Hindawi
Security and Communication Networks
Volume 2020, Article ID 8897282, 17 pages
https://doi.org/10.1155/2020/8897282

Research Article

An Improved Broadcast Authentication Protocol for Wireless Sensor Networks Based on the Self-Reinitializable Hash Chains

Haiping Huang,[1,2] Qinglong Huang,[1,2] Fu Xiao,[1,2] Wenming Wang,[1,3] Qi Li,[1] and Ting Dai[4]

[1]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210003, China*
[2]*Jiangsu High Technology Research Key Laboratory for Wireless Sensor Networks, Nanjing 210023, China*
[3]*University Key Laboratory of Intelligent Perception and Computing of Anhui Province, Anqing Normal University, Anqing 246011, China*
[4]*Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA*

Correspondence should be addressed to Haiping Huang; hhp@njupt.edu.cn

Received 20 June 2020; Revised 24 July 2020; Accepted 31 July 2020; Published 1 September 2020

Academic Editor: Honghao Gao

Broadcast authentication is a fundamental security primitive in wireless sensor networks (WSNs), which is a critical sensing component of IoT. Although symmetric-key-based μTESLA protocol has been proposed, some concerns about the difficulty of predicting the network lifecycle in advance and the security problems caused by an overlong long hash chain still remain. This paper presents a scalable broadcast authentication scheme named DH-μTESLA, which is an extension and improvement of μTESLA and Multilevel μTESLA, to achieve several vital properties, such as infinite lifecycle of hash chains, security authentication, scalability, and strong tolerance of message loss. The proposal consists of the (t, n)-threshold-based self-reinitializable hash chain scheme (SRHC-TD) and the d-left-counting-Bloom-filter-based authentication scheme (AdlCBF). In comparison to other broadcast authentication protocols, our proposal achieves more security properties such as fresh node's participation and DoS resistance. Furthermore, the reinitializable hash chain constructed in SRHC-TD is proved to be secure and has less computation and communication overhead compared with typical solutions, and efficient storage is realized based on AdlCBF, which can also defend against DoS attacks.

1. Introduction

With the rapid development of Internet of Things (IoT) and 5G technology, the number of sensing terminals, such as various sensor nodes and tiny IoT devices, has also increased dramatically [1–3]. Edge computing is a new emerging paradigm that overcomes the scalability problem of traditional wireless sensor networks (WSNs) architecture [4–7]. The combination of wireless sensor networks and edge computing can more effectively deploy the network and process a large amount of sensory data from sensor nodes.

In hostile and harsh conditions, such as large-scale agricultural monitoring and homeland border detection, sensor nodes are usually deployed to the monitoring area by aircraft, and the base station may be a temporarily deployed edge server such as mobile weather station or UAVs (unmanned aerial vehicles for agricultural surveillance), whose computation and storage capacities are not always powerful. These sensor nodes are difficult to recycle and need to be replenished after damage or exhaustion. For effectively acquiring and perceiving data from massive sensors, the base station (or edge server) usually sends commands or application updating data packets to vast sensor nodes through broadcasting. It is necessary for sensor nodes to authenticate the identity of the sender, together with the validity and integrity of these messages [8–11]. Thus, the broadcast authentication becomes an essential service in practical and secure wireless sensor networks or IoT. The broadcast authentication protocol in wireless sensor network needs to meet the following three principles [12]: (1) any malicious receiver being able to hardly forge any packet from the sender; (2) low communication, computation, and storage overheads; and (3) tolerance of message loss or fault.

Based on the three principles, many broadcast authentication protocols applied in WSNs have appeared sequentially. μTESLA [13] is one of the most representative broadcast authentication protocols. The contributions of μTESLA lie in the low computation strength and high authentication speed by using the special asymmetry mechanism, which is realized by delaying the disclosure of symmetric keys. Subsequently, an improved version named Multilevel μTESLA [14] has been proposed to extend the capability of μTESLA. With the continuous concern regarding broadcast authentication, some novel protocols [15–19] sequentially emerge in WSNs, followed by the Multilevel μTESLA protocol.

Although these protocols are distinguished from each other, most of them are the enhanced versions of μTESLA or Multilevel μTESLA, which contribute to the safety and efficiency by improving the organization form of broadcast authentication. However, the disadvantages of them cannot be ignored. First, a traditional asymmetric mechanism, such as non-light-weight public key cryptography (PKC) in [20], is not encouraged to be used in broadcast authentication due to its high computation overheads and impracticability in resource-constrained sensor networks. Second, the light-weight hash key chain is adopted by most of these protocols. However, an overlong hash chain may lead to the increase of storage overhead and information divulging risk on the base station, especially in the edge computing scenario, and a short one will be consumed quickly, because the running time of the network is uncertain and the entire lifetime of a large-scale network is hard to predict. Third, a long hash chain has security risks itself. Håstad and Näslund [21] show that, in a hash chain composed of the same hash functions, the attackers inverting the k-th iteration are actually k times easier than inverting a single hash function. Kwon and Hong [22] find some future keys of μTESLA by utilizing time-memory-data-tradeoff technique in a 64-bit hash chain.

Furthermore, in complex and dangerous environments, it is necessary to authenticate the fresh node supplemented in edge computing scenario. The absence of the trusted authentication interactions is easy for the vicious nodes to create fake data packets. In a severe case, the base station will suffer DoS attacks as an authentication centre. Similarly, the sensor nodes also need the ability to quickly distinguish valid messages from fake ones against DoS attacks. This means that these protocols cannot achieve a satisfactory security level.

In order to address the above problems in the existing protocols, further improve the security, and broaden application scenarios, this paper puts forward a reformed broadcast authentication protocol named DH-μTESLA. The main contributions of our proposal can be described as follows:

(i) We design a (t, n)-threshold-based self-reinitializable hash chain scheme (SRHC-TD), which constructs a reinitializable hash chain securely to improve the lifecycle of hash chains while keeping a desirable efficiency. In our scheme, hash chains will continue to generate without the need to predict and determine the lifetime of the network in advance. This scheme also has a strong tolerance to message loss.

(ii) A d-left-counting-Bloom-filter-based authentication scheme (AdlCBF) is proposed, which can handle the secure authentication of fresh sensor nodes. The AdlCBF scheme will effectively achieve the demands of authentication speed, memory space, and data security with the increasing number of sensor nodes joining the network. This scheme ensures that our protocol is well scalable for a large-scale network, and it can effectively resist DoS attacks on the base station caused by the request of massive illegal nodes.

The remainder of this paper is organized as follows. Section 2 gives the overview of related work. Some notations and concepts are defined in Section 3. The SRHC-TD scheme and AdlCBF scheme are described in Section 4 and Section 5, respectively. In Section 6, we describe the DH-μTESLA protocol in detail. Section 7 illustrates the security analysis. The evaluation and comparison of the proposed schemes are described in Section 8. Section 9 concludes this paper.

2. Related Work

2.1. Hash Chain. The hash chain was first proposed by Lamport in 1981 and has been widely used in various applications due to its high security and efficiency, including one-time password (OTP) system [23] and broadcast authentication [24]. However, after all hash values of a hash chain are consumed, a new hash chain must be generated, which is expensive in most applications. For example, in OTP system, the commitment of a new hash chain and corresponding parameters need to be reregistered to the server or the client, which will consume significant computation and communication overhead [25]. In μTESLA protocol, the problem of unicasting the initial parameters on a node-to-node basis is quite expensive [24]. In order to solve this problem, many schemes have been put forward.

Bicakci and Baykal [26] and Di Pietro et al. [27] employed public key cryptography to generate a hash chain. Although the problem of limited length was solved, the computational overhead was increased (one public key operation takes hundreds of times as long as one hash operation). Park [25] constructed an infinite length hash chain by employing multiple short hash chains for OTP system. Nevertheless, this scheme cannot be used in broadcast authentication because the hash chain will be broken if any authentication message is lost.

Goyal firstly proposed the reinitializable hash chain (RHC) scheme [28], whose main idea was that when an RHC is exhausted, a new RHC can be regenerated safely and undeniably. In 2006, Zhang and Zhu put forward the self-updating hash chain (SUHC) scheme based on Hard Core Predicate algorithm [29]. The main idea of SUHC is that it distributes the first chain's every key value along with one bit of the second one's commitment. In this manner, the receiver would gain all bits of the second chain's commitment when the first one is exhausted. On the basis of [29], Zhang et al. designed the self-renewal hash chain (SRHC) scheme [30] as the improvement of SUHC, which has a different selection algorithm of random numbers. Xu et al. also proposed a self-updating one-time password (SUOTP) mutual authentication protocol in a similar way [31]. However, in all these schemes, the commitment can be reconstructed if and only if all the random numbers were received integrally.

2.2. Broadcasting Authentication Protocol in WSNs. μTESLA, originated from TESLA, was developed for source-constrained networks [13]. However, it did not overcome the problems that TESLA protocol experiences with mobile nodes losing the authentication packets caused by high velocities, requirement of loose time synchronization, limited length of hash chain, and reliability. Therefore, lots of solutions were proposed in order to address the above-mentioned problems.

Liu et al. [16] proposed a Scalable μTESLA that introduces the use of Merkel hash tree in μTESLA for the distribution of initial parameters and commitments and also enhances scalability by increasing the number of senders. Liu and Ning proposed Multilevel μTESLA to extend the capability of μTESLA in three aspects [14]. Firstly, it predetermines and broadcasts the initial parameters rather than unicasting them by point-to-point authentication used in μTESLA. Moreover, it adopts the multilevel hash chain to distribute broadcast messages, which prolongs the usage cycle while not increasing the length of hash chain compared with μTESLA. Finally, it uses redundant message transmission and random selection strategies to distribute key chain commitments, which improves the survivability against DoS attacks.

Recently, Al Dhaheri et al. [32] proposed a TLI-μTESLA based on Multilevel μTESLA, which reduced the delay between the sender and the receiver. Kwon and Hong presented an extendable broadcast authentication scheme called X-TESLA, which considers the problem arising from sleep modes, networks failures, and idle sessions [22]. Furthermore, a long-duration TESLA [33] was proposed to overcome the finite length of hash chain used in μTESLA by employing a hierarchical hash chain.

Apart from like-μTESLA protocols, there exist other types of broadcast authentication protocols that can be applied in WSNs. Groza et al. [34] designed a light-weight broadcast authentication for controller area network that achieves immediate authentication at small costs in bandwidth. Shim et al. [35] proposed an identity-based broadcast authentication scheme called EIBAS using ID-based signature. Similarly, a Chebyshev-map-based broadcast authentication is presented by Luo et al., which also uses ID-based signature [8]. However, the overhead of ID-based signature is higher than symmetric primitive. Besides, Bloom filter (BF) and counting Bloom filter (CBF) have been explicitly and widely used in the broadcast authentication schemes [36–39] to generate and verify authentication information, realize the public key management when combined with hash chain, and compare multiple MACs (message authentication codes) for reducing the message size. Kim and An employ BF-based source authentication (BFBSA) to achieve the security of packets with variable sizes in WSNs [38]. Bao et al. [39] proposed a light-weight authentication by combining BF and TESLA, which prevents active attacks and adds a privacy-preserving feature for vehicular ad hoc networks.

3. Preliminaries

3.1. Notation. The symbols and notations used in our protocol are listed in Table 1.

Additionally, $M_{\langle \text{mtype} \rangle} = (\langle \text{mtype} \rangle, \langle \text{sadr} \rangle, \langle \text{dadr} \rangle, \langle * \rangle)$ is designed as a message format, where $\langle \text{mtype} \rangle$ is the message type, $\langle \text{sadr} \rangle$ is the source address, $\langle \text{dadr} \rangle$ is the destination address, and $\langle * \rangle$ represents some additional options.

3.2. Basic Definitions

Definition 1 (partition). Partition is the process of transforming a binary number of L bits into m 2^l-radix numbers. This process can be simply represented as $L = (m, l)$ or $m = \lceil L/l \rceil$. If l cannot divide L exactly, several zeros whose number is exactly the remainder should be filled on the front of the binary number.

Definition 2 (repetition vector, value vector, and repetition degree). Suppose that there are m $(m \geq 1)$ variables, and the value of each variable is in a set containing n $(n \geq 1)$ integers. For a group of these variables, there are q $(1 \leq q \leq \min(m, n))$ different values among these variables, which are denoted as v_i, $i = 1, 2, \ldots, q$. Then, $m - q$ is called repetition degree and $(v_1, v_2, \ldots, v_q)$ is called value vector. Suppose that there are p_i $(1 \leq p_i \leq m)$ variables whose value is v_i, such that $\sum_{i=1}^{q} p_i = m$. Then, $(p_1, p_2, \ldots, p_q)$ is called repetition vector.

Definition 3 (repetition rate). For m variables, the values are a set of n integers, such that repetition degree is $m - q$. Then the number of assignments of these variables is

$$S_{m,n}^{q} = C_{m}^{p_1}, C_{m-p_1}^{p_2}, \ldots, C_{p_q}^{p_q} \cdot \frac{n!}{(n-q)!}. \quad (1)$$

The repetition rate P_q is defined as

$$P_q = \frac{S_{m,n}^{q}}{\sum_{i=1}^{\min(m,n)} S_{m,n}^{i}}. \quad (2)$$

Definition 4 (difficulty degree). Suppose that there are m integers whose repetition degree is $m - q$. Then q is called difficulty degree.

Table 1: Notations used in the protocol.

Symbols	Description
n	Length of hash chain
L	Output length of hash function
S	The secret that needs to be shared
I_i	Secret shares
P_U, P'_U	Seed of hash chain
m_L	Number of secret shares
ξ_i, ζ_i	Key authentication code
T_c	Current time used to synchronize the time of the whole network
Δ	Maximum clock difference between the sender and the receiver
T_i	The start time of interval i
T_{int}	Duration of each time interval
δ	Disclosure delay
S_i	Sensor node
KS_i, KP_i	Private key and public key of sensor node, respectively
S_B	Base station
ID_i, ID_B	The addresses of sensor node and base station, respectively
N, N_i	Random nonce

Definition 5 (average difficulty degree). Suppose that there are m integers, whose difficulty degree is q and repetition rate is P_q. Then average difficulty degree D_q is defined as the weighted sum of difficulty degree:

$$D_q = \sum_{q=1}^{\min(m,n)} P_q \cdot q. \tag{3}$$

Definition 6 (Mignotte's sequence). Let n and t be integers; $n \geq 2$, and $2 \leq t \leq n$. A (t,n)-Mignotte sequence is a sequence of positive integers $m_1 < m_2 < \cdots < m_n$, such that, for all $1 \leq i < j \leq n$ and $(m_i, m_j) = 1$, $m_{n-t+2} \cdot m_{n-t+3} \cdot \cdots \cdot m_n < m_1 \cdot m_2 \cdot \cdots \cdot m_t$.

4. (*t*, *n*)-Threshold-Based Self-Reinitializable Hash Chain Scheme

4.1. (t,n)-Mignotte's Threshold Secret Sharing Scheme. Given a (t,n)-Mignotte sequence, the scheme works as follows [40]:

(1) The secret S is chosen as a random integer such that $\beta < S < \alpha$, wherein $\alpha = m_1 \cdot m_2 \ldots m_t$ and $\beta = m_{n-t+2} \cdot m_{n-t+3} \ldots m_n$.

(2) The secret share I_i is chosen by the formula $I_i = S \bmod m_i$, for all $1 \leq i \leq n$.

(3) Given t distinct shares $I_{i_1}, I_{i_2}, \ldots, I_{i_t}$, the secret S can be recovered using the Chinese Remainder Theorem, and any two such S are congruent moduli $m_{i_1} \cdot m_{i_2} \ldots m_{i_t}$:

$$\begin{cases} x \equiv I_{i_1} \bmod m_{i_1}, \\ x \equiv I_{i_2} \bmod m_{i_2}, \\ \vdots \\ x \equiv I_{i_t} \bmod m_{i_t}. \end{cases} \tag{4}$$

4.2. (t,n)-Mignotte's Threshold Secret Sharing Scheme. The proposed self-reinitializable hash chain has multiple phases: initialization, publication, verification, recombination, and self-renewal. A new hash chain can be reinitialized without extra communication when the last hash chain is exhausted. Figure 1 shows the framework of the broadcast authentication process involving the construction of self-reinitializable hash chain.

4.2.1. Initialization. In the initialization phase, the sender and the receiver negotiate the length of hash chain n and a secure hash function h: $\{0,1\}^* \longrightarrow \{0,1\}^L$ with a security parameter L, which means that the output of h is an L-bits string, and L can be partitioned into (m_L, l_L). At first, the sender does the following steps.

(i) Initialize a seed. Choose an appropriate (t, m_L)-Mignotte sequence, denoted as $x_1, x_2, \ldots x_{m_L}$. Unite all the sequential values x_i into S_U and $h(x_i)$ into P_U, for all $1 \leq i \leq m_L$.

(ii) Initialize a hash chain. Consider P_U as the seed and generate a hash chain of length n, as shown in (I) of Figure 1:

$$P_U, h(P_U), h^2(P_U), \ldots, h^n(P_U), \tag{5}$$

where $n-1$ is an integral multiple of m_L.

(iii) Generate the next chain. Same as the first two steps, we choose a new $(t, m_L)'$-Mignotte sequence and compute S'_U and P'_U. Then, we get a new hash chain:

$$P'_U, h(P'_U), h^2(P'_U), \ldots, h^n(P'_U). \tag{6}$$

Partition $h^n(P'_U)$ into m_L 2^{l_L}-radix numbers, denoted as $c_1, c_2, \ldots, c_{m_L}$, whose repetition degree is $m_L - q_L$ and difficulty degree is q_L:

(i) Let $\alpha = x_1 \cdot x_2, \ldots, x_t$ and $\beta = x_{m_L-t+2} \cdot x_{m_L-t+3}, \ldots, x_{m_L}$. If $t = q_L$ and $\beta < h^N(P'_U) < \alpha$, then select $h^n(P'_U)$

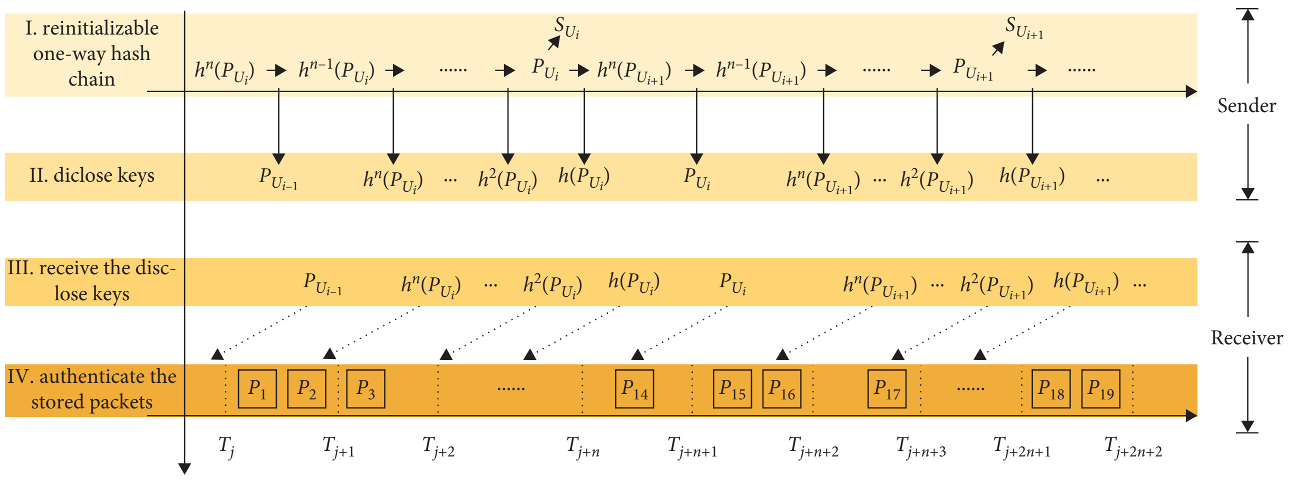
I. reinitializable one-way hash chain
II. diclose keys
III. receive the disc-lose keys
IV. authenticate the stored packets
Sender
Receiver
$h^n(P_{U_i})$
$h^{n-1}(P_{U_i})$
P_{U_i}
S_{U_i}
$h^n(P_{U_{i+1}})$
$h^{n-1}(P_{U_{i+1}})$
$P_{U_{i+1}}$
$S_{U_{i+1}}$
$P_{U_{i-1}}$
$h^2(P_{U_i})$
$h(P_{U_i})$
$h^2(P_{U_{i+1}})$
$h(P_{U_{i+1}})$
P_1 P_2 P_3 P_{14} P_{15} P_{16} P_{17} P_{18} P_{19}
T_j T_{j+1} T_{j+2} T_{j+n} T_{j+n+1} T_{j+n+2} T_{j+n+3} T_{j+2n+1} T_{j+2n+2}

FIGURE 1: The broadcasting process uses a reinitializable one-way hash key chain. Each key is used in its corresponding time interval. After distributing the seed of the previous key chain, the next key chain's commitment $h^n(P_{U_{i+1}})$ will be calculated (I). After δ time intervals, a key will be disclosed (II). When the receiver receives the corresponding key (III), the stored packets can be authenticated (IV).

as the secret S. Otherwise, return to the previous step.

(ii) According to $I_i = S \bmod x_i$, calculate m_L secret shadows $I_1, I_2, \ldots, I_{m_L}$.

(iii) Compute the key authentication codes (KAC), denoted as ζ_i and ξ_{i-1}, where $\xi_{i-1} = h^{n-i}(P_U) \oplus \zeta_i$, $\zeta_i = (x_{c_k+1}, I_{c_k+1}) \ll r_i$, $k = i \bmod m_L$, and r_i is the number made up of the i-th bit, the $2i$-th bit, the $3i$-th bit, and so on of P_U. In particular, $\xi_0 = h^{n-1}(P_U) \oplus \zeta_1$, $\zeta_1 = (x_{c_1+1}, I_{c_1+1}) \ll r_1$, and $r_1 = S_U$.

(iv) Publish $(h^n(P_U), \xi_0)$ to the verifier securely. Actually, this pair of parameters will be sent cryptographically to the nodes in Algorithm 1 or 2 of Section 5.

4.2.2. Publication. In the phase of publication, the sender computes and distributes hash values and the corresponding certification proofs for verification. For the i-th $(i = 1, 2, \ldots, n-1)$ distribution, the sender does the following steps:

(i) Compute or retrieve the link values of hash chain $h^{n-i-1}(P_U)$ and $h^{n-i}(P_U) = h(h^{n-i-1}(P_U))$

(ii) Compute ζ_i and ξ_i

(iii) Construct and publish the certification frame $(h^{n-i}(P_U), \zeta_i, \xi_i)$, while in the n-th distribution, publish the seed P_U and S_U, as shown in (II) of Figure 1

4.2.3. Verification. For the i-th $(i = 1, 2, \ldots, n-1)$ verification, the receiver does the following steps, as shown in (III) and (IV) of Figure 1:

(1) If $\lfloor T_c + \Delta - T_i/T_{\text{int}} \rfloor < i + \delta - 1$, the receiver receives the certification frame $(h^{n-i}(P_U), \zeta_i, \xi_i)$ from the sender:

(i) Compute and verify whether $h(h^{n-i}(P_U))$ is equal to $h^{n-i+1}(P_U)$, where $h^{n-i+1}(P_U)$ is a link value sent and saved in the last valid session

(ii) Compute and verify whether $h^{n-i}(P_U) \oplus \zeta_i$ is equal to ξ_{i-1}

If all checks are passed, the receiver verifies the sender successfully and then stores ζ_i. The receiver also should store ξ_i in the buffer for the next verification.

(2) If $\lfloor T_c + \Delta - T_i/T_{\text{int}} \rfloor \geq i + \delta - 1$, the receiver drops $h^{n-i}(P_U)$ and ζ_i and saves ξ_i. Then, it will wait for the next valid certification frame $(h^{n-j}(P_U), \zeta_j, \xi_j)$, where $j > i$.

(i) Compute and verify whether $h^{j-i+1}(h^{n-j}(P_U))$ is equal to $h^{n-i+1}(P_U)$, where $h^{n-i+1}(P_U)$ is a link value sent and saved in the last valid session.

(ii) Compute and verify whether $h^{n-j}(P_U) \oplus \zeta_j$ is equal to ξ_{j-1}.

If all checks are passed, the receiver verifies the sender successfully and then stores ζ_j and ξ_j for the same reason.

4.2.4. Recombination. After all hash values have been published, the whole hash chain has been exhausted, and the receiver has stored the seed P_U and m_L or fewer ζ_is:

(i) Compute and check whether ζ_{i_1} and ζ_{i_2} share the same sequence value x_{c_k+1} and the same secret shadow I_{c_k+1}, where $i_1 - i_2 \equiv 0 \bmod m_L$, $k = i_1 \bmod m_L$, and $i_1, i_2 = 1, 2, \ldots, n-1$

(ii) After verifications for all KACs, the receiver obtains q_L distinct sequence values x_{c_k+1} and q_L distinct secret shadows I_{c_k+1}, and it also gets the ordering relation of permutation of all the m_L sequence values

Then, the receiver recovers $h^n(P'_U)$ in two ways:

```
(1) for fresh node $S_i$ do
(2)   $S_i \longrightarrow S_B$: $M_{\text{a}} = ('a', \text{ID}_B, \text{ID}_i, \text{KP}_i, N_i)$
(3)   $S_B$: computes and compares the fingerprint $\text{ID}_i, \text{KP}_i$
(4)   if the fingerprint is true then
(5)     $S_B \longrightarrow S_i$: $M_{ar} = ('\text{ar}', \text{ID}_i, \text{ID}_B, A_1, \text{Sign}(A_1, \text{KS}_B))$, where $A_1 = T_c \| (h^n(P_U)_c, \xi_{0c}, \ldots)\|T_{\text{sc}}\|T_{i'}\|T_{\text{int}}\|\delta\|N_B$
(6)   else
(7)     Authentication failure
(8)   end if
(9) end for
```

Algorithm 1: Direct authentication of fresh nodes.

```
(1)  for fresh node $S_i$ do
(2)    $S_i \longrightarrow S_B$: $M_{\text{a}} = ('a', \text{ID}_B, \text{ID}_i, \text{KP}_i, N_i)$
(3)    $S_{\text{CH}} \longrightarrow S_B$: $M_{\text{aa}} = ('a', \text{ID}_B, \text{ID}_{\text{CH}}, \text{KP}_{\text{CH}}, N_{\text{CH}}, (\text{ID}_B, \text{ID}_i, \text{KP}_i, N_i))$
(4)    $S_B$: computes and compares the fingerprints $\text{ID}_i, \text{KP}_i$ and $\text{ID}_{\text{CH}}, \text{KP}_{\text{CH}}$
(5)    if the fingerprint $\text{ID}_i, \text{KP}_i$ is true then
         if the fingerprint $\text{ID}_{\text{CH}}, \text{KP}_{\text{CH}}$ is true then
(6)        $S_B \longrightarrow S_i$: $M_{\text{ar}} = ('a', \text{ID}_i, \text{ID}_B, A_2, \text{Sign}(A_2, \text{KS}_B))$, where $A_2 = 'S_{\text{CH}}\ \text{is legal}'\|T_c\|(h^n(P_U)_c, \xi_{0c}, \ldots)\|T_{\text{sc}}\|T_{i'}\|T_{\text{int}}\|\delta\|N_B$
(7)        $S_B \longrightarrow S_{\text{CH}}$: $M_{\text{arr}} = ('\text{arr}', \text{ID}_{\text{CH}}, \text{ID}_B, A_3, \text{Sign}(A_3, \text{KS}_B))$, where $A_3 = 's_i\ \text{is legal}'\|N_B$
(8)      else
(9)        $S_B \longrightarrow S_i$: $M_{\text{ar}} = ('\text{arr}', \text{ID}_i, \text{ID}_B, A_4, \text{Sign}(A_4, \text{KS}_B))$, where $A_4 = 'S_{\text{CH}}\ \text{is illegal}'\|T_c\|(h^n(P_U)_c, \xi_{0c}, \ldots)\|T_{\text{sc}}\|T_{i,}\|T_{\text{int}}\|\delta\|N_B$
(10)     end if
(11)   else
(12)     if the fingerprint $\text{ID}_{\text{CH}}, \text{KP}_{\text{CH}}$ is true then
(13)       $S_B \longrightarrow S_{\text{CH}}$: $M_{\text{arr}} = ('\text{arr}', \text{ID}_{\text{CH}}, \text{ID}_B, A_5, \text{Sign}(A_5, \text{KS}_B))$, where $A_5 = 's_i\ \text{is illegal}'\|N_B$
(14)     else
(15)       Authentication failure
(16)     end if
(17)   end if
(18) end for
```

Algorithm 2: Authentication of fresh nodes with cluster heads.

(i) Ordered verification: for all the m_L sequence values $x_{c_1}, x_{c_2}, \ldots, x_{c_{m_L}}$, the union of their subscripts is the commitment of the second chain, denoted by $h^n(P'_U)_1$. That is, $h^n(P'_U)_1 = c_1, c_2, \ldots, c_{m_L}$.

(ii) CRT verification: given t $(t = q_L)$ distinct pairs of (x_{c_k+1}, I_{c_k+1}), using CRT, the unique solution modulo $x_{c_1+1} \cdot x_{c_2+1}, \ldots, x_{c_t+1}$ of the set of equations is the commitment of the second chain, denoted by $h^n(P'_U)_2$.

If $h^n(P'_U)_1 = h^n(P'_U)_2$, then the recombination is successful and we can obtain the new hash chain $h^n(P'_U) = h^n(P'_U)_1 = h^n(P'_U)_2$.

4.2.5. Self-Renewal. After recombination, the next chain starts to work and another new chain is also generated including a pair of instances S''_U and P''_U and the corresponding commitment $h^n(P''_U)$. Iterations of the above processes have a result that hash chains work continuously and infinitely.

5. *d*-Left-Counting-Bloom-Filter-Based Authentication Scheme

In the combination of WSNs and edge computing, the base station is usually an edge server or is deployed as an unmanned aerial vehicle along with the sensor nodes, which requires the consideration of the computing and storage capabilities of the base station. These new sensor nodes need to be authenticated to make sure that they are not malicious nodes before joining the network. In the edge computing scenario, the storage space of the base station should be mainly used to cache and process the sensed data to provide data service for IoT applications. However, due to the participation of nodes, the base station needs to maintain a vast lookup table, which will occupy more storage space for authenticating the sensor nodes.

In the AdlCBF scheme, we aim to solve two problems: one is the reduction of storage overhead in the base station, and the other is the nodes' authentication and the parameters distribution of broadcast authentication when a new node joins the network. ECDSA is used to generate a signature $\text{Sign}(A, \text{KS}_A)$ with authentication information A

to be signed and the private key of the source KS_A. In order to reduce storage space, we introduce the d-left counting Bloom filter to construct a dlCBF to store the fingerprint for each node instead of allocating storage space directly.

5.1. Construction of dlCBF. In DH-μTESLA, every sensor node S_i holds a public key KP_i and a private key KS_i allocated by the base station S_B. The base station constructs a dlCBF to store the fingerprint of ID and key information instead of storing them directly. There are three steps to construct a dlCBF, as shown in Figure 2.

Step 1. The base station applies a hash function $H(\cdot)$ to map each node's pair of ID and public key $\langle \mathrm{ID}_i, \mathrm{KP}_i \rangle$ to the true fingerprint $f_{S_i} = H(\mathrm{ID}_i \| \mathrm{KP})_i = (b_i, r_i)$. Each true fingerprint consists of two parts. The first part represents the bucket index b_i which corresponds to the storage location, while the second part r_i is the real practicable element to be stored in the corresponding array of b_i.

Step 2. The base station uses the additional pseudorandom permutations $P_1, P_2, \ldots, P_d$ to expand d locations, which are the d replaceable choices for each true fingerprint f_{S_i}. As shown in Figure 2, they are actually d subarrays $P_1(f_{S_i}) = (b_{i1}, r_{i1}), P_2(f_{S_i}) = (b_{i2}, r_{i2}), \ldots, P_d(f_{S_i}) = (b_{id}, r_{id})$, where $P_j(f_{S_i}) = (b_{ij}, r_{ij})$ represents the tuple of storage location and practicable element corresponding to the j-th choice of f_{S_i} $(j = 1, 2, \ldots, d)$. There exist two kinds of collisions in dlCBF. One is that two different fingerprint replacements map to the same location of subarray; that is, $P_1(f_{S_i}) = (b_{i1}, r_{i1}) \neq P_1(f_{S_i}) = (b_{j1}, r_{j1})$; however, $b_{i1} = b_{j1}$, which means that the bucket of each subarray needs more storage cells for different elements. The other is that two true but different fingerprints have the same replacement; that is, $f_{S_i} \neq f_{S_j}$; however, $P_d(f_{S_i}) = P_d(f_{S_j})$. In this case, a counter is needed to record the number of the same elements stored. So, the storage cost of bucket is the sum of all counting of its storage cells.

Step 3. The base station selects the leftmost one simultaneously with the minimum storage cost as the final storage location from the d choices.

5.2. Authentication Process of Fresh Nodes. When a fresh sensor node applies to join the network, it should be authenticated and only the legal one can acquire the key chain commitment and other configuration information. Because the participation of new nodes always occurs during the whole network lifetime, the authenticating process appears before and after the network initialization phase. In other words, there are two kinds of fresh nodes: one can communicate with the base station directly and the other one is not close to it. Thus, two different cases will be discussed, respectively. In the former case, the base station authenticates the fresh node directly by Algorithm 1. Meanwhile, in the latter case, the fresh node will be authenticated through cluster head by Algorithm 2.

Both Algorithms 1 and 2 only describe the authentication process of single fresh node. When multiple fresh sensor nodes participate in the network synchronously, the cluster heads will aggregate the authentication application messages to generate only one revised authentication message M_{aa} and then send it to the base station that can batch-process them. Thus, these two algorithms can also be used in the multiauthentication cases.

6. DH-μTESLA Protocol

Based on the above schemes, SRHC-TD and AdlCBF, there are five phases to describe the proposed protocol: setup, bootstrap, broadcast, authentication, and a new chain generation. We explain how the base station broadcasts message at the beginning of the network.

6.1. Setup. The base station first generates a sequence of secret messages using SRHC-TD scheme, which is described in Section 4.

6.2. Bootstrap. Any receiver in the network should have the commitment and relative parameters of the reinitializable one-way hash chain acquired by Algorithms 1 and 2 in Section 5, including the length of hash chain n, a secure hash function h: $\{0,1\}^* \longrightarrow \{0,1\}^L$, and the partition pair (m_L, l_L). Besides, the legal node also gains other initial parameters, such as the current time T_c, the maximum clock difference δ between the sender and the receiver, the start time T_i of interval i, and the duration of each time interval T_{int}.

6.3. Broadcast. The lifetime of a sensor node, which is much longer than that of one-way hash chain, is divided into fixed intervals of duration T_{int}. The sender uses the key value of hash chain to compute the message authentication code (MAC) of message packets in the current time interval. Then, the sender broadcasts the packet with MAC in the same time interval and discloses a key value of hash chain with corresponding key authentication code (KAC) after a certain delay $\delta \times T_{\mathrm{int}}$.

6.4. Authentication. When a node receives a message packet with the MAC, it stores the packet and the MAC in the buffer. Once the node receives a key disclosure packet with KAC, the sensor node first verifies the key value with KAC, which is related to the message packet that has been stored, as described in Section 4.

6.5. Generate a New Chain. When the hash chain runs out, the base station should generate the next chain and the node needs to recover the commitment of the new chain.

7. Security Analysis

7.1. Resistance with Chosen Plaintext Attack. Except the initial seed and the commitment, each key of a hash chain is not only the ciphertext (as the output) but also the plaintext (as the

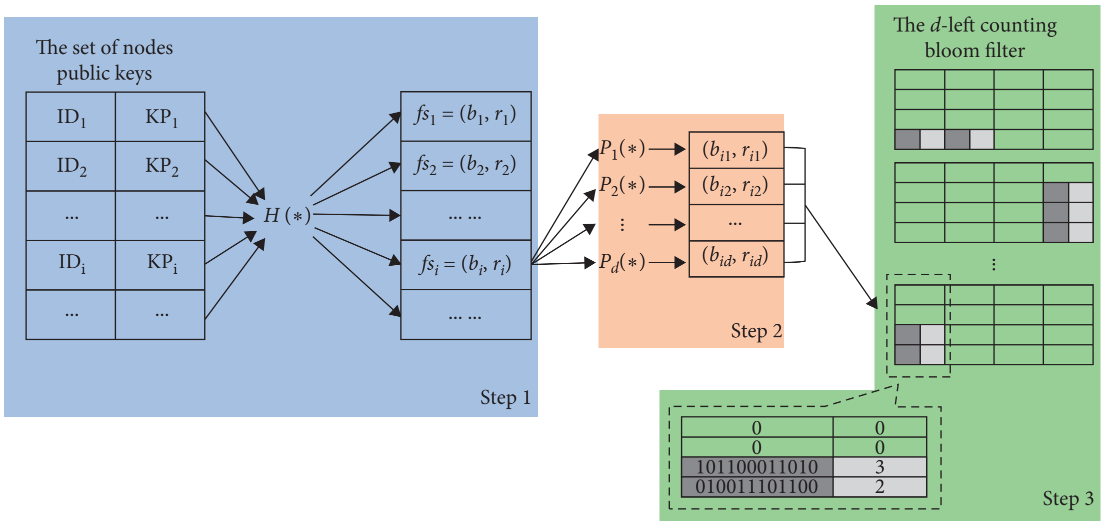

FIGURE 2: The construction of d-left counting Bloom filter.

input) of the hash function. Thus, each value shares the same length, which reduces the key space as well as the cracked time. Thus, it is easy to suffer from the chosen plaintext attack. In our scheme, the KAC consists of two parts, which can be used to check whether the shared secret has been changed. All ζ_i are different because of the left shift operation, which can prevent the chosen plaintext attacks effectively.

7.2. Fault Tolerance. The length of key chain $n-1$ is an integral multiple of m_L, so all (x_{c_i+1}, I_{c_i+1}) can be transmitted cyclically when the network is running. Thus, it can avoid the situation where one missing secret shadow would make the secret $h^n(P_U')$ unsolvable. Before a hash chain is running out, each receiver stores m_L (x_{c_i+1}, I_{c_i+1}) at most, while only q_L distinct sequence values $x_{(c_i+1)}$ and secret shadows $I_{(c_i+1)}$ are needed to recover the secret $h^n(P_U')$. In other words, the number of faults or missing message packets should be less than $m_L - q_L$. The highest packet loss rate that our solution can tolerate is $(m_L - q_L)/m_L$.

Assuming that the probability that a sensor node cannot receive a key disclosure packet is f and $n-1 = k\cdot m_L$ which means that the same secret shadow will be disclosed k times, the probability that a certain secret shadow I_{c_i+1} is not received by the sensor node is reduced to f^k. Suppose that the sensor node has a half chance to receive the packet (it will be lower in practice); that is, $f = 0.5$ and $k = 10$. The probability that the sensor node loses I_{c_i+1} is less than 0.1%. In our scheme, in order to recover the commitment of next hash chain, only q_L secret shadows are needed. Thus, the probability that the sensor nodes receives q_L different secret shadows is $C_{m_L}^{q_L}(1-f^k)^{q_L}\cdot(f^k)^{m_L-q_L}$. Then, the probability that the sensor node cannot obtain the commitment of next hash chain is

$$p_d = 1 - C_{m_L}^{q_L}\left(1-f^k\right)^{q_L}\cdot\left(f^k\right)^{m_L-q_L}. \quad (7)$$

If the length of hash chain is 321, which can cover about 5 minutes with time interval of 1 second, $q_L = 16$ and $m_L = 32$; the probability that the sensor node cannot recover the commitment of next hash chain when the hash chain runs out is $p_d \le 5.9\times 10^{-40}$.

7.3. DoS Attack Tolerance. Both the base station and the sensor node are vulnerable to DoS attack that is hard to prevent. The attacker may forge the message, such as key distribution message and authentication message, to confuse both the sensor node and the base station.

In Multilevel μTESLA, when the lower level chain draws to an end, the upper level is used to authenticate the commitment for the next lower level chain by commitment distribution message (CDM),

$$\mathrm{CDM}_i = \left\|K_{i+2,0}\right\|\mathrm{MAC}_{K_i}\left(i\left\|K_{i+2,0}\right)\right\|K_{i-1}, \quad (8)$$

in its i-th interval, where $\|$ denotes concatenation, $K_{i+2,0}$ is the commitment of the lower level chain, and K_{i-1} or K_i is a key of the upper level chain. The attacker can fabricate a CDM by replacing $K_{i+2,0}$ or $\mathrm{MAC}_{K_i}(i\|K_{i+2,0})$ and the sensor node can easily verify K_{i-1} with K_{i-2} which is released in CDM_{i-1}. Nevertheless, the sensor node has no way to authenticate other parameters in CDM_i because the key will be released later. In order to solve this problem, Multilevel μTESLA provides two variants, needing more buffers (hundreds of bytes) to store multiple CDMs in sensor node or requiring more storage space for additional precomputed chains in the base station and larger payload of CDMs. Meanwhile, in our protocol, the commitment of next chain is released with the distribution of keys. When a sensor node received a certification frame $(h^{n-i}(P_U), \zeta_i, \xi_i)$, it could be authenticated immediately by a single XOR operation. The sensor node can distinguish the validity of the message at a very fast speed without a little buffer (about 8 bytes). With the same condition in Section 7.2, assume that the bandwidth is 10 kbps and the hash function is 64 bits; the relative communication overhead is just $64\cdot 2/10240\cdot 1 = 0.0125$.

On the other hand, a fresh sensor node needs to be authenticated before joining the network. An attacker may launch the flood attack toward the base station with forged messages containing valid public keys, which can be obtained by eavesdropping. However, the attacker cannot alter the ID information of M_{a} or M_{aa}. Even if the ID can be forged, after receiving the same authentication message multiple times, the base station will find that it is being attacked. Meanwhile, a sensor node will be aware that an attack is going on and alert the base station when it fails to receive the authentication response message M_{ar} or receive the wrong responses multiple times in one round. Also, when a cluster head receives the authentication message M_{a}, the authentication response message M_{ar}, or added authentication response message M_{aar} multiple times from the same node, it will be aware that an attack is going on and alert the base station.

7.4. Provable Security. Because SRHC-TD is formed by many chains, we will discuss the provable security of SRHC-TD with two cases: one is the successive key in the same hash chain; and the other is the head-tail key connecting the current hash chain and the next one.

We choose the Random Oracle model to analyse the provable security of SRHC-TD. In the Random Oracle model, attackers have the polynomial calculation ability of any secret parameter k. The reliability of all algorithms of SRHC-TD is determined by the secret parameter k. Specifically, the probability of cracking hash functions is the reciprocal regarding the exponential function of k, the probability of cracking CRT is the reciprocal regarding the power function of k, and the probability of cracking partition algorithm is the reciprocal regarding the logarithmic function of k:

(a) After the ith time interval, we suppose that the receivers (including the attackers) have received $(h^{n-i}(P_U), \zeta_i, \xi_i)$. Attackers must forge h_{fake}^{n-i-1} and ζ_{i+1} within the $(i+1)$th time interval and make them meet the equations $h(h_{\text{fake}}^{n-i-1}) = h^{n-i}(P_U)$ and $h_{\text{fake}}^{n-i-1} \oplus \zeta_{i+1} = \xi_i$. Suppose that the calculation ability of attackers can be represented as the polynomial $T_{\text{adv}}(k)$ and they can query Oracle any number of times within each time interval. If the probability of cracking hash operation is e^{-k}, the probability of breaking SRHC-TD can be expressed as

$$\begin{aligned}\Pr[\text{Adv}(k)=1] &= \Pr\left[h_{\text{fake}}^{n-i-1} = h^{n-i-1}(P_U), h_{\text{fake}}^{n-i-1} \oplus \zeta_{i+1} = \xi_i\right] \\ &= \frac{T_{\text{adv}}(k)}{e^k \cdot e^k},\end{aligned} \tag{9}$$

where $T_{\text{adv}}(k)$ can be expressed as the general form of polynomial. So, we can deduce $\Pr[\text{Adv}(k)=1] = a_n k^n + a_{n-1}k^{n-1} + \cdots + a_1 k + a_0/e^{2k}$. Evaluating the limit value of $\Pr[\text{Adv}(k)=1]$, we can deduce

$$\begin{aligned}\lim_{k\longrightarrow\infty} \Pr[\text{Adv}(k)=1] &= \lim_{k\longrightarrow\infty} \frac{a_n k^n}{e^{2k}} = \lim_{k\longrightarrow\infty} \frac{a_n \cdot n k^{n-1}}{2\cdot e^{2k}} \\ &= \cdots = \lim_{k\longrightarrow\infty} \frac{a_n \cdot n!}{2^n \cdot e^{2k}} = 0.\end{aligned} \tag{10}$$

(b) At the ends of the $(n-1)$-th time interval, only P_U has not been published in the first hash chain. The receivers (including the attackers) have obtained m_L number of ζ_i. Attackers must forge P_U and $P_{U_{\text{fake}}}$ within the nth time interval and make them meet the following conditions: (1) $h(P_{U_{\text{fake}}}) = h(P_U)$; (2) when $i_1 - i_2 \equiv 0 \bmod m_L$, ζ_{i_1} and ζ_{i_2} contain the same sequence value x_{c_k+1} and the same shadow I_{c_k+1}, where $k = i_1 + 1 \bmod m_L$, $i_1, i_2 = 1, 2, \ldots, n-1$; (3) the solution $h^n(P'_U)_1$ derived from the proof method of ordering is the same as the solution $h^n(P'_U)_2$ derived from the proof method of CRT. The fake value $P_{U_{\text{fake}}}$ must meet all the three conditions. We suppose that the calculation ability of attackers can be represented as the polynomial $T_{\text{adv}}(k)$ and they can query Oracle any number of times within each time interval. If the probability of cracking hash operation is e^{-k}, the probability of cracking CRT is k^{-e}, and the probability of cracking partition algorithm is $1/\ln k$; then the probability of breaking SRHC-TD can be expressed as

$$\Pr[\text{Adv}(k)=1] = \Pr\begin{bmatrix} h\left(P_{U_{\text{fake}}}\right) = h(P_U), h^n(P'_U)_1 = h^n(P'_U)_2 \\ x_{\left(c_{i_1}+1\right)\bmod m_L+1} = x_{\left(c_{i_2}+1\right)\bmod m_L+1}, I_{\left(c_{i_1}+1\right)\bmod m_L+1} = I_{\left(c_{i_1}+1\right)\bmod m_L+1}\end{bmatrix} = \frac{T_{\text{adv}}(k)}{e^k \cdot k^e \cdot \ln k}, \tag{11}$$

where $T_{\mathrm{adv}}(k)$ can be expressed as the general form of polynomial $T_{\mathrm{adv}}(k) = a_n k^n + a_{n-1}k^{n-1} + \cdots + a_1 k + a_0$. Evaluating the limit value of $\Pr[\mathrm{Adv}(k) = 1]$, we can deduce

$$\begin{aligned}\lim_{k\longrightarrow\infty} \Pr[\mathrm{Adv}(k) = 1] &= \lim_{k\longrightarrow\infty} \frac{a_n k^n}{e^k \cdot k^e \cdot \ln k} \\ &= \lim_{k\longrightarrow\infty} \frac{a_n k^{n-e}}{e^k \cdot \ln k} = \lim_{k\longrightarrow\infty} \frac{a_n(n-e)k^{n-e-1}}{e^k \cdot (\ln k + 1/k)} = \cdots \\ &= \lim_{k\longrightarrow\infty} \frac{a_n(n-e)!}{e^k \cdot \left(\ln k + n - e/k - \cdots + (n-e-1)!/k^{n-e}\right)} \\ &= \lim_{k\longrightarrow\infty} \frac{a_n(n-e)!}{e^k \cdot \ln k} = 0.\end{aligned} \tag{12}$$

Considering the above two cases, there exists a positive integer N, when $k > N$; for any ε, there must be $\Pr[\mathrm{Adv}(k) = 1] < \varepsilon$. So, the successful probability of attackers can be ignored and the SRHC-TD scheme has provable security.

8. Performance Evaluation

8.1. Evaluation of SRHC-TD. There are two possible ways to store a hash chain in base station. One is to generate a chain and allocate specialized space to store the whole chain. The other one is to store the seed of a chain and compute the link value when it needs to be used. Obviously, the former reduces the computation consumption but increases the memory consumption, while the latter is on the contrary.

Considering the above two situations, the consumptions of SRHC-TD will be compared with those of RHCs [28–30] from two aspects, computation and communication, as follows. Table 2 shows the variable symbols and the corresponding variable names. Table 3 shows the consumptions of SRHC-TD and RHCs in detail.

For ease of analysis, we assume that the length of the hash chain is greater than the output length of the hash function; that is $n > m$. In fact, the length of the hash chain is usually around 1000 because a short chain will lead to frequent regeneration or failure due to excessive packet loss rate. It is remarkable that if the base station only stores seeds, the calculation time increases exponentially. The length of hash chains n is a significant factor affecting the computation time.

From Table 3, we can see that the storage method of the hash chain has no effect on the communication overhead. In the initialization phase, the above schemes have the same communication overhead, while in the publication, verification, and recombination phases, our scheme has the same overhead as SRHC and is lower than RHC and SUHC, as shown in Figure 3. To evaluate the performance of our scheme in terms of computation overhead, we implement the operation used in the proposed scheme and other schemes on an Ubuntu 12.04 virtual machine with an Intel Core i5-4300 CPU @ 2.60 GHz. We use the 64-bit version of RC5 and generate a Mignotte's sequence whose length is 32. In this case, each hash operation takes 0.0037 ms. Besides, the times consumed by shifting, XOR, and modular operation, denoted as $T(M)$, $T(X)$, and $T(I)$, are 0.0002 ms, 0.0002 ms, and 0.0003 ms, respectively. The generation of a 32-bit random number and $T(B)$ takes 0.0004 ms, and it takes 0.0161 ms to recover the commitment. The time consumed in computing r_i is so little that it can be ignored.

Figure 4 shows that, in the initialization phase, the time taken by our scheme increases by approximately 0.26 ms compared to the other three schemes, due to the generation of sharing secrets. However, in publication, verification, and recombination phases, our scheme takes less time, especially when the base station stores the entire hash chain, as shown in Figure 5. If the hash chain contains 1000 keys and the base station stores the entire hash chain, compared to [28–30], the computation overhead of SRHC-TD is reduced by 39.07%, 77.08%, and 63.28%, respectively. If the base station only stores the seed of the hash chain, our scheme's computation overhead is reduced from 0.2% to 0.81%. It is worth noting that when the base station stores the entire chain, the computation time is 1000 times that of the case when it stores the seed. The main reason for this result is that when the base station releases the key which the sensor node used to authenticate the broadcast message, it needs to recover the key first. The base station needs to hash the seed value many times when the base station only stores the seed. However, if the base station stores the entire hash chain, which takes up more storage space, the key can be found easily. Thus, a more efficient way is that the base station stores some "checkpoints" for recovering the key. The greater the number of checkpoints stored by the base station is, the closer the time consumption is, according to Figure 5(a).

Thus, it is easy to compare the consumptions of SRHC-TD with the RHCs, as shown in Table 4. It shows that the SRHC-TD has a better performance in *publication*, *verification*, and *recombination* phase. The computation and communication overhead of SRHC-TD are much less than other schemes, especially when the node only stores the seed of hash chain.

Table 2: The variable symbols and the corresponding variable names.

Symbols	Description
L	The output length of hash function
n	The length of a hash chain
m	The length of Mignotte's sequence
H	The computational overhead of a hash function
R	The computational overhead of generating a random number
B	The computational overhead of mapping a random number to one single bit using Hard Core Predicate
I	The computational overhead of generating a secret shadow I_i in SRHC-TD
P_r	The computational overhead of computing r_i in SRHC-TD
M	The computational overhead of a left shift operation
X	The computational overhead of an XOR operation
C	The computational overhead of computing solutions to equations

Table 3: The consumptions in SRHC-TD and the RHCs.

Storage method			The whole chain	The seed
RHC [28]	Initialization	Computation	$(2n+3)\cdot H+2R$	$(3n+2)\cdot H+2R$
		Communication	$2L$	$2L$
	Publication and verification and recombination	Computation	$(2n+3)\cdot H+2R$	$0.5(n^2+3n-4)H+2(n-1)R$
		Communication	$4nL$	$4nL$
SUHC [29]	Initialization	Computation	$2(n+1)\cdot H+3R+B$	$3n\cdot H+3R+B$
		Communication	$2L$	$2L$
	Publication and verification and recombination	Computation	$5n\cdot H+n\cdot R+2nB$	$0.5(n^2+9n+2)\cdot H+n\cdot R+2nB$
		Communication	$4nL$	$4nL$
SRHC [30]	Initialization	Computation	$2(n+1)\cdot H+3R$	$3nH+3R$
		Communication	$2L$	$2L$
	Publication and verification and recombination	Computation	$3n\cdot H+n\cdot R+2nB$	$0.5(n^2+5n+2)\cdot H+n\cdot R+2nB$
		Communication	$3nL$	$3nL$
SRHC-TD	Initialization	Computation	$2(n+m)\cdot H+2m\cdot R+mI+X+M+P_r$	$(3n+2m-1)\cdot H+2m\cdot R+mI+X+M+P_r$
		Communication	$2L$	$2L$
	Publication and verification and recombination	Computation	$nH+2nM+2nX+2nP_r+C$	$0.5(n^2+n)\cdot H+2nM+2nX+2nP_r+C$
		Communication	$3nL$	$3nL$

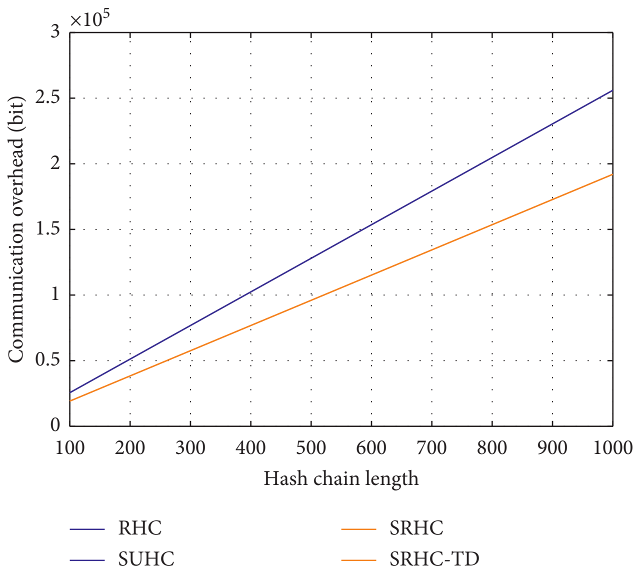

Figure 3: Comparison of communication overhead in publication, verification, and recombination phases.

As we know, the largest contributor to energy consumption in WSNs is communication complexity, and computation overhead is the second, where the energy cost of sending and receiving a packet is several times that of computing or processing a packet [41]. The *publication*, *verification*, and *recombination* phases last much longer than the *initialization* phase over the lifetime of the network, and the *initialization* phase runs in the base station. Although our scheme has a higher consumption overhead in *initialization* phase, our energy consumption is lower than other schemes on the whole.

8.2. Evaluation of AdlCBF. In AdlCBF scheme, the fresh node can be authenticated by the base station using the ECDSA signature and the fingerprint of the node. ECDSA is a light-weight algorithm and it is based on the elliptic curve. The sensor node (or cluster head node) only needs to generate and verify a signature once when it joins the network. Thus, the overhead of authenticating a fresh sensor node is acceptable. Besides, the *d*-left counting Bloom filter is

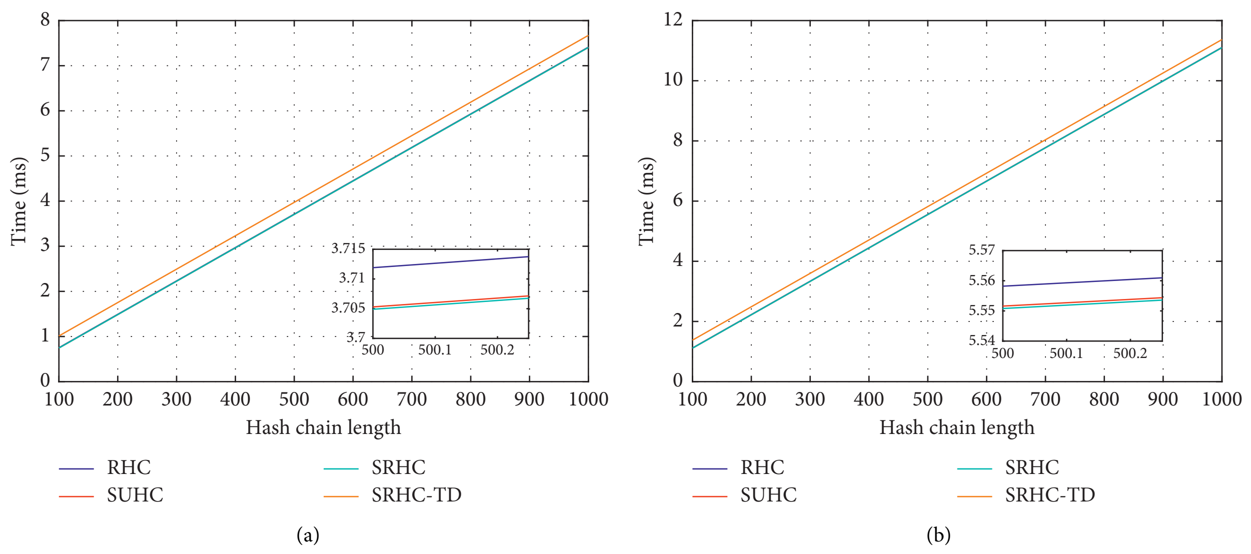

Figure 4: Comparison of time consumption in initialization phase. (a) The base station stores the whole hash chain. (b) The base station only stores the seed of hash chain. The lower right corner of the figure is a partial enlargement.

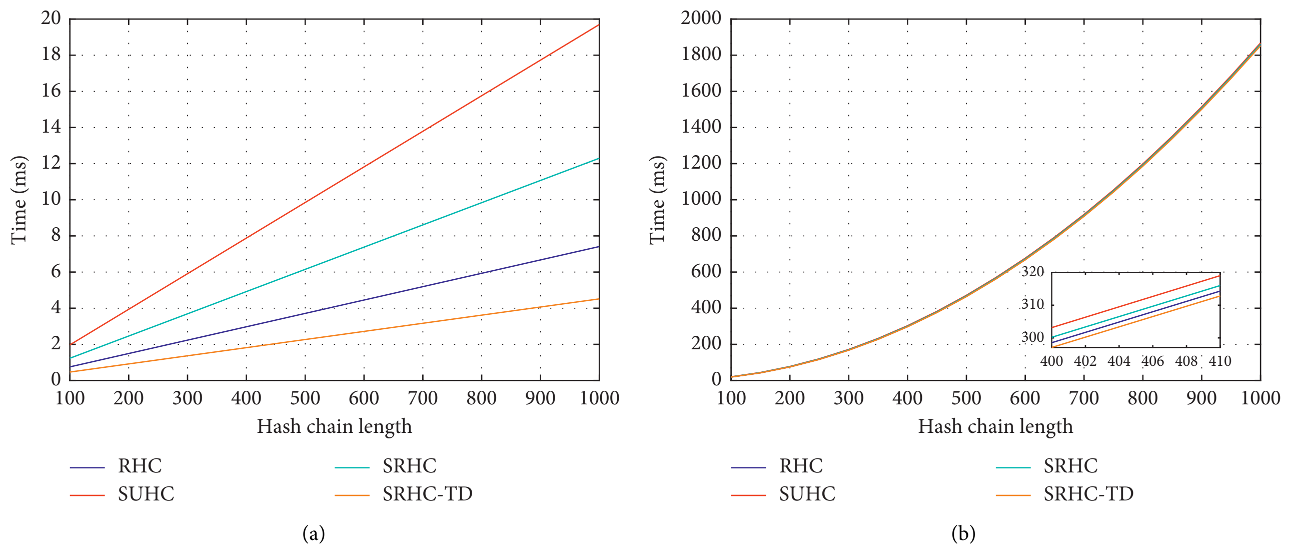

Figure 5: Comparison of time consumption in publication, verification, and recombination phases. (a) The base station stores the whole hash chain. (b) The base station only stores the seed of hash chain. The lower right corner of the figure is a partial enlargement.

Table 4: The comparisons of SRHC-TD and the RHCs.

Storage method		The whole chain	The seed
Initialization	Computation	SRHC < SUHC < RHC < SRHC-TD	SRHC < SUHC < RHC < SRHC-TD
	Communication	SRHC-TD = RHC = SRHC = SUHC	SRHC-TD = RHC = SRHC = SUHC
Publication and verification and recombination	Computation	SRHC-TD < RHC < SRHC < SUHC	SRHC-TD < RHC < SRHC < SUHC
	Communication	SRHC-TD = SRHC < RHC = SUHC	SRHC-TD = SRHC < RHC = SUHC

Table 5: The basic simulation configurations.

Parameters	Value
Elements number	$m = 500000$
Subsequences number	4
Bucket number per subsequence	$m/24 = 20833$
Average load per bucket	6
Cells number per bucket	8
Counter length	2 bits
Storage element length	$r = 14$ bits
Hash function	MD5
Pseudorandom permutation	$P_i(H(x)) = aH(x) \bmod 2^q$
Proportional parameter in permutation a	A random odd number below $[2^q]$

adopted to reduce the storage space. We will compare AdlCBF with Direct Access and CBF, respectively, regarding the storage overhead.

8.2.1. Comparison with Direct Access. The basic simulation configurations of constructing a dlCBF are shown in Table 5, where q stands for a large prime number.

Suppose that the length of public key KP_A in ECDSA is x; then, $2^x - 1 \geq n$, where n is the maximum on the elliptic curve. To guarantee the uniqueness of each node's public key, let $n \geq m$. Thus, we can obtain $x \geq \log_2(n+1) \geq \log_2(m+1) = 18.93$, and the minimum length of public key is 19 bits. In the same way, the minimum length of ID is 19 bits, too. So, if the 500000 elements are stored with the form of plaintext, the required memory space is $500000 \times (19 + 19) = 19000000$ bits. When the 500000 elements are stored in AdlCBF, the required memory space is $4m(r+2)/3 = 10666667$ bits. Thus, the former one is 1.78125 times more than the latter one. In conclusion, AdlCBF has better storage efficiency than the Direct Access.

On the constructed AdlCBF, we repeat operations of adding an element, querying an element, and deleting an element, respectively, 3000 times. We record the consumed time for each operation and calculate the average value of the recorded time for every 300 times. Thus, one set of tests is accomplished and then we keep on conducting the remaining 9 tests similarly. The result is shown in Table 6. The more intuitive representation of changes in consumption time is shown in Figure 6.

As shown in Table 6, the query time is actually the authenticating time in AdlCBF, whose changes are quite small, relative to addition and deletion. In AdlCBF, the main operations are done by CPU, while in the Direct Access, the main operations are done by RAM. As we know, the read-write speed of CPU is two or three orders of magnitude faster than RAM. According to the space and the query time of AdlCBF, we can obtain that the required time in the Direct Access is in the range of 1.78 ms to 17.81 ms. In conclusion, AdlCBF has more satisfactory query efficiency.

8.2.2. Comparison with CBF. The upper bound on false positive probability of AdlCBF is $(1/2)^r \times 2 \times 4 \times 6 = 24 \times 2^{(-r)}$, while the false positive probability of CBF is $(2^{-\ln 2})^c$, where the standard CBF uses cm counters for tracking m elements and each counter is 4 bits [42]. When $c = (r+2)/3$, the two approaches have the same amount of space. However, when considering the false positive probability ratio of CBF to AdlCBF, we obtain $(2^{-\ln 2})^{r+2/3}/(24 \times 2^{-r}) = 52.65$. Thus, with the same storage space, the false positive probability of the standard CBF is larger than that of AdlCBF.

The storage space of AdlCBF is $4\,\mathrm{m}(r+2)/3$, while the total bits of CBF is $4\,\mathrm{cm}$. When $(2^{-\ln 2})^c = 24 \times 2^{-r}$, the two approaches share the same false positive probability. However, when considering the space ratio of CBF to AdlCBF, we obtain $((4\,\mathrm{m} \cdot \ln(24 \times 2^{-r}))/(4\,\mathrm{m}(r+2)/3)) = 2.55$. Thus, with the same false positive probability, the storage cost of the standard CBF is larger than that of AdlCBF.

8.3. Comparison with Other Variants

8.3.1. Overhead. In this section, we discuss in detail the similarities and differences with Multilevel μTESLA [14] and Scalable μTESLA [22] in terms of the storage and computation requirement.

To concentrate on the comparison of storage and computation overhead, we fix the following parameters. First, we establish a two-level μTESLA, where the duration of each low-level time interval is 100 ms, and each low-level key chain consists of 600 keys. Thus, if the top-level hash chain has 300 keys, the two-level μTESLA can cover 300 minutes, where the hash function and MAC are both 64 bits. Second, we set 300 hash chains and each hash chain has 600 keys in Scalable μTESLA. Finally, in our protocol, we set the hash chain to be the same as the above hash chain, and the threshold is set to [16, 32] where each secret fragment of the new commitment will be distributed about 20 times. For clarity and ease of analysis, we assume that the packet rate is 100 per minute and these packets are sent continuously.

(1) Storage Overhead. To generate a new hash chain and obtain the commitment used in the next period, all these protocols have to buffer data in sensor nodes. In the two-level μTESLA, the sensor node uses 624-byte storage space to buffer 39 CDMs, which can achieve authenticating the commitment of next low-level hash chain with a nearly 100% probability, because the sensor node has no way to confirm whether the MAC value in CMD is modified before the

Table 6: The statistics of consumed time.

	Addition	Query	Deletion
Test 1	0.23	0.08	0.32
Test 2	0.21	0.07	0.35
Test 3	0.31	0.13	0.40
Test 4	0.25	0.12	0.42
Test 5	0.20	0.10	0.35
Test 6	0.27	0.10	0.39
Test 7	0.30	0.09	0.30
Test 8	0.24	0.08	0.38
Test 9	0.21	0.07	0.35
Test 10	0.23	0.09	0.36
Average time	0.25	0.10	0.36

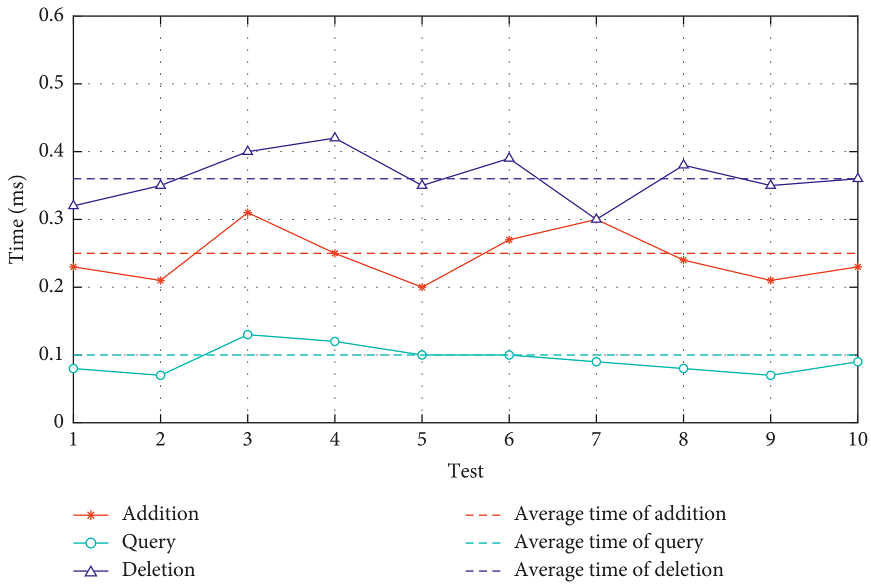

Figure 6: The statistics of consumed time.

corresponding key is released. In the Scalable μTESLA, each sensor node only needs to store a hash value, the root of Merkel hash tree, which occupies 8 bytes. If there are more than one potential sender in the network, the sensor node needs to store two roots of Merkel tree occupying 16 bytes. Meanwhile, in our scheme, the commitment of next hash chain is divided into 32 secret parts and each part is distributed along with keys used to authenticate the broadcast packets. We have to buffer 32 ζ_i's to recover the commitment of a new hash chain which only occupied 256 bytes.

(2) Computation Overhead. We focus on the computation overhead of authenticating or generating the commitment used in the next period of time and authenticate the disclosed key. In Multilevel μTESLA, when the sensor node receives a CDM, the sensor node will buffer it with a certain probability. For each buffered CDM, the sensor node needs to authenticate the top-level key first by a hash operation and the MAC will be authenticated when the corresponding key is released after a certain delay. Thus, during 300 minutes, the computation overhead is $300 \cdot 39 \cdot (T(H) + T(\text{MAC})) + 300 \cdot 100 \cdot T(H)$, where $T(\cdot)$ represents the consumed time of the operation. In the Scalable μTESLA, the sensor node needs $1 + \lceil \log_2 300 \rceil = 10$ hash operations to verify the commitments according to the root of Merkel hash chain. Thus, the computation overhead is $300 \cdot (1 + \lceil \log_2 300 \rceil) \cdot T(H) + 300 \cdot 100 \cdot T(H)$. However, when there are n potential senders in the network, the sensor node needs to verify the validity of the sender with additional $1 + \lceil \log_2 n \rceil$ hash operations first before verifying the commitment. In our proposal, part of the commitment is sent with the distribution of the corresponding key. For each distribution, the sensor node only needs a XOR operation to authenticate the commitment fragment and recover the commitment at the end of the hash chain. Therefore, the computation overhead is $300 \cdot 100 \cdot (T(H) + T(X)) + 299 \cdot T(C)$.

As shown in Table 7, in comparison with Multilevel μTESLA, our solution has less overhead due to the simple and immediate operation of authenticating the commitment of next hash chain. We have a similar computation overhead to that of Scalable μTESLA. Although Scalable μTESLA has smaller storage overhead, the sensor node needs more computation before verifying the commitment when there are

Table 7: Comparisons with other variants.

	Storage overhead (bytes)	Computation overhead (ms)
Multilevel μTESLA	624	197.5800
Scalable μTESLA*	8	122.1000
Ours	256	121.8139

*There is only one potential sender.

Table 8: Comparisons of various broadcast authentication protocols.

	Our scheme	μTESLA	Multilevel μTESLA	Long-duration TESLA	Scalable μTESLA
Structure	Single-chain	Single-chain	Multilevel	Multilevel	Merkel chain
Without using overlong hash chain	√	×	×	×	√
Reinitializable	√	×	×	√*	×
Backward security	√	√	√	×	√
Resistance with chosen plaintext attack	√	×	√	×	×
Resistance with DoS attack (sensor node)	√	×	√	×	√
Resistance with DoS attack (base station)	√	×	×	×	×
Fresh node authentication (by base station)	√	×	×	×	×
Fault tolerance	√	√	√	×	√

*The length of top-level hash chain is unlimited.

multiple potential senders, even if some of these senders only send the message once. More importantly, we eliminate the overlong or redundant hash chains that exist in Multilevel μTESLA and Scalable μTESLA in order to ensure that the hash chains are not exhausted before the whole network dies.

8.3.2. Features. Separately, we further compare DH-μTESLA with μTESLA [13], Multilevel μTESLA [14], Scalable μTESLA [22], and long-duration TESLA [33] in terms of some important aspects, and the details are shown in Table 8.

Our protocol seems more practical, since the structure of hash chain is simple while maintaining reinitialization and backward security. The long-duration TESLA is not secure once a sensor node is compromised. The attacker can forge the authentication message based on the information stored by the node due to the forward hash chain. In order to prolong the lifecycle of the network, long-duration TESLA uses an overlong hash chain which may weaken the security of the hash chain. Scalable μTESLA eliminates this threat using multiple short hash chains; however, the hash chain is not reinitializable.

In μTESLA and its variants, after receiving the broadcast message, the sensor node needs to buffer the message because it cannot be authenticated immediately. Attackers can cause huge damage to the network through DoS attacks. Therefore, resistance to DoS attacks is an important feature. In order to prevent nodes from DoS attacks, Multilevel μTESLA, Scalable μTESLA, and our proposal give a solution. However, none of these variants can handle the authentication of fresh node and DoS attack on the base station, which are important features of our protocol.

Furthermore, all protocols have a good fault tolerance except long-duration TESLA. In μTESLA, if a key disclosure packet of a broadcast message is lost, it also can be authenticated by the subsequent key disclosed. Multilevel μTESLA and Scalable μTESLA can handle the situation where the commitment of next hash chain is lost by broadcasting the same packet that contains the commitment periodically.

9. Conclusion

Our paper focuses on the key management and distribution of broadcast authentication in the combination of WSNs and edge computing with consideration of efficient storage and security issue caused by overlong hash chains, which are also important issues in IoT application scenarios. We propose an enhanced broadcast authentication protocol DH-μTESLA based on μTESLA, including SRHC-TD and AdlCBF schemes. Motivated by the fact that the disadvantages of μTESLA are the limited length of the hash chain and the security issue caused by an overlong hash chain, we design a (t, n)-threshold-based self-reinitializable hash chain scheme (SRHC-TD), which can withstand chosen plaintext attacks and is proven to be secure in Random Oracle model. Besides, in order to keep the scalability of the network, we put forward the d-left-counting-Bloom-filter-based authentication scheme (AdlCBF) and the base station is able to defend against DoS attack. The test simulation shows that, compared to other renewable hash chain schemes, our protocol has less energy consumption and the base station requires less storage space than Direct Access and CBF. Furthermore, compared to Multilevel μTESLA and Scalable μTESLA, our protocol takes less computation overhead in sensor node. In the end, we discussed the advantages of our protocol versus μTESLA, Multilevel μTESLA, long-duration TESLA, and Scalable μTESLA.

There are still some issues worthy of further study in the future work. First, in our paper, we suppose that there is only one base station in the entire network. However, in some very large-scale sensor networks, there may be multiple base stations and the fresh node needs to decide which base station to join. Second, broadcast authentication in IoT with high packet loss rate is still an open issue which would be addressed effectively.

Data Availability

Data are available upon request to the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61672297), the Key Research and Development Program of Jiangsu Province (Grant no. BE2017742), the Sixth Talent Peaks Project of Jiangsu Province (Grant no. DZXX-017), the National Key Research and Development Program (Grant no. 2018YFB0803403), the Postgraduate Research & Practice Innovation Program of Jiangsu Province (Grant no. KYCX19_0908), and the Key Project on Anhui Provincial Natural Science Study by Colleges and Universities (Grant nos. KJ2019A0579 and KJ2019A0554).

References

[1] M. S. Yousefpoor and H. Barati, "Dynamic key management algorithms in wireless sensor networks: a survey," *Computer Communications*, vol. 134, pp. 52–69, 2019.

[2] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1233–1248, 2020.

[3] X. Ma, H. Gao, H. Xu, and M. Bian, "An IoT-based task scheduling optimization scheme considering the deadline and cost-aware scientific workflow for cloud computing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2019, no. 1, 2019.

[4] H. Gao, W. Huang, and Y. Duan, "The cloud-edge based dynamic reconfiguration to service workflow for mobile ecommerce environments: a QoS prediction perspective," *ACM Transactions on Internet Technology*, 2020.

[5] C. Jiang, T. Fan, H. Gao et al., "Energy aware edge computing: a survey," *Computer Communications*, vol. 151, pp. 556–580, 2020.

[6] X. Sun and N. Ansari, "EdgeIoT: mobile edge computing for the internet of things," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 22–29, 2016.

[7] H. Gao, Y. Xu, Y. Yin, W. Zhang, R. Li, and X. Wang, "Context-aware QoS prediction with neural collaborative filtering for internet-of-things services," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4532–4542, 2020.

[8] Y. Luo, Y. Liu, J. Liu et al., "ECM-IBS: a Chebyshev map-based broadcast authentication for wireless sensor networks," *International Journal of Bifurcation and Chaos*, vol. 29, no. 9, Article ID 1950118, 2019.

[9] P. Soni, A. K. Pal, and S. H. Islam, "An improved three-factor authentication scheme for patient monitoring using WSN in remote health-care system," *Computer Methods and Programs in Biomedicine*, vol. 182, p. 105054, 2019.

[10] Y. Harbi, Z. Aliouat, A. Refoufi, S. Harous, and A. Bentaleb, "Enhanced authentication and key management scheme for securing data transmission in the internet of things," *Ad Hoc Networks*, vol. 94, Article ID 101948, 2019.

[11] H. Gao, C. Liu, Y. Li, and X. Yang, "V2VR: reliable hybrid-network-oriented V2V data transmission and routing considering RSUs and connectivity probability," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2020.

[12] M. A. Ferrag, L. A. Maglaras, H. Janicke, J. Jiang, and L. Shu, "Authentication protocols for internet of things: a comprehensive survey," *Security and Communication Networks*, vol. 2017, Article ID 6562953, 41 pages, 2017.

[13] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "SPINS: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[14] D. Liu and P. Ning, "Multilevel μTESLA," *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 4, pp. 800–836, 2004.

[15] W.-H. Chen and Y.-J. Chen, "A bootstrapping scheme for inter-sensor authentication within sensor networks," *IEEE Communications Letters*, vol. 9, no. 10, pp. 945–947, 2005.

[16] D. Liu, P. Ning, S. Zhu, and S. Jajodia, "Practical broadcast authentication in sensor networks," in *Proceedings of the Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, pp. 118–129, San Diego, CA, USA, July 2005.

[17] K. Ren, S. Yu, W. Lou, and Y. Zhang, "Multi-user broadcast authentication in wireless sensor networks," in *Proceedings of the 2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, vol. 58, no. 8, pp. 4554–4564, San Diego, CA, USA, June 2009.

[18] X. Li, N. Ruan, F. Wu, J. Li, and M. Li, "Efficient and enhanced broadcast authentication protocols based on multilevel μTESLA," in *Proceedings of the 2014 IEEE 33rd International Performance Computing and Communications Conference (IPCCC)*, pp. 1–8, Austin, TX, USA, December 2014.

[19] Y. Gao, P. Zeng, and K.-K. R. Choo, "Multi-sender broadcast authentication in wireless sensor networks," in *Proceedings of the 2014 Tenth International Conference on Computational Intelligence and Security*, pp. 633–637, Kunming, China, November 2014.

[20] B. Groza, "Broadcast authentication with practically unbounded one-way chains," *Journal of Software*, vol. 3, no. 3, pp. 11–20, 2008.

[21] J. Håstad and M. Näslund, "Practical construction and analysis of pseudo-randomness primitives," in *Proceedings of the Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Gold Coast, Australia, 2001.

[22] T. Kwon and J. Hong, "Secure and efficient broadcast authentication in wireless sensor networks," *IEEE Transactions on Computers*, vol. 59, no. 8, pp. 1120–1133, 2010.

[23] D. Kogan, N. Manohar, and D. Boneh, "T/Key: second factor authentication from secure hash chains," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security—CCS '17*, pp. 983–999, Dallas, TX, USA, 2017.

[24] K. Grover and A. Lim, "A survey of broadcast authentication schemes for wireless networks," *Ad Hoc Networks*, vol. 24, pp. 288–316, 2015.

[25] C.-S. Park, "One-time password based on hash chain without shared secret and re-registration," *Computers & Security*, vol. 75, pp. 138–146, 2018.

[26] K. Bicakci and N. Baykal, "Infinite length hash chains and their applications," in *Proceedings. Eleventh IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 57–61, Pittsburgh, PA, USA, June 2002.

[27] R. Di Pietro, L. V. Mancini, A. Durante, and V. Patil, "Addressing the shortcomings of one-way chains," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security—ASIACCS '06*, p. 289, Taipei, Taiwan, 2006.

[28] V. Goyal, "How to re-initialize a hash chain," *IACR Cryptology ePrint Archive*, vol. 2004, p. 97, 2004.

[29] H. Zhang and Y. Zhu, "Self-updating hash chains and their implementations," in *Proceedings of the International Conference on Web Information Systems*, pp. 387–397, Springer, 2006.

[30] H. Zhang, X. Li, and R. Ren, "A novel self-renewal hash chain and its implementation," in *Proceedings of the 2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, vol. 2, pp. 144–149, Shanghai, China, December 2008.

[31] F. Xu, X. Lv, Q. Zhou, and X. Liu, "Self-updating one-time password mutual authentication protocol for ad hoc network," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 5, pp. 1817–1827, 2014.

[32] A. Al Dhaheri, C. Y. Yeun, and E. Damiani, "New two-level μTESLA protocol for IoT environments," in *Proceedings of the 2019 IEEE World Congress on Services (SERVICES)*, pp. 84–91, Milan, Italy, July 2019.

[33] Y. Liu, J. Li, and M. Guo, "Long duration broadcast authentication for wireless sensor networks," in *Proceedings of the 2012 IEEE 75th Vehicular Technology Conference (VTC Spring)*, pp. 1–5, Yokohama, Japan, May 2012.

[34] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede, "LiBrA-CAN," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 3, pp. 1–28, 2017.

[35] K.-A. Shim, Y.-R. Lee, and C.-M. Park, "EIBAS: an efficient identity-based broadcast authentication scheme in wireless sensor networks," *Ad Hoc Networks*, vol. 11, no. 1, pp. 182–189, Jan. 2013.

[36] J.-H. Son, H. Luo, and S.-W. Seo, "Denial of service attack-resistant flooding authentication in wireless sensor networks," *Computer Communications*, vol. 33, no. 13, pp. 1531–1542, 2010.

[37] E. Ayday and F. Fekri, "A secure broadcasting scheme to provide availability, reliability and authentication for wireless sensor networks," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1278–1290, 2012.

[38] D. Kim and S. An, "Source authentication schemes for reprogramming with variable packet size in wireless sensor networks," in *Proceedings of the 2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 148–150, Seattle, WA, USA, June 2015.

[39] S. Bao, W. Hathal, H. Cruickshank, Z. Sun, P. Asuquo, and A. Lei, "A lightweight authentication and privacy-preserving scheme for VANETs using TESLA and Bloom Filters," *ICT Express*, vol. 4, no. 4, pp. 221–227, 2018.

[40] L. Harn and C. Lin, "Strong (n, t, n) Verifiable secret sharing scheme," *Information Sciences*, vol. 180, no. 16, pp. 3059–3064, 2010.

[41] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Maui, HI, USA, January 2003.

[42] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.