exit

When a group of nodes $\boldsymbol{G}_2$ wants to exit group $\boldsymbol{G}$ and the remaining nodes form a new group $\boldsymbol{G}' = G/G_2$, the system will execute the Exit algorithm. A detailed explanation of this process is given in Fig. 4.

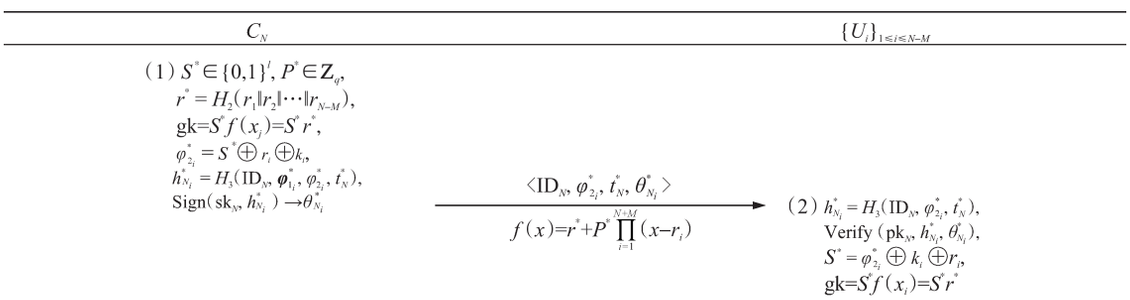

Fig. 4 Exit algorithm

Exit$(\boldsymbol{s}, \boldsymbol{G}, \boldsymbol{G}_2)$. Assuming that there are some group members $\boldsymbol{G}_2\{U_m\}_{1 \leqslant m \leqslant M}$ within the current group $\boldsymbol{G}\{U_i\}_{1 \leqslant i \leqslant N}$ who wish to leave the group, where $M < N$. The remaining $N - M$ nodes execute the Exit algorithm to form a new group key for $\boldsymbol{G}'$.

In the first step, C_N re-selects S^*, P^* and calculates $r^* = H_2(r_1 \| r_2 \| \cdots \| r_{N-M}) \in \{0,1\}^l$, which is the new group session key $\mathrm{gk} = S^* r^*$. Subsequently, C_N encrypts the new group session key material S^* with $\varphi^*_{2_i} = S^* \oplus r_i \oplus k_i$, where r_i is the secret value registered by the U_i in the Register phase, and k_i is the value calculated by C_N in the KeyAgreement. Then, C_N selects the timestamp t^*_N, computes $h^*_{N_i} = H_3(\mathrm{ID}_N, \varphi^*_{2_i}, t^*_N)$, and runs the $\mathrm{Sign}(\mathrm{sk}_N, h^*_{N_i})$ to generate $\theta^*_{N_i}$. Finally, C_N sends the message $\langle \mathrm{ID}_N, \varphi^*_{2_i}, t^*_N, \theta^*_{N_i} \rangle$ and $f(x) = r^* + P^* \prod_{i=1}^{N-M}(x - r_i)$ to node U_i.

In the second step, upon receiving the broadcasted message from C_N, U_i first verifies the correctness of the timestamp t^*_N and checks the signature $\theta^*_{N_i}$ in the broadcast message. U_i computes $h^*_{N_i} = H_3(\mathrm{ID}_N, \varphi^*_{2_i}, t^*_N)$ and then runs the $\mathrm{Verify}(\mathrm{pk}_N, h^*_{N_i}, \theta^*_{N_i})$ algorithm with the node signature public key pk_N.

Upon successful receipt of the message, U_i recovers the new group session key material $S^* = \varphi^*_{2_i} \oplus k_i \oplus r_i$. Finally, U_i applies the function $f(x)$ to evaluate the group session key $\mathrm{gk} = S^* f(x_i) = S^*\left[r^* + P^* \prod_{i=1}^{N-M}(x - r_i)\right] = S^* r^*$, where $x_i = r_i$.

3.4 Correctness

Theorem 1 The proposed AQDA-GKA scheme is correct.

Proof In the group key generation phase, C_N runs the $\mathrm{recMsg}(\boldsymbol{v}_i)$ algorithm to calculate signal $\mathrm{rec}_i = \langle \boldsymbol{v}_i \rangle_2$, $k_i = \lfloor \boldsymbol{v}_i \rceil_2$ and encrypts S with $\varphi_{2_i} = S \oplus r_i \oplus k_i$. Then, C_N can correctly evaluate the group session key $\mathrm{gk} = Sr$, where $r = H_2(r_1 \| r_2 \| \cdots \| r_N)$. Furthermore, C_N designs function $f(x) = r + P \prod_{i=1}^{N}(x - r_i)$ and sends the message $\langle \mathrm{ID}_N, \varphi_{1_i}, \varphi_{2_i}, \mathrm{rec}_i, t_N, \theta_{N_i} \rangle$ to U_i.

Then, U_i calculates $\boldsymbol{v}'_i = \boldsymbol{\varphi}_{1_i} \boldsymbol{s}_i$, obtaining $k'_i = \mathrm{recKey}(\boldsymbol{v}'_i, \mathrm{rec}_i)$. According to Lemma 2, $k'_i = \mathrm{recKey}(\boldsymbol{v}'_i, \langle \boldsymbol{v}_i \rangle_2) = \lfloor \boldsymbol{v}_i \rceil_2 = k_i$. Therefore, U_i can correctly recover the group session key material S using $S = \varphi_{2_i} \oplus k_i \oplus r_i$ evaluate group session key $\mathrm{gk} = Sf(x_i) = Sr$, where $x_i = r_i$.

Theorem 2 The proposed AQDA-GKA scheme can correctly perform the dynamic management of group nodes.

Proof In the node join phase, the group manager C_N redesigns the group key management function $f(x) = r' + P' \prod_{i=1}^{N+M}(x - r_i)$. Therefore, the node U_i can evaluate the new group session key $\mathrm{gk} = Sf(x_i) = Sr'$, where $x_i = r_i$.

In the node exit phase, C_N redesigns the function $f(x) = r^* + P^* \prod_{i=1}^{N-M}(x - r_i)$. Then, C_N encrypts S^* with $\varphi^*_{2_i} = S^* \oplus r_i \oplus k_i$ and signs $h^*_{N_i} = H_3(\mathrm{ID}_N, \varphi^*_{2_i}, t^*_N)$ to generate $\theta^*_{N_i}$. Finally, C_N sends $\langle \mathrm{ID}_N, \varphi^*_{2_i}, t^*_N, \theta^*_{N_i} \rangle$ to U_i.

After U_i accepts the message, it recovers the group session key material S^* using $S^* = \varphi^*_{2_i} \oplus k_i \oplus r_i$ and evaluates the group session key $\mathrm{gk} = S^* f(x_i) = S^* r^*$, where $x_i = r_i$.

4 Security Analysis

4.1 Formal proof

In this section, we formally prove that the proposed

AQDA-GKA scheme can resist quantum computing and collusion attacks, ensuring secure dynamic node management.

Theorem 3 The proposed AQDA-GKA scheme can resist quantum computational attacks.

Proof We demonstrated the number field in the core key agreement algorithm $(\boldsymbol{a}_i, \boldsymbol{c}_i)$, where $\boldsymbol{c}_i = \boldsymbol{a}_i \boldsymbol{s}_i + \boldsymbol{e}_i, \boldsymbol{a}_i \in R_q, \boldsymbol{s}_i, \boldsymbol{e}_i \leftarrow \Phi_\alpha$, due to the closure property of ring computations, $\boldsymbol{c}_i \in R_q$. Then, we can prove that the expression $\boldsymbol{c}_i = \boldsymbol{a}_i \boldsymbol{s}_i + \boldsymbol{e}_i$ aligns with the construction of the RLWE problem. According to Lemma 1, the RLWE problem has been proven to resist quantum computational attacks, which indicates that quantum computing attacks cannot decrypt or recover $\boldsymbol{s}_i$.

Theorem 4 The proposed AQDA-GKA scheme can securely perform dynamic group management, including node join and node exit.

Proof First, we prove that the AQDA-GKA scheme can ensure the secure dynamic joining of nodes. As to legitimate nodes within the group, because $x_i = r_i, \forall i \in [1, M+N]$, there are

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_{M+N} x_i{}^{M+N} = \sum_{i=0}^{M+N} a_i x_i^i =$$

$$P' \prod_{j=1}^{M+N} (x_i - r_j) = P' \prod_{i \neq j}^{M+N} (x_i - r_j)(x_i - r_i) = 0$$

For all other illegal nodes, the polynomial group key management function $f(x) \neq r'$. Therefore, only the legitimate nodes can evaluate the correct new session key because

$$\mathrm{gk}' = Sf(x_i) = S\left[r' + P' \prod_{j \neq i}^{M+N} (x_i - r_j)(x_i - r_i)\right] = S(r' + 0) = Sr'$$

Second, we prove the AQDA-GKA scheme can ensure the secure dynamic exit of nodes.

Theorem 5 The proposed AQDA-GKA scheme can resist the collusion attack.

Proof First, the AQDA-GKA scheme can resist collusion attacks between legitimate nodes and revoked nodes. Further, each legitimate node can decrypt the group session key material according to the specific node identity r_i. The other nodes cannot access the value of r_i for other devices. Thus, the revoked node cannot obtain the updated group session key materials.

Next, the AQDA-GKA scheme can resist collusion attacks between the revoked nodes. The revoked node cannot obtain the corresponding legitimate node's identifier r_i or the secret value k_i. Therefore, the revoked node cannot retrieve the updated session key.

Finally, the AQDA-GKA scheme can resist collusion attacks between revoked nodes and network attackers. As demonstrated previously, network attackers lack key information and cannot assist the revoked node in computing the new session key.

4.2 Security comparison

In this section, we compared the proposed AQDA-GKA scheme with other schemes [13-15] in terms of node secure dynamic join, node secure dynamic exit, authentication, anti-quantum computing attack, and resist collusion attack, as shown in Table 1.

Table 1 Security comparison

Property	Ref. [13]	Ref. [14]	Ref. [15]	AQDA-GKA
Secure dynamic join	×	√	√	√
Secure dynamic exit	×	×	×	√
Authentication	×	√	√	√
Anti-quantum computing attack	√	√	√	√
Resist collusion attack	×	√	×	√

Note: √ denotes schemes that meet the corresponding functions, while × means the opposite.

5 Performance Analysis

The performance environment of the proposed AQDA-GKA scheme is shown in Fig. 1, where the group key negotiation participants are deployed on Alibaba Cloud servers. The C_N configuration is CentOS 8.2, 64-bit, with a 4-core CPU and 8 GB of RAM, while ordinary nodes are situated on devices equipped with an Intel Core i5-8265U CPU and 8 GB of RAM, running CentOS7, 64-bit virtual machines.

5.1 Computation cost

In this section, we primarily analyze the computation cost of our AQDA-GKA scheme with related schemes [13-15].

5.1.1 Computation cost for one ordinary node in the group key agreement process

In this section, we compare the computation cost for one ordinary node in the group key agreement process between the AQDA-GKA scheme and other schemes [13-15]. The computation cost primarily depends on the Key-Agreement algorithm in the key agreement process. We apply multiplication to represent the computation cost of each scheme, where T_{mul} represents one multiplication on R_q, t_{mul} represents one multiplication on $\mathbf{Z}_q$, T_{H} represents hash operations, n_{R} denotes the ring size of the

RLWE problem, n_L denotes the dimension of the LWE problem, and N represents the number of participating nodes. To formalize the calculation, we define $T_{mul} = (n_R \log n_R)t_{mul}$ and $T_H = n_R t_{mul}$. The total computation cost generated for one ordinary node to complete a key agreement process is shown in Table 2.

Table 2 Computation cost for one ordinary node in the group key agreement process

Scheme	Computation cost
Ref.[13]	$2T_{mul}$
Ref.[14]	$3T_{mul} + (N+4)n_R t_{mul}$
Ref.[15]	$6n_L t_{mul} + (n_L n_L n' n_L)t_{mul} + (n' n_L n_L)t_{mul}$
AQDA-GKA	$2T_{mul} + 4n_R t_{mul}$

According to Table 2, we set parameters $n_L = 512, n_R = 1\,024, n' = 32$ and plot $\log_{10} t_{mul}$ on the y-axis; the comparison results are illustrated in Fig. 5.

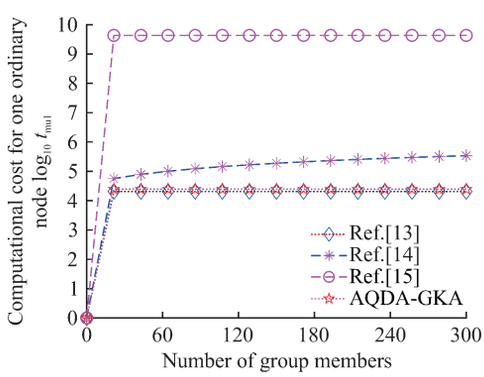

Fig. 5 Computation cost for one ordinary node in the group key agreement process

As shown in Fig. 5, the computation cost for one ordinary node in the key agreement process for the proposed AQDA-GKA scheme ($24\,576\,t_{mul}$) is slightly greater than that for the scheme in Ref. [13] ($20\,480\,t_{mul}$), according to $2T_{mul} + 4n_R t_{mul}$ and $2T_{mul}$, respectively. However, the proposed scheme supports dynamic management of group nodes, identity authentication, and resist collusion attacks, while the scheme [13] cannot. The computational cost for one ordinary node in this scheme ($24\,576\,t_{mul}$) is less than that in Ref. [14] ($1\,024(N+34)t_{mul}$) and Ref. [15] ($4.3 \times 10^9 t_{mul}$), according to $3T_{mul} + (N+4)n_R t_{mul}$ and $6n_L t_{mul} + (n_L n_L n' n_L)t_{mul} + (n' n_L n_L)t_{mul}$, respectively.

5.1.2 Computation cost for all ordinary nodes in the process of dynamic management

In this section, we compare the computation cost of all ordinary nodes in the proposed AQDA-GKA scheme and other schemes [13-15] during dynamic management.

(1) Nodes dynamical join

The total computation cost of ordinary nodes during dynamic join in the proposed AQDA-GKA scheme and other schemes [13-15] is shown in Table 3.

Table 3 Computation cost for all ordinary nodes in the process of node joining

Scheme	Computation cost
Ref.[13]	∞
Ref.[14]	$(N-3)(m+2)n_R t_{mul} + (N-3)T_{mul}$
Ref.[15]	$2(N-1)(m+1)n_L t_{mul}$
AQDA-GKA	$(N-1)t_{mul}$

According to Table 3, we set parameters $N = 50$ and use m to represent the number of people who will join the group. We plot $\log_{10} t_{mul}$ on the y-axis and illustrate the comparison results in Fig. 6.

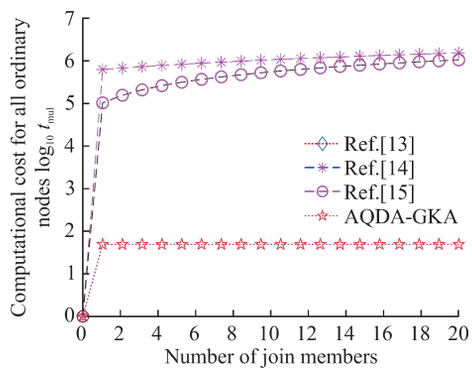

Fig. 6 Computation cost for all ordinary nodes in the process of joining

The computation cost for all ordinary nodes during dynamic join in the proposed scheme is significantly less than that in Ref. [13]. Because the scheme [13] cannot realize dynamic join, the computation cost is to be infinite ∞. As shown in Fig. 6, the computation cost for all ordinary nodes in the process of dynamic join in the proposed scheme ($49t_{mul}$) is less than that in Ref. [14] ($48\,128(m+12)t_{mul}$) and Ref. [15] ($50\,176(m+1)t_{mul}$), according to $(N-3)(m+2)n_R t_{mul} + (N-3)T_{mul}$ and $2(N-1)(m+1)n_L t_{mul}$, respectively.

(2) Nodes dynamical exit

In the process of nodes dynamic exit, the total computation cost for the proposed scheme and other schemes [13-15] is shown in Table 4.

Table 4 Computation cost for all ordinary nodes in the process of node exit

Scheme	Computation cost
Ref.[13]	∞
Ref.[14]	$(3T_{mul} + (N+4)n_R t_{mul})(N-m)$
Ref.[15]	$(6n_L t_{mul} + (n_L+1)(n' n_L)n_L t_{mul})(N-m)$
AQDA-GKA	$(2n_R+1)(N-m)t_{mul}$

According to Table 4, we set the number of participating nodes $N = 50$ and use m to represent the number of nodes that will exit the group. We plot $\log_{10} t_{mul}$ on the y-axis and illustrate the comparison results in Fig. 7.

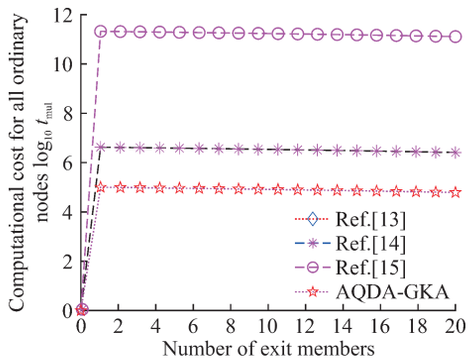

Fig. 7 Computation cost for all ordinary nodes in the process of node exit

The computation cost for all ordinary nodes during dynamic exit in the proposed scheme is far less than that for the scheme in Ref. [13]. Because the scheme[13] cannot realize dynamic group management, the computation cost is infinite. As shown in Fig. 7, we can see that the computation cost for all ordinary nodes in the process of dynamic exit for the proposed scheme ($(50-m)t_{mul}$) is less than that in the scheme in Ref. [14] ($86\,016(50-m)t_{mul}$) and Ref. [15] ($4.3\times10^9\times(50-m)t_{mul}$), according to $(3T_{mul}+(N+4)n_R t_{mul})(N-m)$ and $(6n_L t_{mul}+(n_L+1)(n'n_L)n_L t_{mul})(N-m)$, respectively.

5.2 Communication cost

In this section, we analyze the communication cost of the proposed scheme with related schemes[13-15]. We set l_s represents the length of the signature, l_r represents the length of the random string, l_t represents the length of the timestamp, l_{ID} represents the length of node identity information, and N represents the number of participating nodes. Here, q_R and q_L represent the lengths of the modulus q for RLWE and LWE, respectively.

The communication cost for the proposed scheme and other schemes[13-15] in the process of completing the key agreement is shown in Table 5.

Table 5 Communication costs of different protocols

Scheme	Communication cost
Ref. [13]	$3n_R(\log q_R+1)N$
Ref. [14]	$2N(l_{ID}+1+n_R\log q_R+l_S)+n_R$
Ref. [15]	$N((n_L+n'+1)n_L\log q_L+n')+2N(l_{ID}+l_r+l_t+l_S)$
AQDA-GKA	$N(2(l_{ID}+l_t+l_S)+3n_R\log q_R+l_r+n_R)$

According to Table 5, we set parameters $l_s=160$, $l_r=160$, $l_t=32$, $l_{ID}=128$, $q_L=2^{12}$, $q_R=2^{32}$ and plot $\log_{10}L$ on the y-axis, where L represents the bit length of information in the communication process. The comparison results are illustrated in Fig. 8.

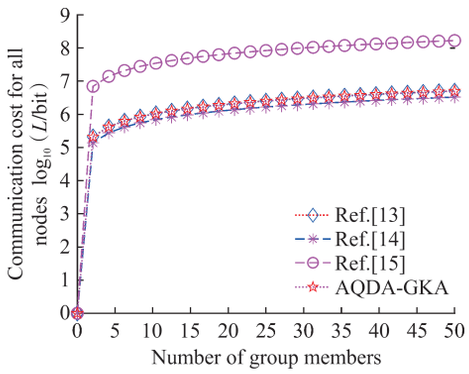

Fig. 8 Communication costs for different schemes

As shown in Fig. 8, the communication cost for the proposed scheme is slightly less than that for the schemes in Refs. [13, 15], and it is larger than that for the scheme in Ref. [14]. However, the proposed scheme can realize dynamic management of group nodes, while the scheme in Ref. [14] cannot.

6 Conclusions

In this paper, we propose a quantum-resistant dynamic authenticated group key agreement scheme. Our scheme ensures secure dynamic management, resists quantum attacks and prevents collusion. We integrate a signature algorithm to realize the identity authentication of the group nodes. We formally prove that our scheme can resist quantum computing attacks as well as collusion attacks. Finally, the performance analysis shows that the proposed scheme is more efficient than other related schemes.

References

[1] ZHANG H, CHEN L Q, YANG B, et al. Secure lightweight data using scheme in 5G industrial Internet systems[J]. Journal of Southeast University (Natural Science Edition), 2024, 54(3): 772-780. (in Chinese)

[2] ALWEN J, MULARCZYK M, TSELEKOUNIS Y. Fork-resilient continuous group key agreement[M]//Advances in Cryptology—CRYPTO 2023. Cham: Springer Nature Switzerland, 2023: 396-429.

[3] DIFFIE W, HELLMAN M. New directions in cryptography[J]. IEEE Transactions on Information Theory, 1976, 22(6): 644-654.

[4] BURMESTER M, DESMEDT Y. A secure and efficient conference key distribution system: Extended abstract[M]//Advances in Cryptology—EUROCRYPT'94. Berlin, Germany: Springer Berlin Heidelberg, 1995: 275-286.

[5] YANG Z Y, WANG Z Q, QIU F, et al. A group key agreement protocol based on ECDH and short signature[J]. Journal of Information Security and Applications, 2023, 72: 103388.

[6] ABDUSSAMI M, AMIN R, VOLLALA S. Provably secured lightweight authenticated key agreement protocol

for modern health industry [J]. Ad Hoc Networks, 2023, 141: 103094.
[7] CHENG X B, JIANG R, PEI B, et al. Dynamic group authentication and key agreement protocol for D2D secure communication in 5G networks [J]. Journal of Southeast University (Natural Science Edition), 2020, 50(5): 918-928. (in Chinese)
[8] CAO X F, DANG L J, FAN K, et al. A dynamic and efficient self-certified authenticated group key agreement protocol for VANET [J]. IEEE Internet of Things Journal, 2024, 11(17): 29146-29156.
[9] SHOR P W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer [J]. SIAM Review, 1999, 41(2): 303-332.
[10] DING J T, XIE X, LIN X D. A simple provably secure key exchange scheme based on the learning with errors problem [J/OL]. Cryptology ePrint Archive, 2012 [2024-10-15]. https://eprint. iacr. org/2012/688. pdf.
[11] PEIKERT C. Lattice cryptography for the internet [C]// International Workshop on Post-Quantum Cryptography. Cham: Springer International Publishing, 2014: 197-219.
[12] BOS J, COSTELLO C, DUCAS L, et al. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE [C]//Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. Vienna, Austria, 2016: 1006-1018.
[13] HOUGAARD H B, MIYAJI A. Tree-based ring-LWE group key exchanges with logarithmic complexity [C]// 22nd International Conference on Information and Communications Security. Cham: Springer International Publishing, 2020: 91-106.
[14] CHOI R, HONG D, HAN S, et al. Design and implementation of constant-round dynamic group key exchange from RLWE [J]. IEEE Access, 2020, 8: 94610-94630.
[15] WANG Z Q, YANG Z Y, LI F G. A two rounds dynamic authenticated group key agreement protocol based on LWE [J]. Journal of Systems Architecture, 2022, 133: 102756.
[16] CHENG T, LIU Q, SHI Q, et al. Efficient anonymous authentication and group key distribution scheme based on quantum random numbers for VANETs [J]. IEEE Internet of Things Journal, 2024, 11(13): 23544-23560.
[17] REGEV O. On lattices, learning with errors, random linear codes, and cryptography [J]. Journal of the ACM, 2009, 56(6): 1-40.
[18] LYUBASHEVSKY V, PEIKERT C, REGEV O. On ideal lattices and learning with errors over rings [C]//Advances in Cryptology—EUROCRYPT 2010. Berlin, Germany: Springer Berlin Heidelberg, 2010: 1-23.
[19] GENTRY C, PEIKERT C, VAIKUNTANATHAN V. Trapdoors for hard lattices and new cryptographic constructions [C]//Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. Victoria, British Columbia, Canada, 2008: 197-206.

抗量子计算攻击的物联网动态群组认证密钥协商方案

蒋睿, 徐腾昱

(东南大学网络空间安全学院,南京211189)

摘要: 随着量子计算技术的快速发展,基于传统密码学理论在物联网场景中的密钥协商算法将面临安全性威胁,存在信息泄露的风险。本文提出了一种适用于物联网环境可抵抗量子计算攻击的动态认证群组密钥协商方案(AQDA-GKA)。该方案基于环上误差学习难题构建密钥协商机制,以抵抗量子计算攻击。该方案支持节点动态管理,使得任意节点能够自由加入或退出现有群组。此外,形式化证明了所提出的方案能够抵抗量子计算攻击,并且能够有效防御共谋攻击。最后,通过对方案性能与安全性进行分析,证明该方案在确保安全性的同时具备较高的计算效率。

关键词: 群组密钥协商;格密码学;动态身份认证;抵抗共谋攻击;物联网

中图分类号: TN918.4