LBAS: A Batch Authentication Scheme for M2M Scenarios

Zhihuan Xing[1], Yongzhuo Huang[2], Yuqing Lan[3], Xiaoyi Yang[3], Yichun Yu[1], Xiaoxue Cui[3,4] and Dan Yu[4]

[1] School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[2] School of Cyber Science and Technology, Beihang University, Beijing 100191, China
[3]School of Software, Beihang University, Beijing 100191, China
lanyuqing@buaa.edu.cn
[4]China Standard Intelligent Security Co., Ltd., Beijing 100097, China

Abstract. In some IoT (Internet of Things) Machine-to-Machine (M2M) scenarios, existing device authentication schemes often struggle to balance security, lightweight design, and flexibility, making them unsuitable for batch authentication among low-performance devices. Therefore, this paper proposes a Lightweight Batch Authentication Scheme (LBAS) tailored for M2M scenarios involving low-performance devices. LBAS employs a two-step authentication strategy to ensure both security and flexibility. It organizes devices into a complete binary tree structure, reducing the number of mutual authentications between devices and achieving lightweight authentication. Additionally, LBAS provides a comprehensive trust domain management strategy to handle different behaviors of devices within the trust domain, ensuring availability. We formalized the correctness of the LBAS authentication phase using BAN (Burrows–Abadi–Needham) logic and analyzed two common types of man-in-the-middle attacks, demonstrating the protocol's security. Compared to traditional publish-subscribe schemes, LBAS does not require a proxy server and reduces average time overhead by 44.3%.

Keywords: IoT, M2M, Security, Batch Authentication.

1 Introduction

The Internet of Things (IoT) is transforming our world at an unprecedented pace, primarily by enabling various devices to interconnect, facilitating intelligent management and control. However, in IoT systems, device communication typically requires a central base station for relaying messages, which can decrease communication efficiency and performance and may render the system vulnerable to attacks [1]. To address these issues, Machine-to-Machine (M2M) technology has gradually gained attention in both academia and industry. M2M technology represents a significant development direction for future wireless communication networks, allowing devices to communicate directly without the need for a central base station. Existing M2M technologies include cellular-based M2M communication, Wi-Fi direct communication, Bluetooth Low

Energy (BLE) direct communication, among others. Currently, these technologies are widely applied in IoT scenarios such as smart homes, intelligent transportation, and smart security. Research indicates[2] that this mode of communication holds substantial potential for enhancing communication reliability and security, while reducing latency and energy consumption.

While M2M communication technology brings these advantages, it also introduces numerous security and performance challenges. For instance, attackers can exploit eavesdropping, replay attacks, man-in-the-middle attacks, and other methods to compromise M2M communications, thereby stealing user privacy information, forging user identities, and disrupting communication services. Moreover, in many IoT scenarios, devices are often characterized by limited resources, low performance, and large quantities, making them incapable of supporting complex authentication strategies. Thus, establishing secure communication links between low-performance M2M devices without compromising user data becomes a fundamental requirement for M2M communication.

To address the security and performance issues in M2M communication, this paper proposes a Lightweight Batch Authentication Scheme (LBAS) for IoT device M2M communication. This scheme enables trust relationships between devices and significantly reduces the number of authentications during batch device mutual authentication, thereby minimizing performance consumption while ensuring security. Specifically, the main contributions of this paper are summarized as follows:

1. We design a novel authentication scheme, LBAS, to tackle the problem of batch authentication for low-performance devices in M2M scenarios.
2. We manage trust domain devices using a tree topology to avoid repeated handshakes and use sentinel nodes to monitor the status of root nodes, greatly reducing authentication costs.
3. We develop different communication schemes for device addition, deletion, heartbeat, disconnection, and election behaviors.
4. We formally analyze the correctness of the LBAS authentication phase using BAN logic and examine two common man-in-the-middle attack methods to demonstrate the protocol's security.
5. We compare LBAS with common IoT authentication schemes such as the TLS proxy scheme and the traditional DTLS scheme, proving that our scheme incurs lower overhead.

2 Related Work

2.1 Authentication Technology

With the proliferation and development of IoT, various IoT authentication protocols have emerged, while the underlying technologies remain relatively stable and can be categorized into the following types[3]:

1. Biometric Authentication[4]: Biometric authentication uses fingerprints, iris recognition, facial recognition, etc., to perform authentication, providing high security. It is mainly used for user-to-device authentication rather than device-to-device authentication.
2. Hardware Authentication[5]: Hardware-based authentication employs devices like Trusted Platform Modules (TPM) or USB keys to prevent malicious software attacks. This approach offers strong security but is more costly and limited, as it cannot be used with devices lacking the necessary hardware.
3. Pre-Shared Key (PSK) Authentication[6]: PSK authentication requires pre-allocated shared keys between communicating parties to ensure connection security. This method lacks a comprehensive authentication mechanism, is susceptible to man-in-the-middle attacks, and it is challenging to securely change pre-set content on already deployed devices.
4. Public Key Infrastructure (PKI) Authentication[7]: PKI authentication uses third-party digital certificates to ensure device legitimacy. This approach is secure and flexible but requires multiple public key cryptography computations, demanding high device performance, making it unsuitable for large-scale device mutual authentication.
5. Central Cloud Platform or Edge Base Station Indirect Authentication: This method typically leverages blockchain technology[8] or database technology to store device information remotely, making it unsuitable for M2M scenarios.

2.2 Authentication Protocols

In recent years, with the rapid development of IoT technology and the increasing number of IoT devices, balancing the security, complexity, and flexibility of authentication protocols in M2M scenarios has become a research focus.

In academia, Li et al.[9] optimized M2M communication methods by combining blockchain mobile edge computing (MEC) and unmanned aerial vehicle (UAV) technology, using drones as temporary communication base stations and offloading complex computational tasks to the nearest MEC server. Although MEC servers cleverly avoid blockchain-induced performance issues, the deployment costs of MEC itself are high. Studies like[10] and [11] used smart contracts for device-to-device authentication. However, this method requires storing Ethereum in central cloud servers, proxy servers, or edge base stations, making it unsuitable for M2M scenarios.

In[12], the authors investigated anonymous batch authentication in vehicular ad-hoc networks (VANETs) and proposed the CPAV authentication scheme. This scheme verifies the identity of road-side units (RSUs) and the integrity and validity of messages instead of verifying each message's identity individually, reducing the authentication burden. However, CPAV requires prior registration of vehicle and RSU information with a management center, making it unsuitable for IoT scenarios lacking a central server.

Wang et al.[13] proposed AAKA-D2D, an anonymous authentication and key agreement security protocol for D2D communication in 5G networks. This protocol allows two nearby devices to obtain pseudonyms and keys from a base station for anonymous

identity authentication and use DHKE to negotiate session keys. Madanchi et al.[14] proposed AKATGB, a tree-based authentication and key agreement protocol for D2D group communication in 5G scenarios. This protocol organizes user devices in the last layer of a tree structure, where sibling nodes first authenticate each other and generate shared keys, then exchange shared keys with devices of different parent nodes for group communication. Both schemes reduce the overhead of mutual authentication between 5G devices from the perspectives of anonymous and batch authentication but still rely on base station involvement, making them unsuitable for M2M scenarios.

Zhu et al.[15] proposed LHAP, a scalable and lightweight authentication protocol for ad-hoc networks, using hop-by-hop authentication, one-way key chains, and broadcast authentication protocols to authenticate each data packet and prevent unauthorized nodes from injecting traffic. This protocol assumes that each node has a genuine CA certificate for identity authentication and uses broadcast communication and authentication between nodes. However, for low-performance IoT devices, the O(n) time complexity of certificate verification among numerous devices is an unbearable overhead.

Alireza et al.[16] proposed a lightweight authentication mechanism based on hash and XOR operations for industrial IoT devices with limited computational power and bandwidth. This scheme uses secure elements (SE) or trusted platform modules (TPM) as the root of trust, registering with an authentication server and generating keys based on this root of trust. The issue lies in the reliance on hardware chips for the trust chain, making it economically unrealistic to install chips in every low-performance device. Additionally, the authentication process requires the participation of a central authentication server, preventing direct device-to-device authentication.

Furthermore, many researchers have worked on simplifying existing IoT authentication protocols. Kim et al.[6] optimized TCP protocol authentication overhead in smart grid scenarios, proposing a symmetric pre-shared key (PSK) based authentication scheme for low-performance devices. Obaidat et al.[17] used a timestamp and one-time key encryption scheme to replace CoAP's default DTLS. Park et al.[18] designed a multicast CoAP security framework based on UDP multicast to replace the multicast DTLS handshake. Hamad et al.[19] used TLS and ABE to ensure secure communication in the publish-subscribe model of MQTT. These schemes aim to optimize computational overhead during protocol authentication but do not reduce the number of batch authentications and may compromise the original protocol's security.

Among the aforementioned authentication schemes, some require the participation of central servers or base stations, making them unsuitable for M2M scenarios. Others require pre-installed hardware chips or pre-shared keys, increasing device costs and lacking flexibility for iterations and updates. Some focus on simplifying the complexity of authentication protocols, but this simplification also reduces security.

3 Scheme Design

We propose an M2M-oriented authentication scheme to improve device authentication efficiency. The entire scheme can be divided into three phases: constructing a trust

domain, the authentication process, and trust domain management. The main symbols and abbreviations are listed in **Table 1**.

Table 1. Main Symbols and Abbreviations

Notations	Descriptions	Purpose
$E_x(text)$	Encryption function	Use *x* key to encrypt plaintext
$D_x(text)$	Decryption function	Using the *x* key to decrypt ciphertext
$H(text)$	Hash function	Calculate the hash value of text
M	Manufacturer	Producer of equipment
Lic_j	*j*th device License	To verify the legitimate identity of the device
prk_M	Manufacturer's private key	Activation and batch keys computation
puk_M	Manufacturer's public key	Activation key computation
Id_{HFj}	Hardware Fingerprint	Unique identification of device identity
ET	Expired Time	Expiration time of device License
$DAttr$	Device attribute	Device type, device name, device ID...
CS	Capability Set	Communication method and algorithm...
$Dinfo_j$	Device info	Including MAC, *DAttr*, *UAddr*...
HP_j	Hardware parameters	Including CPU info, Main board number...
PIN	PIN code	Used to verify the user's identity
SK	Session key	Used to encrypt communication content
$UAddr_j$	Unicast address	Unicast address of the device
T_x	Timestamp or time interval	T_{cur} denotes the current timestamp, T_{device} denotes the timestamp sent by a certain device
MAC_j	MAC address	MAC address of device *j*
Pos_j	*j*th position	The *j*th position of the trust domain
Dev_x	Device	When *x* is "root", the device is the trusted root device, otherwise it is a regular device

3.1 Construct a Trusted Domain: Authentication Domain Structure

When organizing trusted heterogeneous devices in M2M scenarios into a trust domain, a mesh topology increases the communication burden on the root node, while a simple tree structure faces issues such as degrading into a linked list, and structures like AVL trees require complex algorithms to maintain, which are unsuitable for low-performance devices. Therefore, we adopt a complete binary tree topology to manage the trust domain. This approach reduces the complexity of batch authentication while not sacrificing too much performance in maintaining the topology.

We categorize all devices in the trust domain into root devices, sentry devices, and ordinary devices.

- Root Device: Responsible for listening to broadcast messages, authenticating newly added devices, generating and transmitting session keys, and relaying all control signaling messages.
- Sentry Devices: Responsible for maintaining the online status of the Root device. Considering reliability and performance, we use TCP long connections and periodically send heartbeats to monitor the Root device's online status. If the TCP connection is broken, the Root device is considered offline, and a sentry device must upgrade to the new root device and re-elect sentry devices.
- Ordinary Devices: Responsible for receiving control information from the parent node and forwarding it to child nodes. Parent nodes must maintain the online status of child nodes and promptly report any disconnections to the Root device for removal.

To avoid the overhead of repeated handshakes during batch authentication, new devices only perform handshakes with the Root device in the trust domain, which then propagates the trust relationship to other devices in the domain.

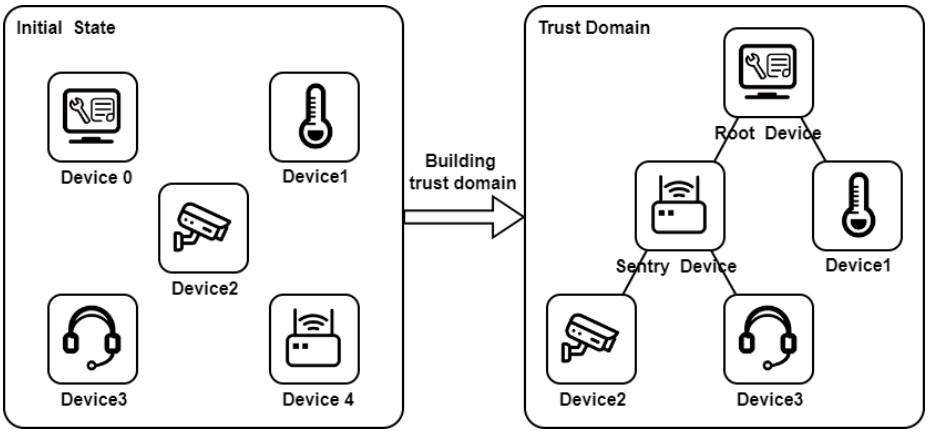

Fig. 1. Diagram of building a trust domain

3.2 Authentication Process: Design of The Authentication Protocol

A new device goes through four stages from manufacturing to integration into a trust domain: system setup stage, initial filtering stage, handshake authentication stage, and trust propagation stage.

The system initialization stage ensures the device has a legitimate identity. PIN code authentication provides an initial filter for the device.

A four-way handshake ensures the legitimacy of both parties' identities. Finally, the trust propagation mechanism extends the trust relationship to other devices in the trust domain, ensuring the confidentiality, integrity, and availability of communication

within the domain. The following sections will detail the specific processes of these four stages.

System Setup Stage. Before a device leaves the factory, the manufacturer issues an identity certificate $Lic = E_{prk_M}(Id_{HF} \| ET \| CS \| DAttr)$ to the device using prk_M, where $Id_{HF} = H(Dinfo_A \| HP_A)$ is unique, stable, and non-forgeable. Next, the device needs to generate its own RSA keys puk_A and prk_A, and carry the manufacturer's public key puk_M. Finally, when initializing the device, the user needs to input a PIN code.

Initial Filtering Stage. In this stage, the device first needs to filter IP addresses, protocols, and other criteria according to firewall policies. Additionally, the device must filter based on the PIN code. The PIN code is typically a 6-digit number defined by the user and is used to determine whether devices belong to the same user, ensuring all devices under the same user share the same PIN code. If the PIN codes between devices differ, they cannot authenticate each other. PIN code authentication effectively prevents devices from different users from authenticating each other, thereby stopping attackers from connecting to the trust domain through legitimate devices and causing information leakage.

Handshake Authentication Stage. After the coarse-grained PIN code authentication, a strict identity verification is needed to ensure the legitimacy of the connecting device. The entire verification process is conducted in four steps, using new Device A and the trusted device Root as an example, as shown in **Fig. 2**:

1. First, the new device A broadcasts within the local area network and sends $Request_1 = (UAddr_A \| PIN \| puk_A)$. To ensure that subsequent information transmission is not intercepted by attackers, we use public key encryption to encrypt the communication content. Therefore, the new device needs to carry its own public key during the first broadcast. Additionally, considering that frequent broadcasting can lead to bandwidth congestion and network delays, the entire communication process involves broadcasting only the first message, with the remaining communications being unicast. To facilitate unicast communication, Device A also needs to carry its own unicast address.
2. Upon receiving the broadcast, devices in the trust domain determine whether they are the Root device. If not, they discard the broadcast message. Regardless of how many devices are in the trust domain, the Root device will perform the handshake authentication with the new device, reducing the complexity of handshakes to $O(1)$ and significantly alleviating the burden on devices during batch authentication. For the Root device, it must first filter the message based on the PIN code and reply with $Response_1$:
   a. The root device checks whether the PIN code matches the PIN code in the trust domain; if not, it rejects the connection.

b. The Root device extracts hardware parameters to compute its own hardware fingerprint: $Id_{HF_{Root}} = H(Dinfo_{Root} \| HP_{Root})$

c. The root device replies with a unicast message, using the current timestamp to prevent the reuse of $Id_{HF_{Root}}$:

$$Response_1 = E_{puk_A}(Lic_{Root} \| puk_{Root} \| UAddr_{Root} \| H(Id_{HF_{Root}} \| T_{cur}) \| T_{cur})$$

d. The root device caches Device A's information in a semi-authenticated queue and sets a timeout. If the handshake times out or authentication fails, it removes Device A from the semi-authenticated queue.

3. Upon receiving $Response_1$, Device A should verify the identity of the Root device and respond with its own identity information.

a. Device A first needs to use its private key to decrypt $Response_1$ to ensure that the authentication information is intact and that even if attackers intercept the message, they cannot obtain any valid information from it:

$$(Lic_{Root} \| puk_{Root} \| UAddr_{Root} \| H(Id_{HF_{Root}} \| T_{cur}) \| T_{cur}) = D_{pri_A}(Response_1)$$

b. Next, Device A should use the manufacturer's public key to decrypt Lic_{Root} and verify its legitimacy:

$$D_{puk_M}(Lic_{Root}) = (Id'_{HF_{Root}} \| ET'_{Root} \| CS'_{Root} \| DAttr'_{Root})$$

If the decryption is successful, it indicates that Lic_{Root} is indeed a legitimate certificate issued by the manufacturer. However, since an attacker might obtain the certificate from a legitimate device, it is also necessary to prove the certificate's matching with the device.

c. Third, Device A should check whether ET'_{Root} has expired and verify the real-time nature and uniqueness of the hardware fingerprint:

$$H(Id'_{HF_{Root}} \| T_{cur}) == H(Id_{HF_{Root}} \| T_{cur})$$

$$Id_{HF_{Root}}' == Id_{HF_{Root}}$$

If they do not match, the device's identity is suspect, and the handshake is abandoned. ET'_{Root} ensures the certificate is within its validity period, while comparing the hash values of the hardware fingerprint and timestamp effectively prevents man-in-the-middle attacks.

d. After the verification steps, Device A can confirm the legitimate identity of the Root device. Next, Device A should prepare its own authentication information, encrypt it with the Root device's public key, and send it to the Root device:

$$Id_{HF_A} = H(Dinfo_A \| HP_A)$$

$$Request_2 = E_{puk_{Root}}(Lic_A \| H(Id_{HF_A} \| T_{cur}) \| T_{cur})$$

4. Upon receiving the message, the Root device will also verify the legitimacy of Lic_A and Id_{HF_A}. If the authentication is successful, it will decide, based on the security

policy, whether to regenerate SK' and synchronize the new device's information with the other devices in the trust domain.

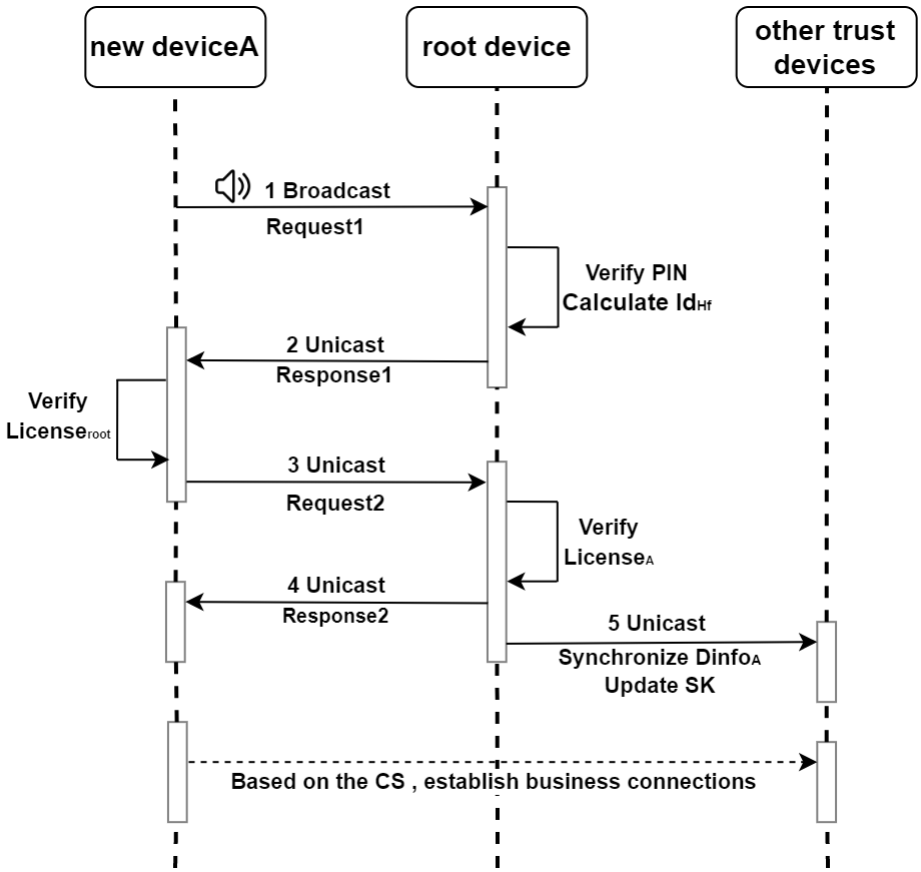

Fig. 2. Diagram of Handshake Authentication Stage Sequence

Trust Propagation Stage. After Device A and the Root device mutually confirm each other's identity, the Root device will transmit Device A's CS , $DAttr$, and $Dinfo_A$ to the other devices in the trust domain. Upon receiving this information, these devices will cache it locally and use the cached information for direct connection when needed. Trust propagation reduces the number of authentications between devices and improves the efficiency of batch authentication.

Specifically, after the new device successfully authenticates with the Root device, it will be inserted at the end of the authentication tree, and the Root device will send an $AddDevice$ message to the child devices.The message is serialized using protobuf and encrypted with SK . After receiving the $AddDevice$ message, child devices will update their own topology and forward the message to their child devices until it

reaches the bottommost device, as shown inFig. 3. Diagram of Trust Propagation **Fig. 3**.

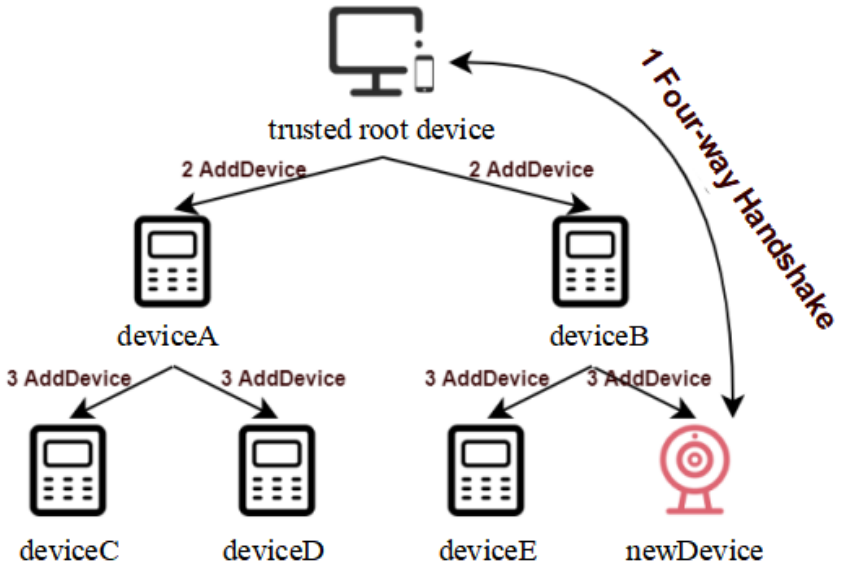

Fig. 3. Diagram of Trust Propagation

3.3 Manage The Trusted Domain: Trusted Domain Networking Strategy

As previously mentioned, the LBAS protocol organizes a large number of M2M devices in a trust domain using a complete binary tree topology. This approach neither requires the participation of a central server or base station nor effectively reduces the overhead of batch authentication between devices. However, device failures and network anomalies are inevitable in real-world scenarios. Therefore, the LBAS scheme must also ensure system stability to handle various unexpected situations.

Trusted Domain Adjustment Strategy for Voluntary Device Departure. When deleting an ordinary device or when an ordinary device voluntarily leaves the trust domain, the device to be deleted, Device A, sends a *Bye* message to the Root device. Once the Root device receives this message, it sends an *Upgrade* message to the bottommost node, allowing this node to replace the deleted node's position in the current topology. Simultaneously, the Root device propagates a *DelDevice* message downwards, step by step, to update the global topology of the trust domain, ensuring that the entire network is promptly informed of changes to the devices in the trust domain, as shown in **Fig. 4**:

$$Upgrade = E_{SK}(Pos_{parent} \| Pos_A \| Pos_{end})$$
$$DelDevice = E_{SK}(Pos_A \| Id_A \| Id_{end} \| SK_{end})$$

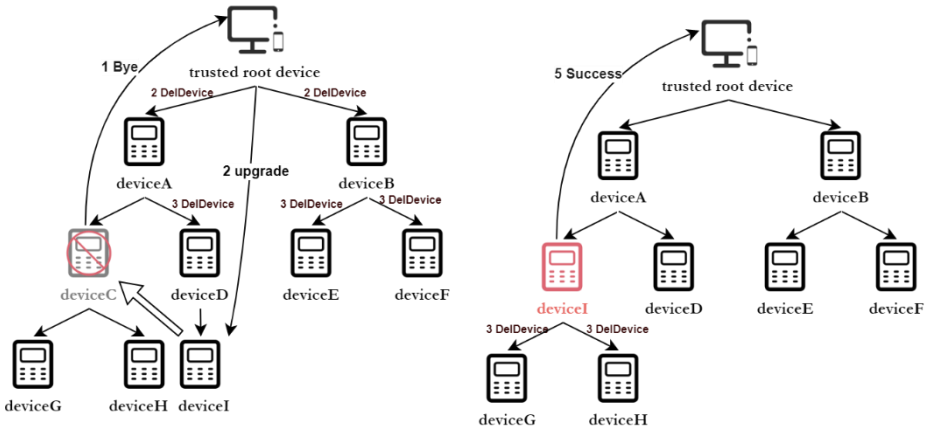

Fig. 4. Diagram of Proactive Deletion of an Ordinary Device

If a sentry device is deleted or voluntarily leaves, the Root device will select a new sentry node from the trusted domain. This new sentry node must meet the capability requirements and have the longest uptime. The Root device then sends an *UpgradeSentry* message. Upon receiving the message, the newly selected sentinel node must establish a TCP long connection with the new Root device and periodically send heartbeat messages:

$$UpgradeSentry = E_{SK}(Id_{max(t_{hb}-t_{add})})$$

If the trusted Root device is deleted or voluntarily leaves, the following actions will occur, as shown in **Fig. 5**:

1. The trusted Root device sends a *Bye* message to the sentry device.
2. Upon receiving the *Bye* message, the sentry device automatically upgrades to become the new trusted Root node and selects a new sentry node according to the rules, sending an *UpgradeSentry* message.
3. The trusted Root device needs to send a *ChangeRoot* message to its child nodes, informing them of the topology change.
4. The trusted Root device sends an *Upgrade* message to the bottommost node, allowing it to replace the sentry node's current position in the topology.
5. Once the bottommost node is successfully upgraded, it replies with a *Success* message to the new Root node and continues to send *ChangeRoot* messages to its child nodes.

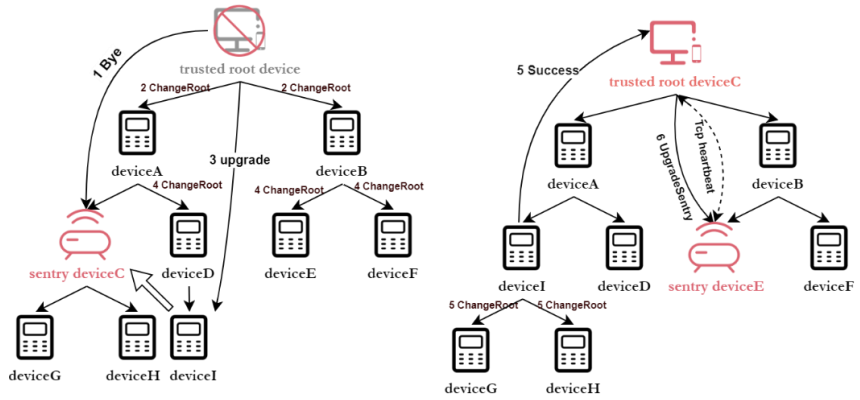

Fig. 5. Diagram of Proactive Deletion of a Root Device

Trusted Domain Adjustment Strategy for Exceptional Situations. In LBAS protocol, the lifecycle of each ordinary node is managed by its parent node. The Root node periodically sends heartbeat messages downward to determine the online status of each node in the authentication tree.

Upon receiving the heartbeat message, nodes must immediately reply with an ACK and forward it to their child nodes. If the parent node does not receive an ACK message, it will immediately mark the child node as dead and retry to avoid sporadic disconnections due to network issues. If the node cannot be contacted after three attempts, the parent node will assume the device is offline.

The only difference between an ordinary node disconnecting and an ordinary node being deleted is that, in the case of disconnection, the *Bye* message is sent by its parent node, as shown in **Fig. 6**:

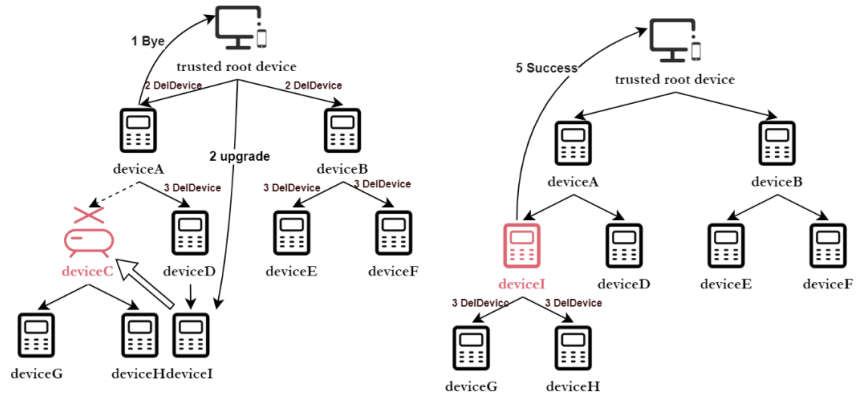

Fig. 6. Diagram of Disconnection of an ordinary Device

If the trusted Root node becomes unreachable, the sentry node will detect the TCP connection break and send an *Upgrade* message to the bottommost node, allowing it

to take the sentry node's place. The sentry node then becomes the new trusted Root node and sends *ChangeRoot* messages to its child nodes, selecting a new sentry node and maintaining the TCP connection, as shown in **Fig. 7**.

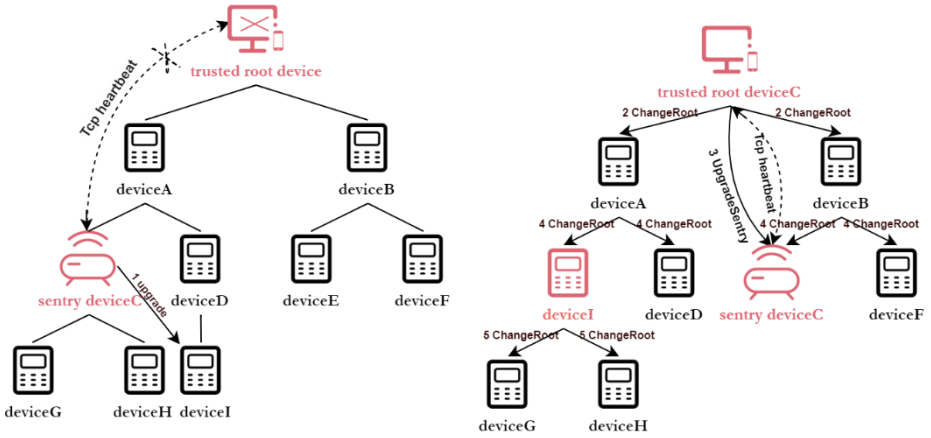

Fig. 7. Diagram of Disconnection of a Root Device

4 Security Analysis and Performance Evaluation

4.1 Security Analysis

BAN Logic Verification. BAN logic [20] is widely used to prove security properties such as key agreement and mutual authentication. In this section, the correctness of the LBAS scheme's authentication phase is analyzed using BAN logic. First, the transmission messages during the authentication phase of the LBAS scheme are converted into BAN logic language, along with the content and security objectives of the scheme. Then, the basic assumptions of the LBAS scheme are presented, and based on these assumptions and BAN logic rules, it is demonstrated that the proposed scheme can meet the security objectives.

The initial basic assumptions of LBAS are as follows:

$$Dev_A| \equiv (puk_{Root}) \tag{1}$$

$$Dev_{Root}| \equiv (puk_A) \tag{2}$$

$$Dev_i| \equiv (puk_M) \tag{3}$$

$$Dev_i| \equiv Lic_i \rightarrow M \tag{4}$$

$$Dev_i| \equiv (Id_{HF_i}, T_{cur}) \tag{5}$$

Assumption (1), (2) and (3) represent that the new device believes puk_{Root} is the Root device's public key, the Root device also believes puk_A is the new device's public

key, and all devices believe puk_M is the manufacturer's public key. Due to the visibility characteristics of public keys, these basic assumptions are feasible.

In a trusted environment, identity authentication is based on certificates issued by the manufacturer, and the protocol requires verification of the manufacturer's signature during operation. Therefore, assumption (4) is feasible. Each device can obtain its own hardware parameters during operation and calculate its hardware fingerprint using a specified algorithm. Additionally, devices calibrate the time at startup and can retrieve the current timestamp at any moment. Hence, devices have authority over their own hardware fingerprints and timestamps, making assumption (5) feasible.

The goal of this section is:

$$Dev_{Root} | \equiv Dev_A, Dev_A | \equiv Dev_{Root} \tag{6}$$

If this formula holds, it indicates that the new device A and the trusted root device have successfully authenticated each other and mutually confirmed the session key. The message contents and logical analysis involved in the authentication phase of the LBAS protocol are as follows:

$$Dev_A \rightarrow Dev_i : Request_1(UAddr_A, PIN, puk_A) \tag{7}$$

A new device (Dev_A) broadcasts a $Request_1$ message (contains its $UAddr, PIN$ and its public key puk_A, indicating its desire to join the trust domain and notifying the Root device to initiate the authentication process.

$$Dev_{Root} \rightarrow Dev_A : Response_1\{Msg_1\}_{puk_A} \tag{8}$$

Upon receiving the broadcast $Request_1$, the Root device filters it based on the PIN and $UAddr$. If the request passes the filtering process, it calculates H_i. Lic_{Root} denotes the device's factory-issued certificate.

$$Msg_1 = \{Lic_{Root}, UAddr_{Root}, H_i, puk_{Root}, T_{cur}\} \tag{9}$$

$$H_i = hash(Id_{HF_{Root}}, T_{cur}) \tag{10}$$

$$Lic_{Root} = \{Id_{HF}, DAttr, CS, ET\}_{prk_M} \tag{11}$$

The new device will verify the identity of the Root device after receiving $Response_1$. According to Equation (5) and (8), we can obtain:

$$\frac{Dev_i | \equiv \rightarrow M, Dev_A \lhd \{Lic_{Root}\}_{puk_A}}{Dev_{Root} | \equiv Dev_A | \sim Lic_{Root}} \tag{12}$$

This proves that device Dev_A can trust the certificate Lic_{Root} is issued by the manufacturer. Next, the new device Dev_A uses the manufacturer's public key to decrypt Lic_{Root} and obtain the hardware fingerprint of the Root device. According to Equation (5), (6), (7) and (8), it can be seen that:

$$\frac{Dev_{Root} \mid\Rightarrow (Id_{HF_{Root}}, T_{cur}), Dev_A \lhd \{Lic_{Root}\}_{puk_A}}{Dev_{Root} \mid\equiv Dev_A \sim \{H_i, T_{cur}\}} \tag{13}$$

This indicates that Dev_A believes T_{cur} originates from the Root device and that H_i is the hash value of the Root device's Id and T_{cur}. Therefore, Dev_A only needs to verify whether the Root device's $Id_{HF_{Root}}$ is consistent with the $Id'_{HF_{Root}}$ in the Lic_{Root}, which is to verify:

$$H_i == hash(Id'_{HF_{Root}}, T_{cur}) \tag{14}$$

If the verification is successful, according to the trust rules, it is understood that:

$$Dev_A \mid\equiv Dev_{Root} \tag{15}$$

In summary, our scheme can meet the security objectives.

An attacker impersonates a new device to join the trust domain. An attacker intercepts the four-way handshake messages of a legitimate device C joining the trust domain and attempts to impersonate device C to join the trust domain and steal information. When the Root device receives the third handshake message $Request'_2$, it will verify the device's identity. However, whether the attacker uses the current timestamp or device C's timestamp, it will fail the verification, as shown in **Fig. 8**.

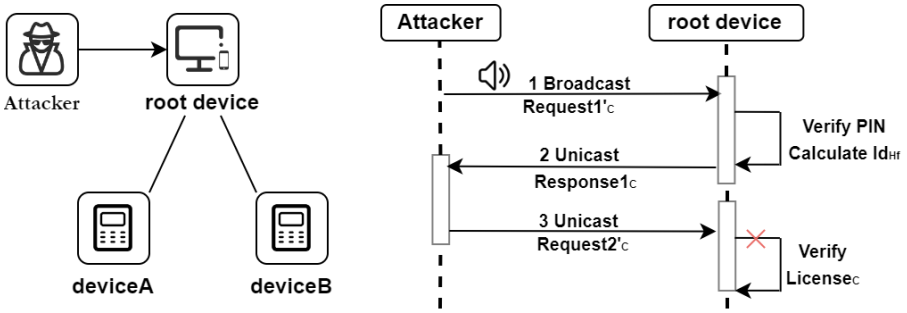

Fig. 8. Diagram of an Attacker Impersonating a New Device

If the attacker uses T_{cur} to construct $Request'_2$:

$$(Lic'_C \| H(Id'_{HF_C} \| T_{deviceC}) \| T_{cur}) = D_{pri_{Root}}(Request'_2) \tag{16}$$

$$H(Id'_{HF_{Root}} \| T_{deviceC}) == H(Id_{HF_{Root}} \| T_{cur}) \quad \times \tag{17}$$

If the attacker uses $T_{deviceC}$ to construct $Request'_2$:

$$(Lic'_C \| H(Id'_{HF_C} \| T_{deviceC}) \| T_{deviceC}) = D_{pri_{Root}}(Request'_2) \tag{18}$$

$$H(Id'_{HF_{Root}} \| T_{deviceC}) == H(Id_{HF_{Root}} \| T_{deviceC}) \quad \checkmark \tag{19}$$

$$T_{cur} - T_{deviceC} < T_{gap} \quad \times \tag{20}$$

An attacker impersonates the trusted root device to connect to a new device. If an attacker intercepts a handshake message and attempts to impersonate the trusted root node to establish a trust relationship with a new node, as shown in **Fig. 9**, the new device can identify the attacker by verifying the timestamp in the second handshake. This is similar to impersonating a new device. By checking the validity of the hardware fingerprint's timestamp, the new device can distinguish between the attacker and the trusted root device.

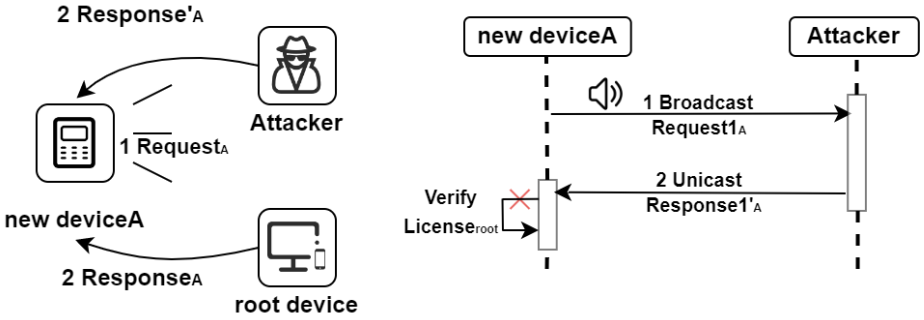

Fig. 9. Diagram of an Attacker Impersonating the Trusted Root Device

4.2 Performance Evaluation

In the following section, we analyze LBAS from the perspectives of complexity analysis and data fitting, using a 64-bit Ubuntu Linux 20.04 operating system and a 12th Gen Intel® Core™ i7-12700 CPU. (In the experimental setup, asymmetric computation uses the RSA algorithm with a 1024-bit key length, and hash computation uses the SHA256 algorithm).

Complexity Analysis. We compare the LBAS with the following two common device authentication schemes in the IoT domain:

- Broker Scheme: Each device only needs to establish a trust relationship with a broker within the local network to obtain information about other devices. This scheme uses the TLS 1.2 (Transport Layer Security) protocol for identity verification. During the TLS handshake authentication process, four data communications occur, and two public key operations are performed on the server and client sides.
- Traditional Scheme: When a device joins a trust domain, it needs to perform a handshake with each device in the trust domain to establish a trust relationship. This scheme typically uses the DTLS 1.2 (Datagram Transport Layer Security) protocol for end-to-end data protection. During the DTLS handshake authentication process, six data communications occur, and two public key operations are performed on the server and client sides.

The time cost of the schemes can be divided into two parts: network communication cost and encryption computation cost. In an internal network environment, the message transmission time T_c and processing time T_d can be considered constants, meaning the total time for message processing and transmission depends only on the number of message exchanges. The main overhead is the identity authentication time T_a, which specifically includes the time for digest computation T_h and the time for a single public key operation T_k (where T_k = public key encryption time + decryption time). We analyze the complexity of the aforementioned schemes, as shown in **Table 2**.

Table 2. Time Complexity of the Three Authentication Schemes

Scheme	1-1 authentication	1-N authentication	N-N authentication
LBAS	$T_{a1}=4T_c+2T_k+T_h$	$T_{a1}+\log n(T_c+T_d)$	$nT_{a1}+log\frac{(2n)!}{n!}(T_c+T_d)$
Broker Scheme	$T_{a2}=4T_c+2T_k$	$T_{a2}+T_c+nT_d$	$nT_{a2}+nT_c+\frac{n^2}{2}T_d$
Traditional Scheme	$T_{a3}=6T_c+2T_k$	nT_{a3}	$\frac{n^2}{2}T_{a3}$

Fig. 10 compares the average latency of the aforementioned three schemes for 50, 100, 150, and 200 mutual authentications between devices. The latency per handshake for LBAS and the broker scheme is very similar, but both are approximately 17.6% higher than the traditional scheme. A single handshake does not reflect the advantage of LBAS in trust propagation. Therefore, we further compare the time overhead of each scheme in batch authentication scenarios.

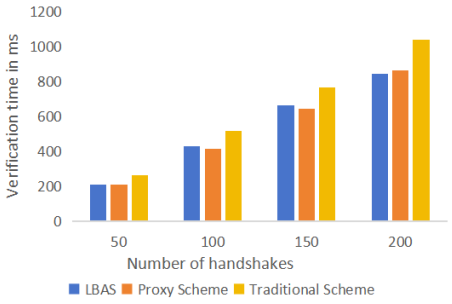

Fig. 10. 1-1 devices authentication

Data Fitting. compares the latency required to establish trust between a new device and N trusted devices for the three schemes. The traditional scheme requires the new device to handshake sequentially with N devices, resulting in a complexity of $O(n)$.

In contrast, both the broker scheme and LBAS only need to handshake with the broker or the trusted root device, resulting in a complexity of $O(1)$. The test results show that the difference between the broker scheme and LBAS is negligible when the number of devices is small. However, as the number of devices increases, LBAS significantly outperforms, with the time overhead being approximately 47.3% lower than that of the broker scheme. This is because, in the broker scheme, the broker needs to propagate the trust relationship to each trusted device, resulting in a complexity of $O(n)$. In LBAS, trusted devices are organized into a complete binary tree, reducing the trust propagation complexity to $O(logn)$.

Fig. 12 compares the latency required for N unfamiliar devices to establish mutual trust relationships for the three schemes. The time overhead for the traditional scheme is significantly higher than that of LBAS and the broker scheme. Furthermore, the time overhead of LBAS is approximately 41.2% lower than that of the broker scheme, demonstrating the superiority of LBAS in batch authentication of a large number of devices.

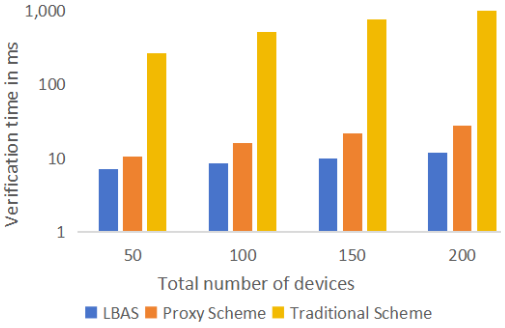

Fig. 11. 1-N devices authentication

Fig. 12. N-N devices authentication

References

1. Osamy, W., et al.: IPDCA: intelligent proficient data collection approach for IoT-enabled wireless sensor networks in smart environments. Electronics 10, 997 (2021)
2. Verma, P.K., et al.: Machine-to-Machine (M2M) communications: A survey. Journal of network and computer applications 66, 83-105 (2016)
3. Kumar, A., et al.: A comprehensive survey of authentication methods in Internet-of-Things and its conjunctions. Journal of Network and Computer Applications (2022)
4. Kumar, V., et al.: Light Weight Authentication Scheme for Smart Home IoT Devices. Cryptography 2022, 6, 37. (2022)
5. Kondoh, T., Ueno, M.: A Proposal of Applying TPM Based Authentication on IoT Devices Access Control Techniques. IEICE Tech. Rep. 116, 223-226 (2017)
6. Kim, S., et al.: Performance evaluation of random access for M2M communication on IEEE 802.16 network. IEEE (2012)

7. Shen, S., Tang, S.: Cross-Domain Grid Authentication and Authorization Scheme Based on Trust Management and Delegation. In: International Conference on Computational Intelligence & Security. (2008)
8. Hammi, M.T., et al.: Bubbles of Trust: A decentralized blockchain-based authentication system for IoT. Computers & Security (2018)
9. Li, M., et al.: UAV-assisted data transmission in blockchain-enabled M2M communications with mobile edge computing. IEEE Network 34, 242-249 (2020)
10. Abdelrazig Abubakar, M.W.R.W.A.N., et al.: Blockchain-based identity and authentication scheme for MQTT protocol. In: 2021 The 3rd International Conference on Blockchain Technology, pp. 73-81. (2021)
11. Yang, X., et al.: A Hybrid Blockchain-Based Authentication Scheme for Smart Home. In: 2020 IEEE 5th International Conference on Signal and Image Processing, pp. 893-897. (2020)
12. Vijayakumar, P., et al.: Computationally efficient privacy preserving anonymous mutual and batch authentication schemes for vehicular ad hoc networks. Future generation computer systems 78, 943-955 (2018)
13. Wang, M., et al.: AAKA-D2D: Anonymous Authentication and Key Agreement Protocol in D2D Communications. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, pp. 1356-1362. (2019)
14. Madanchi, M., Abolhassani, B.: Authentication and Key Agreement Based Binary Tree for D2D Group Communication. In: 2020 ICEE, pp. 1-5. (2020)
15. Zhu, S., et al.: LHAP: a lightweight hop-by-hop authentication protocol for ad-hoc networks. In: 23rd International Conference on Distributed Computing Systems Workshops, 2003. Proceedings., pp. 749-755. (2003)
16. Esfahani, A., et al.: A lightweight authentication mechanism for M2M communications in industrial IoT environment. IEEE Internet of Things Journal 6, 288-296 (2017)
17. Obaidat, M.A., et al.: A Secure Authentication and Access Control Scheme for CoAP-based IoT. In: 2022 CIoT, pp. 145-149. (2022)
18. Park, C.-S.: Security Architecture for Secure Multicast CoAP Applications. IEEE Internet of Things Journal 7, 3441-3452 (2020)
19. Hamad, M., et al.: SEEMQTT: Secure End-to-End MQTT-Based Communication for Mobile IoT Systems Using Secret Sharing and Trust Delegation. IEEE Internet of Things Journal 10, 3384-3406 (2023)
20. Burrows, M., et al.: A logic of authentication. ACM Transactions on Computer Systems (TOCS) 8, 18-36 (1990)