MACHINE TO MACHINE AUTHENTICATED KEY AGREEMENT WITH FORWARD SECRECY FOR INTERNET OF THINGS

Batamu Anderson Chiphiko[1] and Hyunsung Kim[1,2]

[1]Dept. of Mathematical Sciences, University of Malawi, Malawi
[2](Corresponding Author) Dept. of Cyber Security, Kyungil University, Korea

ABSTRACT

Internet of things (IoT), is the interconnection via the Internet of computing devices embedded in everyday objects, enabling them to send and receive data. The communication is through the internet hence susceptible to security and privacy attacks. Consequently, authenticated key agreement (AKA) of communicating entities in IoT is of paramount importance as a security and privacy credential. However, IoT devices have resource-constrained feature, hence implementation of heavy security and privacy features becomes a challenge. Research on AKA in IoT has been done since year 2006. Current research trends on AKA are together with forward secrecy (FS) feasibility, which ensures that future SKs remain safe even if the long-term master keys get compromised. However, most of researches use public key cryptosystems to achieve FS, which requires heavy computations that is not good for the resource-constrained IoT environment. The main purpose of this Thesis is to devise a new machine AKA with FS for IoT, denoted as $M2M_{AKA}$-FS. To design $M2M_{AKA}$-FS, we devise a new lightweight FS framework first, which does not rely on the public key cryptosystem but based on a hash chain. The security and privacy building blocks of $M2M_{AKA}$-FS and the FS framework are symmetric key cryptosystem, one-way hash function, fuzzy commitment and challenge-response mechanism. Results of formal security and privacy analysis show that $M2M_{AKA}$-FS provides mutual authentication, SK agreement with FS, anonymity and unlinkability and is resilient against various active attacks. Performance analysis shows that $M2M_{AKA}$-FS achieves the lightweight requirements for IoT environments compared to the related protocols.

KEYWORDS

Internet of things, Forward secrecy, Lightweight environment, Hash chain, Authenticated key agreement, Security, Privacy.

1. INTRODUCTION

The Internet of things (IoT) refers to the billions of physical devices around the world that are connected to the internet. Today, there are rapid developments of IoT technologies and it has led to enormous IoT applications, such as smart city, smart health, smart home, smart energy, smart industry, smart agriculture, etc. [1-7]. In all these applications, IoT devices collect and transmit data to different device or servers. In the accumulation and transmission of data, security and privacy must be ensured for users, resources, devices, and data. There are possibilities that IoT devices could provide a channel for attackers to penetrate residential and business networks [8]. Attackers could target connected devices to transmit harmful code or activate a malware message planted on a device, collect sensitive personal information across devices, or even extract information from the device. In March 2021, hackers broke into Verkada [9]. The attackers were able to browse live feeds of over 150,000 cameras installed in factories, hospitals, classrooms, jails, and other locations, as well as access sensitive material belonging to

Verkada software clients [10-12]. In 2016, Mirai malware attack infected approximately 2.5 million inter-connected devices [13]. Mirai malware transforms connected devices, like baby monitors and doorbells, into an army that hackers can control remotely. Following Mirai malware attack, a huge number of IoT devices were hit by Reaper, Star Wars, WireX, Satori, Hajime, and Neris botnet attacks [14-15]. These incidents revealed how dangerous unprotected IoT could be, and also fueled continuing security and privacy disputes about how IoT technologies should be utilized, how sensitive data should be retained, and how access to this data should be handled.

Providing security and privacy mechanisms for IoT is important. Authentication is the basic building block of any security services even in IoT environment. Also, to provide confidentiality, it is necessary to perform key agreement, which is a mechanism where two or more parties agree on a key to be used for cryptographic algorithms [16]. In 1976, Diffie and Hellman proposed the initial key agreement protocol that is taken as building block for most of the new key agreement protocols [17-18]. However, it is exposed to man in the middle (MITM) attack, which the attacker secretly intercepts and relays messages between two parties who believe they are communicating directly with each other. MITM attack is feasible in the Diffie and Hellman key agreement protocol because it does not provide authentication process. To solve this problem, there are many trials to combine the authentication and the key agreement, which is known as authenticated key agreement (AKA).

IoT AKA protocols must be designed with the considerations on privacy, which can be provided by adopting mechanisms of anonymity and untraceability [19]. Anonymity is the concept of decoupling or removing the connection to a particular entity from the data collected [20], whereas untraceability requires that users and/or subjects are unable to determine whether the same user and/or object caused certain specific operations in the system by removing any relation between two observed items or pieces of data [15]. In Verkada and Mirai malware attacks, an adversary was able to get users identity. Furthermore, the attacker recorded live feed of users' private activities and used against them. In these attacks, IoT devices were used to reveal user's identity as well as to trace users' whereabouts. Seliem et al. discussed the major IoT privacy issues and concerns such as identification, tracking, profiling and monitoring [21]. The lack of well-designed IoT-oriented privacy techniques will inhibit the users' adoption to any IoT technology [1].

Adopting the generalized security and privacy mechanisms could be a simple solution to IoT security and privacy issues. However, many IoT devices usually have limited memory, reduced computing power, small physical area to implement the assembly, low battery power or no battery and real-time response. Most of the IoT devices deal with the real-time application where quick and accurate response with essential security using available resources is a challenging task [23]. In these circumstances, if the conventional public key cryptographic mechanisms are applied to IoT devices, their performance may not be acceptable [24]. Conventional public key cryptography places a severe demand on computing capability, memory and battery consumption that cannot be met in typical IoT devices [25]. Symmetric key cryptography easily provides the lightweight property of IoT environment. This is so because symmetric key cryptosystems rely on XOR operations, permutations, substitutions, and some other simple operations, which are not heavy to compute [21].

To design an AKA protocol for IoT environment, we need to consider both security and privacy as well as resource constraints of IoT objects. An IoT AKA protocol must not only be lightweight but also hold the following five required properties [28-29]:

- Known key security where every protocol must create independent key

- Key-compromise impersonation resilience where compromise of the long-term key of a specific principal does not allow an adversary to establish a SK with that principal by masquerading as a different principal.

- Unknown key-share (UKS) resilience where there should not be potential that a principal is tricked into sharing a key with unintended party

- Key control where both principals specify the key together

- Forward secrecy (FS) where if the long-term key of one or more of the parties is revealed, the secrecy of future SKs should not be affected.

FS guarantees the secrecy of all the future SKs in the condition that long-term secret key is known somehow at some point [19]. In addition, a different SK for each session allows only a smaller number of messages encryption, making it more difficult for the attacker for the SK exposure attack. Moreover, even if the attacker is able to find a SK somehow, that SK is not useful in decrypting data of future sessions [30]. Some AKA protocols employing public key cryptography are designed to provide for FS [36, 38-40]. However, public key cryptography requires heavy computations, which make it too big overburden for resource-constrained IoT devices [38]. So, with the resource-constrained nature of IoT devices, designing lightweight protocols is a requirement. The challenge is how to design a lightweight AKA protocol that achieves FS.

FS is a strong notion of security for an AKA protocol. Nevertheless, in the course of designing AKA protocols that provide FS, the other security and privacy concerns should not be ignored. Similarly, the resource-constrained environment for IoT has to be considered. From the literature available, it is imperative that we design a FS framework to provide SK agreement with FS and to design a new lightweight authenticated key agreement protocol that provides FS and the other security and privacy requirements for IoT environment. Thus the contributions of this paper are:

- To propose a FS framework and an AKA protocol with FS for IoT environment.
- To propose a new AKA protocol with FS basing on FS framework but having the other authentication and key agreement requirements for IoT environments.

The rest of this paper is organized as follows; Chapter 2 provides cryptographic preliminaries. In Chapter 3, we propose a FS framework and a new AKA protocol for IoT environment. After that, we provide analysis of the proposed protocol focused on the security and performance in Chapter 4. Finally, Chapter 5 concludes the paper.

2. BACKGROUND AND PRELIMINARIES

This section discusses the basic cryptographic primitives that the proposed AKA protocol, $M2M_{AKA}$-FS relies on.

2.1. Integrity Primitive

The cryptographic concept of hash function is used to provide integrity check of the message in the FS framework and $M2M_{AKA}$-FS. Furthermore, we discuss how a chain of hash values from hash function can be used as a security building block to a FS framework. [30]. We define a hash function as follows:

Definition 2.1. A **hash function** is a function $h(.): D \to R$ such that:

1. The domain D is a set that may be finite or infinite
2. The range R is a finite set
3. The function is computationally easy to evaluate for any given input $x \in D$
4. The function has preimage resistance, second preimage resistance and collision resistance.

The intractability of a hash function is based on the three properties, preimage resistance, second preimage resistance and collision resistance [41].

Definition 2.2. A function $h(.): D \to R$ is **preimage resistant** if for a given $y \in R$ it is computationally infeasible to find an $x \in D$ such that $h(x) = y$.

Definition 2.3. A function $h(.): D \to R$ is **second preimage resistant** if for a given $x \in D$ it is computationally infeasible to find $x' \in D$ with $x' \neq x$ such that $h(x') = h(x)$.

Definition 2.4. A function $h(.): D \to R$ is **collision resistant** if it is computationally infeasible to find x and $x' \in D$ with $x' \neq x$ such that $h(x) = h(x')$.

The input to a hash function is a message of arbitrary length but the output is of a fixed length. The messages returned by hash functions are called message digests or simply hash values. The sender will send both message x and its digest y, the receiver computes his message digest $y' = h(x)$ with the x he received and verifies if it matches the digest $\boldsymbol{y}$ he received. If it matches, then the message has not been tampered. If we repeatedly apply hash function with an initial input (seed), we can generate a chain of hash values which can be used as OTS values.

Definition 2.5. (Chain of hash values) A chain of hash values is a sequence of values derived via consecutive applications of a cryptographic hash function to a seed.

A random value r_k is selected as a seed and a hash of r_k is computed, $h(r_k)$, to get a hash value d_{i1}. Then $d_{i2} = h(d_{i1})$ is computed by using d_{i1} as input to the hash function. This continues until a chain of n hash values is generated. The hash values d_{ij} are the vertices and the computations $h(d_{ij})$ are the edges. They all point in one direction such that one cannot obtain d_{in-1} given d_{in} due to the hash function properties, **definitions 2.2**, **2.3** and **2.4**. Figure 1, demonstrates how a chain of hash values can be generated.

$$\begin{tikzcd} r_k \arrow[r, "h(r_k)"] & d_{i1} \arrow[r, "h(d_{i1})"] & d_{i2} \arrow[r, "h(d_{i2})"] & d_{i3} \arrow[r, "h(d_{i3})"] & \dots \arrow[r, "h(d_{in-2})"] & d_{in-1} \arrow[r, "h(d_{in-1})"] & d_{in} \end{tikzcd}$$

Figure 1. Hash chain secret values generation

All the values in this chain, d_{ij} for $j \in \{1, 2, 3, \dots, n\}$ are then stored securely and will be used one by one from d_{in} to d_{i1} to provide for FS to SKs through one time use of the hash values, which should be opposite sequence from the generation. For the chain of hash values to provide FS, we need the following definitions:

Definition 2.7: (Negligible function) A function$\omega(n)$is negligible if for all$c > 0$, there exists $n_0 \in \mathbb{Z}^+$such that $|\omega(n)| \leq \frac{1}{n^c}$ for all $n \geq n_0$.

Definition 2.8: (Hash Chain FS Property) Given a hash function output $h(d_{ij})$ of length *n* bits, it is computationally infeasible to find d_{ij} in polynomial time.

Assumption 2.1: (FS assumption) Any probabilistic polynomial time adversary against the hash chain FS property of a hash function has a negligible advantage in finding the integer d_{ij} (input) for the given $h(d_{ij})$ (output) of size *n* bits.

$$Adv_{\mathcal{A}}^{FS}(n) = Pr\left[\mathcal{A}\left(1^n, h(d_{ij})\right)\right] = d_{ij} \leq \omega(n).$$

Definition 2.9: (FS collision resistance property) Given an input and a hash output $(d_{ij}, h(d_{ij}))$ with$h(d_{ij})$of length n bits, it is computationally infeasible to find d_{il} in polynomial time such that$h\left(d_{ij}\right) = h(d_{il})$ and$d_{ij} \neq d_{il}$.

Assumption 2.2: (FS collision resistance assumption) Any probabilistic polynomial time adversary against the FS collision resistance property of the hash function has a negligible advantage in finding the integer d_{il} for the given d_{ij} and $h(d_{ij})$ of size *n* bits.

$$Adv_{\mathcal{A}}^{CR}(n) = Pr\left[\mathcal{A}\left(1^n, d_{ij}, h(d_{ij}) = h(d_{il})\right)\right] = d_{il} \leq \omega(n).$$

The sender and receiver will both keep the last hash value they used. Sender sends a commit request, with a secret value d_{ij-1} to the receiver. Receiver with d_{ij}, uses the secret value d_{ij-1}, to verify the sender by computing $h(d_{ij-1})$. If $h(d_{ij-1})$ is equal to d_{ij}, the commitment receiver has, then the sender is legit. The receiver will now take d_{ij-1} as new commitment whereas the sender will send d_{ij-2} in the next login. This is repeated *n*-times until all the hash chain secret values are exhausted. An adversary who gets the hash value d_{ij} for the current session, will not be able to use to access another session in future. The next session he will need to compute d_{ij-1} such that $d_{ij} = h(d_{ij-1})$ which has a negligible probability of succeeding due to **definitions 2.8** and **2.9** of hash chain FS. Since the adversary cannot use d_{ij} to log in, in the future session then we can use a chain of hash values as OTS values that can provide key agreement with FS.

2.2. Confidentiality Primitive

Confidentiality is the concept of hiding or scrambling information so that only the intended recipient has access. For confidentiality, we *encrypt* a message: given a message, we pair it with a key and produce a meaningless jumble that can only be made useful again by reversing the process using the same key. In the proposed FS framework and *M*2M_{AKA}*-FS*, we have adopted the symmetric key cryptography (SKC) due to resource constrained nature of IoT environment. SKC uses a single shared secret to establish a secure channel between entities. Ciphers in this category are called symmetric because they use the same key to encrypt and to decrypt the data.

Definition 2.10: SKC is formed with a triple of algorithms $SKC = \{G(k), E_K(\cdot), D_K(\cdot)\}$ where [42]

- $G(k)$, the key generation algorithm, that takes a security parameter *k* and returns a symmetric encryption key *K*.

- $E_K(M)$, the encryption algorithm, is a deterministic algorithm that takes *K* and a message $M \in \{0,1\}^*$ to produce a cipher text *C*.

- $D_K(C)$, the decryption algorithm, is a deterministic algorithm that takes *K* and cipher text *C* to produce either a message $M \in \{0,1\}^*$ or a special symbol $\perp$ to indicate that the cipher text was invalid.

Thus, we require that for all K that can be output by $G(k)$, for all $M \in \{0,1\}^*$, and for all C that can be output by $E_K(M)$, we have that $M = D_K(C)$. We also require that $G(k)$, $E_K(M)$ and $D_K(C)$ can be computed in polynomial time.

2.3. Machine Authentication Factor

Most connections for IoT objects do not require human intervention for authentication, hence machine to machine authentication is necessary. Machine to machine authentication uses the following factors, what the machine is, i.e., biometric features, and what the machine has e.g. memory card. Machine fingerprinting is a technique to authenticate devices using unique features extracted from the machines' distinctive characteristics. A machine's radio frequency (RF) emission is used as the fingerprint for the authentication of the device [40]. Apparently each distinctive machine emits a unique RF signal which can be used to identify it. The machine's unique RF signal can be captured and preserved in a template such that each time the machine wants connection, it can be verified by comparing the signal it is emitting with the previously recorded signal. The verification procedure uses a fuzzy commitment scheme $F: (\{0,1\}^n, \{0,1\}^n) \rightarrow (\{0,1\}^l, \{0,1\}^n)$, together with error correcting codes $C \subseteq \{0,1\}^n$, which can commit a codeword $c \in C$ using a n bit witness b_i as $F(c, b_i) = (\alpha, \delta)$, where $\alpha = h(c)$ and $\delta = b_i \oplus c$. The commitment $F(c, b_i) = (\alpha, \delta)$ can be opened using witness b_i' which is relatively close to b_i, but no need to be the same as y (Li et al. 2017). To open the commitment using b_i', the receiver computes $c' = f(b_i' \oplus \delta) = f(c \oplus (b_i' \oplus b_i)$, and checks $\alpha = ?\, h(c')$. If they are equal, the commitment is opened successfully. Otherwise, the witness b_i' is not valid. Due to the noisy characteristic, i.e., the input biometric information is not the same as the template exactly, it can be used in fuzzy commitment scheme. The biometric template can be seen as the witness b_i and c can be opened by the input biometric b_i', which is close to b_i[20, 38,40]. Here, $F(.)$ is the commitment scheme while $f(.)$ is a decoding function of the commitment scheme hence $F(.)$ is used to register unique RF signals whereas $f(.)$ is used to extract the distinctive features for comparison in the presence of the witness.

2.4. CK Threat Model

The Canetti & Krawczyk's adversary model (CK-adversary model) can be used to evaluate the security of AKA protocols in IoT. Based on the CK adversary model, $\mathcal{A}$ is supposed to have the following capacities:

- $\mathcal{A}$ has full control of the communication channel between the communicating parties, such as intercepting, eavesdropping, inserting, modifying, and deleting any transmitted messages over the public channel.
- To characterize FS, $\mathcal{A}$ may also be allowed to corrupt valid parties to attain a long-term secret key.
- $\mathcal{A}$ can extract the secret parameters stored in IoT device memory chip using side-channel attacks when the IoT device is stolen or obtained by $\mathcal{A}$.
- $\mathcal{A}$ may be a legitimate but malicious user.
- $\mathcal{A}$ can launch multiple types of known attacks, such as user impersonation attack, replay attack and known session-specific temporary information attack.

Given such capabilities to $\mathcal{A}$, the design of an AKA protocol should be such, that will still provide session key security when attacked by $\mathcal{A}$. Under the CK model a session key is security is defined as follows:

Definition 2.11: (Session-key security). A key-agreement protocol π is called Session Key secure (or SK-secure) if the following properties hold for $\mathcal{A}$:

C1: If two uncorrupted parties complete matching sessions, then they both output the same SK.

C2: The probability that $\mathcal{A}$ can distinguish the session key from a random value is no more than 1/2 plus a negligible fraction in the security parameter.

3. Machine to Machine Authenticated Key Agreement with Forward Secrecy for IoT

This chapter proposes a FS framework and a new machine to machine authenticated key agreement protocol, *M2M*$_{AKA}$*-FS*, with FS in IoT environments. Firstly, we design a FS framework that can be adopted in any protocol to provide SK agreement, which uses a chain of hash values as OTS values. Then, we design *M2M*$_{AKA}$*-FS* as an AKA protocol based on the FS framework for IoT environments. *M2M*$_{AKA}$*-FS* emphasizes on SK agreement with FS in IoT environments that supports lightweight, machine to machine authentication, privacy-preserving and efficiency in performance. The security and privacy of the FS framework and *M2M*$_{AKA}$*-FS* is based on symmetric key cryptography, hash function and fuzzy commitment.

3.1. Network Model

Figure 2 shows the network environment and its description is as follows:

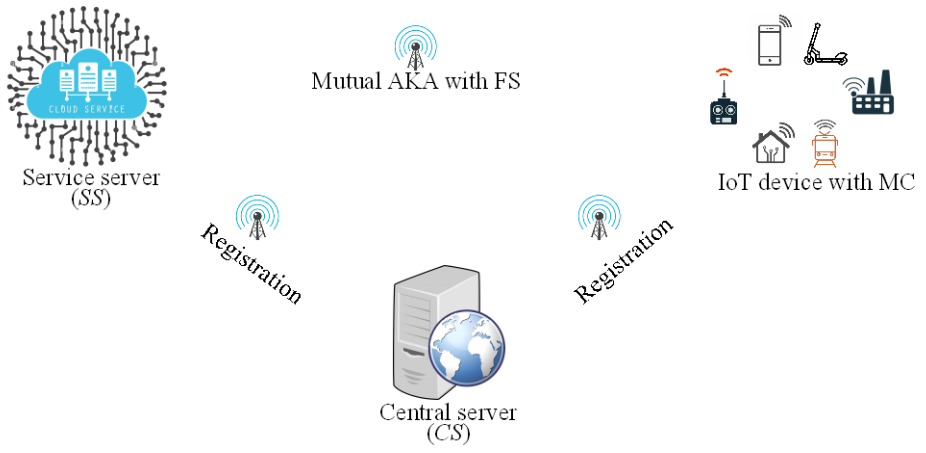


Figure 2. Network model

IoT devices with *MC*: These are pieces of hardware, such as sensors, actuators, gadgets, appliance, or machines that are programmed for certain applications and can transmit data over the internet or other networks. They consist of software and hardware for generating sensing data, computing meta information, sending reports, receiving instructions, and acting accordingly. They are usually resource constrained due to their size hence they often have small battery power, processing capacity and storage space. Their main role is to collect data and send the data to *SS* through *CS* or directly to *SS* so that *SS* can take the necessary actions in real time. They are installed with some sensors that collect environment data required for the target services and a *MC* is installed for secure data storage.

Central server (*CS*): A fully trusted server that is responsible for registration and login of IoT devices and *SS*. It consists of software and hardware for identification and credentials, it stores unique identification and secret credentials such as keys. It facilitates communication and data exchanges between IoT device and *SS* and responsible for authentication of the entities.

Service server (*SS*): It consists of software and hardware for receiving instructions, and acting accordingly. This supports various rich and convenient services to IoT device. Responsible for acting basing on data the IoT device collect and submit. Makes decisions basing on the data and can instruct IoT device accordingly. Nevertheless, it does not have authority and credential to directly communicate with IoT device, but rather through the *CS*.

3.2. Design Goals

To design an AKA protocol, we need to consider three aspects at the same time, which are the network environment, security goals and privacy goals. This subsection will consider them one by one. IoT offers numerous advantages and services to the users. An important aspect of pervasive IoT devices is its constrained resources. So, various energy efficient lightweight mechanisms should be designed to store, process and transfer the data as per application requirements and with an optimized resource management.

AKA-Goal 1: Lightweight property: Given the constraints of IoT environment, it is desirable that the AKA protocol should be lightweight in computation and communication. AKA protocol primitives should consume fewer resources without compromising the required level of security and privacy.

AKA-Goal 2: Mutual authentication: An unidirectional authentication is common for various network environments. However, an AKA protocol for IoT requires to provide much high-level security than the others due to their complicated applications for the real-life.

AKA-Goal 3: SK agreement with FS: An important security feature of the SK agreement is to preserve security of future communications even if the long term secret key of the system is exposed to $\mathcal{A}$. FS ensures that, if the system is breached, $\mathcal{A}$ should not get further access to future communications.

AKA-Goal 4: Resilience to various attacks: An AKA protocol should support all security goals and must resist against various security attacks, both passive and active. Such attacks may include:

- Eavesdropping attack: It is a passive attack which aims to achieve data or scan open ports and vulnerabilities of the network [3, 16].

- Replay attack: It is an active attack in which $\mathcal{A}$ interferes with a protocol run by insertion of some messages from previous protocol runs or parallel sessions. It can be considered as a combination of eavesdropping and modification attacks. A protocol is vulnerable to the replay attack if it fails to provide freshness of the message [21, 44].

- Impersonation attack: It is an active attack which aims to defeat the authenticity. In this attack, $\mathcal{A}$ tries to impersonate one or more legal entities [4, 12, 45].

- MITM attack: It is a variant of the impersonation attack, where $\mathcal{A}$ resides between two entities, and convincingly impersonates both victims. MITM is feasible when a protocol lacks authentication [3, 46].

- Unknown key share (UKS) attack: In a UKS attack, two entities share a SK, but they have different views about the identity of their peer. The UKS attack is feasible when a key exchange protocol fails to provide an authentication binding between the SK and identifiers of the legal entities [3, 12, 16, 46].

AKA-Goal 5: Anonymity: The property that provides device identity protection of IoT devices. $\mathcal{A}$ who has recorded past messages should not be able to discover the identities of IoT devices. $\mathcal{A}$ should not be able to tell what messages belong to what entity. We develop mathematical techniques that enable anonymity in $M2M_{AKA}$-FS.

AKA-Goal 6: Unlinkability: $\mathcal{A}$ cannot distinguish whether two sessions are executed by the same IoT device or not.

3.3. FS Framework

A FS framework is designed to provide a platform for any two entities to agree on a fresh SK with provision of FS. It is based on a chain of hash values and the challenge-response mechanism, which is mentioned in **definition 2.5**. It uses a synchronized credential for each session between entities with session dependent random values, which satisfies **definitions 2.2** and **2.3** properties. Note that the purpose of the FS framework is to achieve FS when any security protocol requires to agree on a SK that could be used as a basic security building block of $M2M_{AKA}$-FS. It has two phases, credential setup and SK agreement. One is to set up credentials as OTS values between entities, which is based on a hash chain. The other is to agree on a SK based on the synchronized credential established in the credential setup. Random numbers are used for both phases to derive different secret values and to provide freshness of messages.

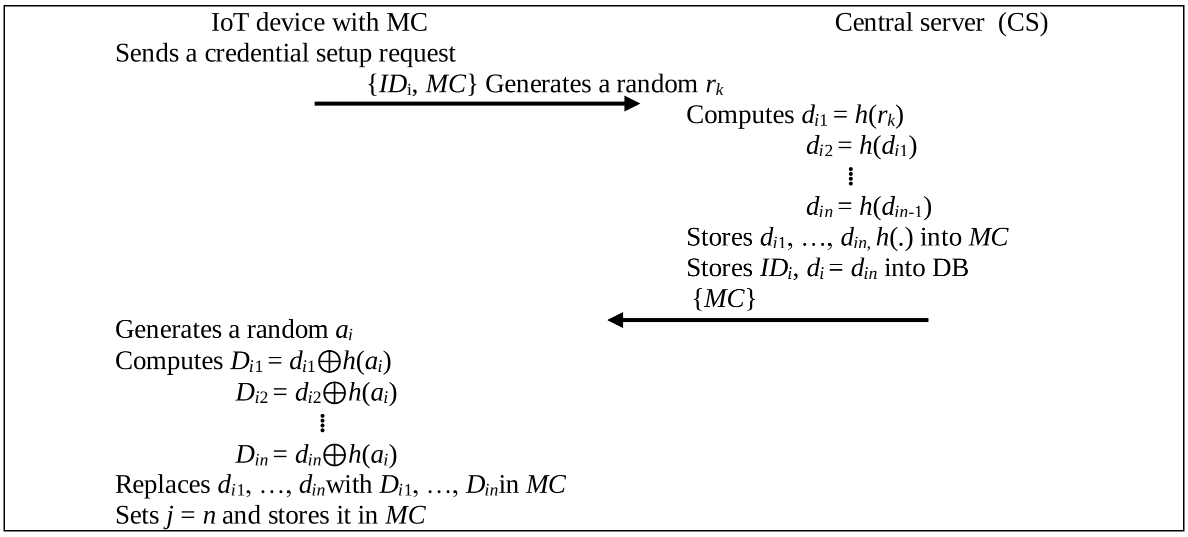
3.1.1. Credential Setup Phase

The aim of this phase is to set up the fresh security credentials for FS, which entities could use to synchronize with each other. These security credentials are generated as a chain of hash values as per **definitions 2.5** and **2.6** and demonstrated in Figure 1. Figure 3 demonstrates the setup phase and its description is as follows:

C1. For a credential setup with CS, IoT device sends a request message of its ID_i and an MC to CS.

C2. Upon receiving the request, CS generates a random number r_k, uses it as a seed of a hash chain to compute n hash values as $d_{i1} = h(r_k)$, $d_{i2} = h(d_{i1})$, ..., $d_{in} = h(d_{in-1})$. Each value d_{ik}, $1 \le k \le n$, in the hash chain is stored in MC together with the hash function $h(\cdot)$. CS keeps both of ID_i and $d_i = d_{in}$ in its database as IoT's starting credential and sends back MC to IoT device.

C3. Upon receiving MC, IoT device generates a random number a_i and makes MC to compute $D_{i1} = d_{i1} \oplus h(a_i)$, $D_{i2} = d_{i2} \oplus h(a_i)$, …, $D_{in} = d_{in} \oplus h(a_i)$, replace d_{i1}, …, d_{in} with D_{i1}, …, D_{in} in MC and keep a_i in secret. MC sets and stores $j = n$.



IoT device with MC — Central server (CS)

Sends a credential setup request

$\{ID_i, MC\}$ $\longrightarrow$

Generates a random r_k

Computes $d_{i1} = h(r_k)$

$d_{i2} = h(d_{i1})$

$\vdots$

$d_{in} = h(d_{in-1})$

Stores $d_{i1}, \ldots, d_{in}, h(.)$ into MC

Stores ID_i, $d_i = d_{in}$ into DB

$\{MC\}$ $\longleftarrow$

Generates a random a_i

Computes $D_{i1} = d_{i1} \oplus h(a_i)$

$D_{i2} = d_{i2} \oplus h(a_i)$

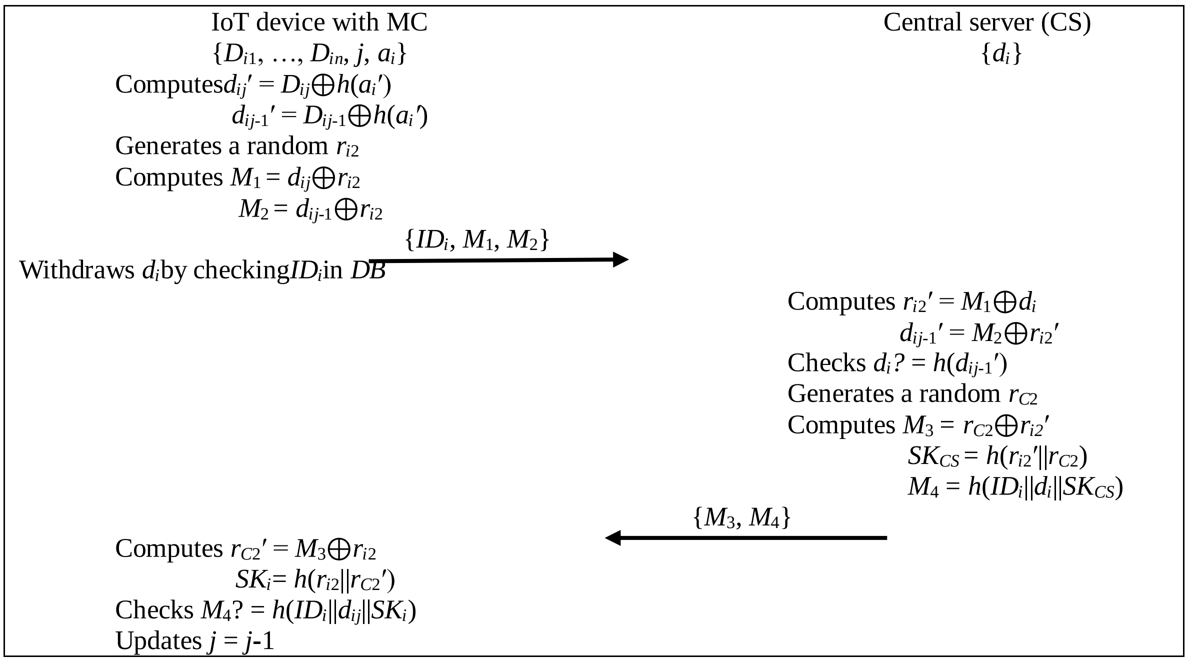
$\vdots$

$D_{in} = d_{in} \oplus h(a_i)$

Replaces $d_{i1}, \ldots, d_{in}$ with $D_{i1}, \ldots, D_{in}$ in MC

Sets $j = n$ and stores it in MC

Figure 3. Credential setup phase of FS framework



IoT device with MC — Central server (CS)

$\{D_{i1}, \ldots, D_{in}, j, a_i\}$ — $\{d_i\}$

Computes $d_{ij}' = D_{ij} \oplus h(a_i')$

$d_{ij-1}' = D_{ij-1} \oplus h(a_i')$

Generates a random r_{i2}

Computes $M_1 = d_{ij} \oplus r_{i2}$

$M_2 = d_{ij-1} \oplus r_{i2}$

$\{ID_i, M_1, M_2\}$ $\longrightarrow$

Withdraws d_i by checking ID_i in DB

Computes $r_{i2}' = M_1 \oplus d_i$

$d_{ij-1}' = M_2 \oplus r_{i2}'$

Checks $d_i ? = h(d_{ij-1}')$

Generates a random r_{C2}

Computes $M_3 = r_{C2} \oplus r_{i2}'$

$SK_{CS} = h(r_{i2}' || r_{C2})$

$M_4 = h(ID_i || d_i || SK_{CS})$

$\{M_3, M_4\}$ $\longleftarrow$

Computes $r_{C2}' = M_3 \oplus r_{i2}$

$SK_i = h(r_{i2} || r_{C2}')$

Checks $M_4 ? = h(ID_i || d_{ij} || SK_i)$

Updates $j = j-1$

Figure 4. SK agreement phase of FS framework

3.3.2. SK Agreement Phase

The aim of this phase is for the two entities that setup the FS framework credentials to agree on a SK between them basing on the settled credentials. This phase focuses on the basic notion design on how to agree on a SK with the provision of FS. To agree on a SK, each entity needs to check authenticity of the counterpart. The session dependent synchronized credential is used for that purpose. IoT device sends to *CS* the security credential it committed. *CS* authenticates IoT device by computing hash function of the received security credential and verifies if it matches the commitment *CS* has. If it matches, then the security credential received came from a legit sender. After a successful verification, both IoT device and CS update their commitment for the next session. This ensures that each value of the hash chain is used for authentication only once.

Figure 4 demonstrates the FS framework SK agreement phase.

3.4. M2M$_{AKA}$-FS

M2M$_{AKA}$-FS is designed to provide a machine to machine authenticated key agreement with FS in IoT environments. The security of *M2M$_{AKA}$-FS* is based on symmetric key cryptography and hash function to suit the lightweight environment of IoT. A fuzzy commitment scheme is adopted to verify validity of IoT devices basing on machine fingerprinting as authentication factor. Dynamic identity of IoT device has been used to provide for anonymity whereas random numbers provide unlinkability of IoT devices as well as for freshness of the SK. *M2M$_{AKA}$-FS* has three parties, which are IoT device with *MC*, *CS* and *SS*. IoT device checks ownership of *MC* before sending login request message to *CS*. *M2M$_{AKA}$-FS* has four phases which are: Initialization phase, IoT device registration phase, *SS* registration phase, and login and AKA phase.

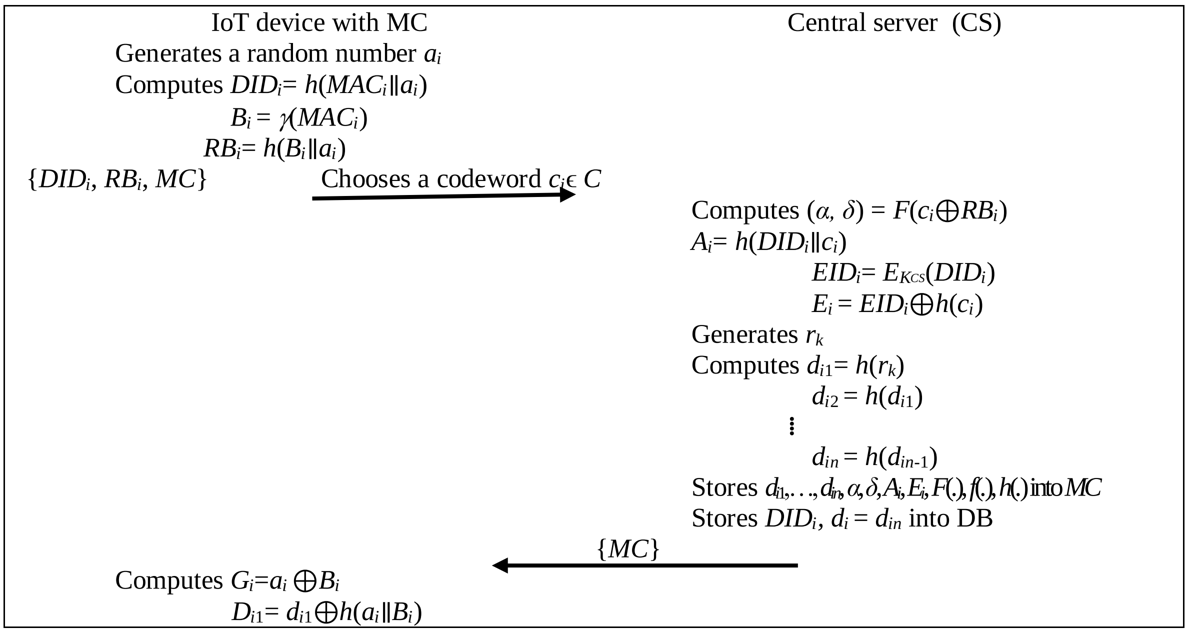
3.4.1. Initialization Phase

In this phase *CS* defines the following required system parameters that are necessary for the execution of *M2M$_{AKA}$-FS* as follows:

Step 1 : First, a group Z_p is selected and a code set $C \in \{0, 1\}^n$.
Step 2 : *CS* picks a long term private key $K_{CS} \in Z_p$ and keeps it secret.
Step 3 : *CS* selects a collision-resistant one-way cryptographic hash function $h(.)$.
Step 4 : *CS* defines two fuzzy commitment functions $f(.)$ and $F(.)$.
Step 5 : *CS* selects two asymmetric key functions $E(.)$ for encryption and $D(.)$ for decryption based on AES.
Step 6 : *CS* publishes the parameters $\{Z_p, h(.), f(.), F(.), E(.), D(.)\}$ to the targeted network.

3.4.2. IoT Device Registration Phase

Before IoT device communicates with *SS*, it needs to be registered to *CS* based on the FS framework so that it becomes a part of the system. Figure 5 shows the flow of this phase and the description of it is as follows:


IoT device with MC
Generates a random number a_i
Computes $DID_i= h(MAC_i \| a_i)$
$B_i = \gamma(MAC_i)$
$RB_i= h(B_i \| a_i)$
Chooses a codeword $c_i \in C$
$\{DID_i, RB_i, MC\}$ $\longrightarrow$

Central server (CS)
Computes $(\alpha, \delta) = F(c_i \oplus RB_i)$
$A_i= h(DID_i \| c_i)$
$EID_i= E_{K_{CS}}(DID_i)$
$E_i = EID_i \oplus h(c_i)$
Generates r_k
Computes $d_{i1}= h(r_k)$
$d_{i2} = h(d_{i1})$
⋮
$d_{in} = h(d_{in-1})$
Stores $d_{i1},\ldots,d_{in}, \alpha, \delta, A_i, E_i, F(.), f(.), h(.)$ into *MC*
Stores DID_i, $d_i = d_{in}$ into DB
$\longleftarrow$ $\{MC\}$

Computes $G_i=a_i \oplus B_i$
$D_{i1}= d_{i1} \oplus h(a_i \| B_i)$

$D_{i2}= d_{i2}\oplus h(a_i\|B_i)$

⋮

$D_{in} = d_{in}\oplus h(a_i\|B_i)$

Stores G_i in MC

Replaces $d_{i1}, \ldots, d_{in}$ with $D_{i1}, \ldots, D_{in}$

Sets $j = n$ and stores it in MC

Figure 5. IoT device registration phase of M2M$_{\text{AKA}}$-FS

R1. IoT device generates a random number a_i and computes an amplified dynamic identity $DID_i= h(MAC_i\|a_i)$. After that, it derives its machine fingerprint $B_i = \gamma(MAC_i)$, secures the fingerprint by computing $RB_i= h(B_i\|a_i)$ and sends the registration request message $\{DID_i, RB_i\}$ together with MC to CS via a secure channel.

R2. Upon receipt of the registration request from IoT device, CS chooses a random code word $c_i \in C$ for IoT device and computes $(\alpha, \delta) = F(c_i, RB_i)$, where $\alpha = h(c_i)$ and $\delta = c_i\oplus RB_i$, $A_i = h(DID_i\|c_i)$, $EID_i= E_{KCS}(DID_i)$ and $E_i= EID_i\oplus h(c_i)$ where $E(.)$ is the symmetric encryption function as per **definition 2.10** of SKC, and K_{CS} is the master secret key of CS. CS generates a random number r_k and computes a chain of hash values with the seed r_k as $d_{i1} = h(r_k)$, $d_{i2} = h(d_{i1})$, ...,and $d_{in} = h(d_{in\text{-}1})$. CS stores $\{d_{i1}, \ldots,d_{in}, \alpha, \delta, A_i, E_i, \gamma(.), f(.), h(.)\}$ into an MC, sets $d_i = d_{in}$ and stores DID_i and d_i in its database. CS sends MC to IoT device via a secure channel.

R3. After MC installation, IoT device computes $G_i= a_i\oplus B_i$ and secures the hash chain values by computing $D_{i1} = d_{i1}\oplus h(a_i\|B_i)$, $D_{i2}=d_{i2}\oplus h(a_i\|B_i)$, …, $D_{in} = d_{in}\oplus h(a_i\|B_i)$. IoT device stores $G_i, D_{i1}, \ldots, D_{in}$ into MC and deletes $d_{i1}, \ldots, d_{in}$ from MC and sets counter j to n and stores it in MC. Now MC contains the parameters $\{D_{i1}, \ldots, D_{in}, \alpha, \delta, A_i, E_i, G_i,\gamma(.), f(.), h(.), j\}$.

3.4.3. SS Registration Phase

Like IoT device, SS also needs to be registered with CS based on the FS framework before providing any service to IoT device. SS selects an identifier ID_{SS} and sends it to CS via a secured channel, which is established based on the pre-relationship with CS. After receiving the registration request from SS, CS computes a secret key $K_{CS\text{-}SS} = h(ID_{SS}\|K_{CS})$ for SS to be used for encryption and decryption as per

definition 2.10 of SKC. CS generates a random number r_p for FS framework credential setup. With seed r_p, CS computes a chain of hash secret values $d_{s1} = h(r_p)$, $d_{s2} = h(d_{s1})$, …, $d_{sm}= h(d_{sm\text{-}1})$ and sends $\{ID_{SS}, K_{CS\text{-}SS}, d_{s1}, d_{s2}, \ldots, d_{sm}\}$ to SS securely and CS keeps ID_{SS} and $d_s = d_{sm}$ in its DB. SS sets $l = m$ and stores $\{ID_{SS}, K_{CS\text{-}SS}, d_{s1}, d_{s2}, \ldots, d_{sm}, l\}$ in its DB.

3.4.4. Login and AKA Phase

When and IoT device would like to get services from SS, it firstly checks the ownership of MC and requests to login and agree on SK with SS through CS. CS checks authenticity of IoT device as well as of SS. This is achieved through verification of credentials basing on both the confidentiality and integrity primitives as defined in **definitions 2.10** and **2.1**. SK is agreed when the two have been mutually authenticated. Finally, all the parties update their security credentials to ensure that the next login is unique from others. Figure 6 shows the detailed conceptual flow of this phase and its description as follows:

A1. IoT device imprints its machine fingerprint $B_i'=\gamma(MAC_i)$ and MC derives $a_i'=G_i\oplus B_i'$, computes $RB_i'=h(B_i'\Vert a_i')$ and $c_i'=f(\delta\oplus RB_i')$, and validates whether $h(c_i')$ is equal to α. MC terminates the sessionif the validation is failed. Otherwise, IoT device passes the fingerprint verification and MC computes $E_i'=h(MAC_i\Vert a_i')$ and $A_i'=h(DID_i'\Vert c_i')$, and validates if A_i' is equal to A_i. MC terminates the sessionif it fails. Otherwise, MC generates random numbers r_{i1} and $r_{i2}$and computes $EID_i'=E_i\oplus h(c_i')$, $M_1=h(DID_i'\Vert EID_i')\oplus r_{i1}$ and $M_2 = h(EID_i'\Vert r_{i1})\oplus ID_{SS}$. MC retrieves the hash chain secret values by computing $d_{ij}'=D_{ij}\oplus h(a_i'\Vert B_i')$ and $d_{ij-1}'=D_{ij-1}\oplus h(a_i'\Vert B_i')$. IoT device computes $M_3=d_{ij}\oplus r_{i2}$,$M_4=d_{ij-1}\oplus r_{i2}$ and $M_5=h(DID_i\Vert ID_{SS}\Vert d_{ij-1}\Vert r_{i1}\Vert r_{i2})$ and sends the login and key agreement request message $\{EID_i', M_1, M_2, M_3, M_4, M_5\}$ to CS.

A2. On receiving the login and key agreement request, CS computes $DID_i''= D_{KCS}(EID_i')$,$r_{i1}'=M_1\oplus h(DID_i''\Vert EID_i')$ and $ID_{SS}'=M_2\oplus h(EID_i'\Vert r_{i1}')$, $r_{i2}'=M_3\oplus d_i$ and $d_{ij-1}'=M_4\oplus r_{i2}'$ and validates $d_i?=h(d_{ij-1}')$. If it does not hold, the session is terminated. Otherwise, CS validates M_5 ?= $h(DID_i''\Vert ID_{SS}'\Vert d_{ij-1}'\Vert r_{i1}'\Vert r_{i2}')$. If not, the session is terminated. Otherwise, CS generates a random number $r_{CS}\in Z_n^*$ and computes $K_{CS\text{-}SS}'=h(ID_{SS}'\Vert K_{CS})$, $M_6=DID_i''\oplus K_{CS\text{-}SS}'$ and $M_7=h(DID_i''\Vert K_{CS\text{-}SS}')\oplus r_{CS}$. CS then uses the hash chain value of SS to secure r_{i2} by computing $M_8=r_{i2}'\oplus d_S$. CS computes $M_9=h(DID_i''\Vert K_{CS\text{-}SS}'\Vert d_S\Vert r_{i2}'\Vert r_{CS})$. After that, CS sends a message $\{M_6, M_7, M_8, M_9, M_{10}\}$ to SS.

A3. On receiving the message from CS, SS retrieves $DID_i'''=M_6\oplus K_{CS\text{-}SS}$, $r_{CS}'=M_7\oplus h(DID_i'''\Vert K_{CS\text{-}SS})$ and $r_{i2}''=M_8\oplus d_{Sl}$, and validates $M_9?=h(DID_i''\Vert K_{CS\text{-}SS}'\Vert d_{Sl}\Vert r_{i2}''\Vert r_{CS}')$. SS terminates the session if the validation fails. Otherwise, SS generates two random numbers r_{S1} and r_{S2} and computes $M_{10}=r_{S1}\oplus K_{CS\text{-}SS}$ and $SK_{SS} = h(DID_i'''\Vert ID_{SS}\Vert r_{i2}''\Vert r_{CS}'\Vert r_{S2})$. Then SS computes $M_{11}=d_{Sl}\oplus r_{S2}$, $M_{12}=d_{Sl\text{-}1}\oplus r_{S2}$ and $M_{13}=h(K_{CS\text{-}SS}\Vert SK_{SS}\Vert r_{S1})$. SS returns to CS with a message $\{M_{10}, M_{11}, M_{12}, M_{13}\}$. If SS does not reject failure message in ΔTtime, SS updates its commitment by updating $l=l$-1.

A4. After getting the message from SS, CS retrieves $r_{S1}'=M_{10}\oplus K_{CS\text{-}SS}'$, $r_{S2}'=M_{11}\oplus d_{Sl}$ and $d_{Sl\text{-}1}'=M_{12}\oplus r_{S2}'$. CS validates $d_S? =h(d_{Sl\text{-}1}')$. If not, the session is terminated and a reject message is sent to SS. Otherwise, C Scomputes $SK_{CS} = h(DID_i''\Vert ID_{SS}'\Vert r_{i2}'\Vert r_{CS}\Vert r_{S2}')$ and validates M_{13} ?= $h(K_{CS\text{-}SS}'\Vert SK_{CS}\Vert r_{S1}')$. CS computes $M_{14}=r_{i1}'\oplus r_{CS}$, $M_{15}=r_{CS}'\oplus r_{S2}'$, $M_{16}=E_{KCS}(DID_i''\oplus r_{CS})\oplus r_{i1}'$ and $M_{17}=h(DID_i''\Vert SK_{CS}\Vert r_{CS}\Vert r_{S2}'\Vert M_{16})$. Finally, CS

IoT device with MC	**Central server (CS)**	**Service server (SS)**
Derives $B_i' = \gamma(MAC_i)$		
Computes $a_i' = G_i\oplus B_i'$		
$RB_i' = h(B_i'\Vert a_i')$		
$c_i' = f(\delta\oplus RB_i')$		
Checks $h(c_i')$?= α		
Computes $E_i' = h(MAC_i\Vert a_i')$		
$A_i' = h(DID_i'\Vert c_i')$		
Checks A_i' ?= A_i		
Generates r_{i1},$r_{i2}\in Z_n^*$		
Computes $EID_i' = E_i'\oplus h(c_i')$		
$M_1 = h(DID_i'\Vert EID_i')\oplus r_{i1}$		
$M_2 = h(EID_i'\Vert r_{i1})\oplus ID_{SS}$	Computes $DID_i'' = D_{KCS}(EID_i')$	
$d_{ij}' = D_{ij}\oplus h(a_i'\Vert B_i')$	$r_{i1}' = M_1\oplus h(DID_i''\Vert EID_i')$	
$d_{ij-1}' = D_{ij-1}\oplus h(a_i'\Vert B_i')$	$ID_{SS}' = M_2\oplus h(EID_i'\Vert r_{i1}')$	
$M_3 = d_{ij}\oplus r_{i2}$	$r_{i2}' = M_3\oplus d_i$	
$M_4 = d_{ij-1}\oplus r_{i2}$	$d_{ij-1}' = M_4\oplus r_{i2}'$	
$M_5 = h(DID_i\Vert ID_{SS}\Vert d_{ij-1}\Vert r_{i1}\Vert r_{i2})$	Checks d_i ?= $h(d_{ij-1}')$	
$\{EID_i', M_1, M_2, M_3, M_4,$	M_5 ?= $h(DID_i''\Vert ID_{SS}'\Vert d_{ij-1}'\Vert r_{i1}'\Vert r_{i2}')$	
$M_5\}$ $\longrightarrow$	Generates $r_{CS}\in Z_n^*$	

	Computes $K_{CS\text{-}SS}' = h(ID_{SS}'\|\|K_{CS})$ $M_6 = DID_i''\oplus K_{CS\text{-}SS}'$ $M_7 = h(DID_i''\|\|K_{CS\text{-}SS}')\oplus r_{CS}$ $M_8 = d_S\oplus r_{i2}'$ $M_9 = h(DID_i''\|\|K_{CS\text{-}SS}'\|\|d_S\|\|r_{i2}'\|\|r_{CS})$ $\{M_6, M_7, M_8, M_9\}$ $\longrightarrow$	Computes $DID_i''' = M_6\oplus K_{CS\text{-}SS}$ $r_{CS}' = M_7\oplus h(DID_i'''\|\|K_{CS\text{-}SS})$ $r_{i2}'' = M_8\oplus d_{Sl}$ Checks $M_9 ?= h(DID_i'''\|\|K_{CS\text{-}SS}\|\|d_{Sl}\|\|r_{i2}''\|\|r_{CS}')$ Generates $r_{S1}, r_{S2} \in Z_n^*$ Computes $M_{10} = r_{S1}\oplus K_{CS\text{-}SS}$ $K_{SS} = h(DID_i'''\|\|ID_{SS}\|\|r_{i2}''\|\|r_{CS}'\|\|r_{S2})$ $M_{11} = d_{Sl}\oplus r_{S2}$ $M_{12} = d_{Sl\text{-}1}\oplus r_{S2}$ $M_{13} = h(K_{CS\text{-}SS}\|\|SK_{SS}\|\|r_{S1})$ $\{M_{10}, M_{11}, M_{12}, M_{13}\}$ $\longleftarrow$
	Computes $r_{S1}' = M_{10}\oplus K_{CS\text{-}SS}'$ $r_{S2}' = M_{11}\oplus d_S$ $d_{Sl\text{-}1}' = M_{12}\oplus r_{S2}'$ Checks $d_S ? = h(d_{Sl\text{-}1}')$ Computes $SK_{CS} = h(DID_i''\|\|ID_{SS}'\|\|r_{i2}'\|\|r_{CS}\|\|r_{S2}')$ Checks $M_{13} ? = h(K_{CS\text{-}SS}'\|\|SK_{CS}\|\|r_{S1}')$ If $M_{13} ?\neq h(K_{CS\text{-}SS}'\|\|SK_{CS}\|\|r_{S1}')$ Rejects ID_{ss} $\longrightarrow$	If does not receive any message after ΔT Updates $l = l\text{-}1$
Computes $r_{CS}'' = M_{14}\oplus r_{i1}$ $r_{S2}'' = M_{15}\oplus r_{CS}''$ $SK_i = h(DID_i'\|\|ID_{SS}\|\|r_{i2}\|\|r_{CS}''\|\|r_{S2}'')$ Checks $M_{17} ?= h(DID_i'\|\|SK_i\|\|r_{CS}''\|\|r_{S2}''\|\|M_{16})$ If $M_{17} \neq h(DID_i'\|\|SK_i\|\|r_{CS}''\|\|r_{S2}''\|\|M_{16})$ Rejects EID_i $\longrightarrow$ Otherwise, updates $j = j\text{-}1$ Computes $EID_{i\_new} = M_{16}\oplus r_{i1}$ $E_{i\_new} = EID_{i\_new}\oplus h(c_i)$ $A_{i\_new} = h(DID_i'\oplus r_{cs}''\|\|c_i)$ $G_{i\_new} = G_i\oplus r_{cs}''$ Replaces $E_{i\_new}, A_{i\_new}, G_{i\_new}$ into E_i, A_i, G_i	Else updates $d_S = d_{Sl\text{-}1}'$ Computes $M_{14} = r_{i1}'\oplus r_{CS}$ $M_{15} = r_{CS}\oplus r_{S2}'$ $M_{16} = E_{KCS}(DID_i''\oplus r_{CS})\oplus r_{i1}'$ $M_{17} = h(DID_i''\|\|SK_{CS}\|\|r_{CS}\|\|r_{S2}'\|\|M_{16})$ $\{M_{14}, M_{15}, M_{16}, M_{17}\}$ $\longleftarrow$ If does not receive any message after ΔT Updates $d_i = d_{ij\text{-}1}'$	

Figure 6. Login and AKA phase of M2M$_{AKA}$-FS

sends a message $\{M_{14}, M_{15}, M_{16}, M_{17}\}$ to IoT device. If *CS* does not receive reject message from IoT device within ΔTtime, *CS* updates its commitment values $d_i=d_{ij\text{-}1}$ and $d_S=d_{Sl\text{-}1}$.

A5. Upon receiving the message from *CS*, *MC* computes $r_{CS}''=M_{14}\oplus r_{i1}$, $r_{S2}''=M_{15}\oplus r_{CS}''$. *MC* computes the SK, $SK_i=h(DID_i'\|ID_{SS}\|r_{i2}\|r_{CS}''\|r_{S2}'')$ and validates $M_{17}?=h(DID_i'\|SK_i\|r_{CS}''\|r_{S2}''\|M_{16})$. If it does not hold, IoT device sends a reject message to *CS* and the session is terminated. Otherwise, the authentication process is successful and *MC* updates its parameters by computing $j=j\text{-}1$, $EID_{i\_new}=M_{16}\oplus r_{i1}$, $E_{i\_new}=EID_{i\_new}\oplus h(c_i)$, $A_{i\_new}= h(DID_i'\oplus r_{CS}''\|c_i)$ and $G_{i_new}=G_i\oplus r_{CS}''$and replaces E_{i_new}, A_{i_new} and G_{i_new} into E_i, A_i and G_i,respectively.

Finally, IoT device can access *SS* on *MC* for any communication via *CS*, and a SK, $SK_i= (SK_{CS} = SK_{SS})$ with FS property is shared among IoT device, *CS* and *SS*.

4. Security and Performance Analysis

In this section we provide security and performance analysis of *M2M$_{AKA}$-FS*. Firstly, we prove the security correctness of *M2M$_{AKA}$-FS* by using BAN Logic. Then based on CK threat model, we demonstrate how *M2M$_{AKA}$-FS* achieves the five AKA protocol properties as well as the security and privacy goals. Finally, we compare the performance of *M2M$_{AKA}$-FS* with those of five earlier protocols: [2, 21, 36, 37, 40], Shuai et al., Kapito et al., Xiong et al., Yang et al. and Li et al.

4.1. Formal Analysis

With the formal validation BAN logic, we provide the proof of correctness of *M2M$_{AKA}$-FS*. We demonstrate that a SK with FS can be agreed successfully after the process of mutual authentication among *MC* and *SS*. Now, the basic notations of BAN-logic are given below:

- $P \mid\equiv X$: *P* believes *X*.
- $P \lhd X$: *P* sees *X*. i.e., *P* has received message containing *X*.
- $P \mid\sim X$: *P* said *X*. i.e., *P* has sent message containing *X*.
- $\#(X)$: *X* is fresh. i.e., *X* is usually a temporary value.
- $P \mid\Rightarrow X$: *P* has jurisdiction over *X*.
- (X, Y): *X* or *Y* is part of message (X, Y).
- $\langle X \rangle_Y$: *X* is encrypted with *Y*.
- $P \overset{K}{\leftrightarrow} Q$: *P* and *Q* can communicate with the shared secret key *K*.

Next, we introduce some BAN logic rules as follows:

1. **Message meaning rule**: $\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P,\ P \lhd \langle X \rangle_K}{P \mid\equiv Q \mid\sim X}$
   If *P* believes that *K* is a shared secret key between *P* and *Q* and *P* has received messages *X* containing *K*, *P* believes that *Q* has sent message *X*.

2. **Nonce-verification rule**: $\frac{P \mid\equiv \#(X),\ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$
   If *P* believes that *X* is a fresh message and *Q* has sent messages containing message *X*, P believes that *Q* believes message *X*.

3. **Jurisdiction rule**: $\frac{P \mid\equiv Q \mid\Rightarrow X, P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$
   If *P* believes that *Q* controls message *X* and *Q* believes message *X*, *P* believes message *X*.

4. **Freshness rule**: $\frac{P \mid\equiv \#(X)}{P \mid\equiv \#(X,Y)}$
   If *P* believes that *X* is a fresh message, *P* believes (*X*, *Y*) is fresh message.

5. **Belief Rule**: $\frac{P \mid\equiv (X,Y)}{P \mid\equiv (X)}$

If *P* believes message (*X*, *Y*), *P* believes message *X*.

M2M$_{AKA}$-FS needs to satisfy the following goals to ensure its security under BAN logic, using the above assumptions and postulates.

a. AKA-goals

AKA-Goal 1: AKA goals$MC|\equiv(MC \overset{SK}{\leftrightarrow} SS)$
AKA-Goal 2: $SS|\equiv(SS \overset{SK}{\leftrightarrow} MC)$
AKA-Goal 3: $MC|\equiv SS|\equiv(SS \overset{SK}{\leftrightarrow} MC)$
AKA-Goal 4: $SS|\equiv MC|\equiv (MC \overset{SK}{\leftrightarrow} SS)$

b. Key agreement with FS goals
AKA-FS-Goal 1: $CS|\equiv(CS \overset{d_i}{\leftrightarrow} MC)$
AKA-FS-Goal 2: $CS|\equiv(CS \overset{d_s}{\leftrightarrow} SS)$

Idealized form: The arrangement of the transmitted messages among IoT device with *MC*, *CS* and *SS* in *M*2*M_{AKA}*-*FS* to the idealized forms is as follows:

Message 1. $MC \rightarrow CS$: $< EIDi' >_{K_{CS}}$, $< M_1 >_{K_{CS}}$, $< M_2 >_{K_{CS}} < M_3 >_{d_{ij}'} < M_4 >_{d_{ij-1}'} < M_5 >_{d_{ij-1}'}$
Message 2. $CS \rightarrow SS$: $< M_6 >_{K_{CS-SS}'}, < M_7 >_{K_{CS-SS}'}, < M_8 >_{d_s}, < M_9 >_{K_{CS-SS}'}$
Message 3. $SS \rightarrow CS$: $< M_{10} >_{K_{CS-SS}}, M < M_{11} >_{d_{sm}}, < M_{12} >_{d_{sm-1}}, < M_{13} >_{K_{CS-SS}}$
Message 4. $CS \rightarrow MC$: M_{14}, M_{15}, $< M_{16} >_{K_{CS}}, < M_{17} >_{SK_{CS}},$

Assumptions: The following are the initial assumptions of *M*2*M_{AKA}*-*FS*:

A1: $MC|\equiv\#(r_{i1}, r_{i2}, d_{ij\text{-}1})$
A2: $CS|\equiv\#(r_{cs})$
A3: $SS|\equiv\#(r_{s1}, r_{s2}, d_{sm\text{-}1})$
A4: $MC|\equiv(MC \overset{K_{CS}}{\longleftrightarrow} CS)$
A5: $CS|\equiv(CS \overset{K_{CS}}{\longleftrightarrow} MC)$
A6: $CS|\equiv(CS \overset{K_{CS-SS}}{\longleftrightarrow} SS)$
A7: $SS|\equiv(SS \overset{K_{CS-SS}}{\longleftrightarrow} CS)$
A8: $MC|\equiv SS|\Longrightarrow MC \overset{SK}{\leftrightarrow} SS$
A9 $SS|\equiv MC|\Longrightarrow SS \overset{SK}{\leftrightarrow} MC$
A10: $CS|\equiv(CS \overset{d_{ij}}{\leftrightarrow} MC)$
A11: $CS|\equiv(CS \overset{d_{sm}}{\longleftrightarrow} SS)$

Proof: In the following, we prove the test goals, to show that *M*2*M_{AKA}*-*FS* provides a secure AKA with FS, using the BAN logic rules and the assumptions.

Based on message 1, we could derive:

Step 1. $CS \triangleleft (< EID_i >_{K_{CS}} < M_1 >_{K_{CS}} < M_2 >_{K_{CS}} < M_3 >_{d_{ij}} < M_4 >_{d_{ij-1}} < M_5 >_{d_{ij-1}})$

If step 1 holds according to assumptions A5 and A11 and message meaning rule, we can infer that*CS* believes the message is from *MC*:

Step 2. $CS|\equiv MC|\sim(< EID_i >_{K_{CS}} < M_1 >_{K_{CS}} < M_2 >_{K_{CS}} < M_3 >_{d_{ij}} < M_4 >_{d_{ij-1}} < M_5 >_{d_{ij-1}})$

According to assumption A1 and freshness rule, we get that *CS* believes the freshness of the message:

Step 3: $CS|\equiv\#(< EID_i >_{K_{CS}}< M_1 >_{K_{CS}}< M_2 >_{K_{CS}}< M_3 >_{d_{ij}}< M_4 >_{d_{ij-1}}< M_5 >_{d_{ij-1}})$

According to steps 2 and 3 and nonce verification rule, we get that *CS* believes that *MC* believes the message:

Step 4. $CS|\equiv MC|\equiv(< EID_i >_{K_{CS}}< M_1 >_{K_{CS}}< M_2 >_{K_{CS}}< M_3 >_{d_{ij}}< M_4 >_{d_{ij-1}}< M_5 >_{d_{ij-1}})$

According to step 4, assumptions A4 and A10 and belief rule, we get that *CS* believes that *MC* believes the established keys, K_{CS} and d_{ij} with *CS*:

Step 5. $CS|\equiv MC|\equiv(MC \overset{K_{CS}}{\longleftrightarrow} CS)$ and $CS|\equiv MC|\equiv(MC \overset{d_{ij}}{\leftrightarrow} CS)$

According to jurisdiction rule, we get that *CS* believes the established keys, K_{CS} and d_{ij} with *MC*:

Step 6. $CS|\equiv(CS \overset{K_{CS}}{\longleftrightarrow} MC)$ and $CS|\equiv(CS \overset{d_{ij}}{\leftrightarrow} MC)$ **(AKA-FS-Goal 1)**

Based on message 2, we derive

Step 7. $SS\lhd\ (< M_6 >_{K_{CS-SS}}< M_7 >_{K_{CS-SS}}< M_8 >_{d_s}< M_9 >_{K_{CS-SS}})$

According to assumption A7 and message meaning rule, we get that *SS* believes the message is from *CS*:

Step 8. $SS|\equiv CS|\sim(< M_6 >_{K_{CS-SS}}< M_7 >_{K_{CS-SS}}< M_8 >_{d_s}< M_9 >_{K_{CS-SS}})$

According to assumption A2 and the freshness rule, we get that *SS* believes the freshness of the message:

Step 9: $SS|\equiv\#(< M_6 >_{K_{CS-SS}}< M_7 >_{K_{CS-SS}}< M_8 >_{d_s}< M_9 >_{K_{CS-SS}})$

According to steps 8 and 9 and the nonce verification rule, we get that *SS* believes that *CS* believes the message:

Step 10. $SS|\equiv CS|\equiv(< M_6 >_{K_{CS-SS}}< M_7 >_{K_{CS-SS}}< M_8 >_{d_s}< M_9 >_{K_{CS-SS}})$

According to step 10, assumption A6 and belief rule, we get that SS believes that *CS* believes the established key $K_{CS\text{-}SS}$ with *SS*:

Step 11. $SS|\equiv CS|\equiv(CS \overset{K_{CS-SS}}{\longleftrightarrow} SS)$

According to jurisdiction rule, we get that *SS* believes the established key $K_{CS\text{-}SS}$ with *CS*:

Step 12. $SS|\equiv(SS \overset{K_{CS-SS}}{\longleftrightarrow} CS)$

According to steps 8, 9 and 10 and nonce verification rule, we conclude that *SS* believes that *MC* believes the established session key *SK* with *SS*:

Step 13. $SS|\equiv MC|\equiv (MC \overset{SK}{\leftrightarrow} SS)$ **(AKA-Goal 4)**

According to assumption A8 and the jurisdiction rule, we get that *SS* believes the established session key *SK* with *MC*:

Step 14. $SS|\equiv(SS\overset{SK}{\leftrightarrow} MC)$ **(AKA-Goal 2)**

Based on message 3, we derive

Step 15. $CS\lhd\ (< M_{10} >_{K_{CS-SS}}< M_{11} >_{d_{sm}}< M_{12} >_{d_{sm-1}}< M_{13} >_{K_{CS-SS}})$

According to A6 and message meaning rule, we get that *CS* believes the message is from *SS*:

Step 16. $CS|\equiv SS|\sim(< M_{10} >_{K_{CS-SS}}< M_{11} >_{d_{sm}}< M_{12} >_{d_{sm-1}}< M_{13} >_{K_{CS-SS}})$

According to assumption A3 and freshness rule, we get *CS* believes the freshness of the message:

Step 17: $CS|\equiv\#(< M_{10} >_{K_{CS-SS}}< M_{11} >_{d_{sm}}< M_{12} >_{d_{sm-1}}< M_{13} >_{K_{CS-SS}})$

According to steps 16 and 17 and nonce verification rule, we get *CS* believes that *SS* believes the message:

Step 18. $CS|\equiv SS|\equiv(< M_{10} >_{K_{CS-SS}}< M_{11} >_{d_{sm}}< M_{12} >_{d_{sm-1}}< M_{13} >_{K_{CS-SS}})$

According to step 18, assumptions A7 and A13 and belief rule, we get that *CS* believes that *SS* believes the established keys, $K_{CS\text{-}SS}$ and d_s with *SS*:

Step 19. $CS|\equiv SS|\equiv (SS \overset{K_{CS-SS}}{\longleftrightarrow} CS)$ and $CS|\equiv SS|\equiv (SS \overset{d_s}{\leftrightarrow} CS)$
According to steps 16 and 17 and nonce verification rule, we get that *CS* believes that *SS* believes the established key *SK* with *SS*:

Step 20. $CS|\equiv SS|\equiv (SS \overset{SK}{\leftrightarrow} CS)$

According to assumptions A6 and A11 and the jurisdiction rule, we get that *CS* believes the established keys, *SK* and d_s with *SS*:

Step 21. $CS|\equiv (CS \overset{SK}{\leftrightarrow} SS)$ and $CS|\equiv (CS \overset{d_s}{\leftrightarrow} SS)$ **(AKA-FS-Goal 2)**

Based on message 4, we could derive

Step 22. $MC \lhd(M_{14}M_{15}< M_{16} >_{K_{CS}}< M_{17} >_{SK_{CS}})$

According to assumption A4 and message meaning rule, we get that *MC* believes the message is from *CS*:

Step 23. $MC|\equiv CS|\sim(M_{14}M_{15}< M_{16} >_{K_{CS}}< M_{17} >_{SK_{CS}})$

According to assumption A2 and freshness rule, we get *MC* believes the freshness of the message:

Step 24: $MC|\equiv\#(M_{14}M_{15}< M_{16} >_{K_{CS}}< M_{17} >_{SK_{CS}})$

According to steps 23 and 24 and nonce verification rule, we get *MC* believes that *CS* believes the message:

Step 25. $MC|\equiv CS|\equiv(< M_{10} >_{K_{CS-SS}} M< M_{11} >_{d_{sm}} < M_{12} >_{d_{sm-1}} < M_{13} >_{K_{CS-SS}})$

According to step 25 and A5 and belief rule, we get that *MC* believes that *CS* believes the established key K_{CS} with *MC*:

Step 26. $MC|\equiv CS|\equiv(CS \overset{K_{CS}}{\longleftrightarrow} MC)$

According to steps 23, 24 and 25 and nonce verification rule, we get that *MC* believes that *SS* believes the established session key *SK* with *MC*:

Step 27. $MC|\equiv SS|\equiv(SS \overset{SK}{\leftrightarrow} MC)$ **(AKA-Goal 3)**

According to assumption A8 and the jurisdiction rule, we get that *MC* believes the established session key *SK* with *SS*:

Step 28. $MC|\equiv(MC \overset{SK}{\leftrightarrow} SS)$ **(AKA-Goal 1)**

4.2. Informal Security Analysis

In this subsection we give informal proof of the security of $M2M_{AKA}$-*FS* basing on CK security model. Firstly, we show that the protocol $M2M_{AKA}$-*FS* is SK secure under CK SK security and that it satisfies the AKA protocol properties as well as the security and privacy goals.

Proposition 1: $M2M_{AKA}$-*FS* is SK secure under the CK model.

Proof: To show that $M2M_{AKA}$-*FS* is SK secure under CK model, we need to show that $M2M_{AKA}$-*FS* satisfies the two conditions of **definition 2.11**.

Case1: Two uncorrupted parties output same SK

In the authentication and login phases of $M2M_{AKA}$-*FS*, *IoT device* and *SS* establish a SK, $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$ with the help of *CS*, which is used for future communication. The SK contains values DID_i, r_{i2}, r_{cs} and r_{s2} which are protected by secret materials K_{CS}, $K_{CS\text{-}SS}$, d_{ij}, and d_{sl}. The secret materials K_{CS}, $K_{CS\text{-}SS}$, d_{ij}, and d_{sl} can be computed and shared by legit parties only. Without these secret materials K_{CS}, $K_{CS\text{-}SS}$, d_{ij}, and d_{sl} a corrupted party cannot output a correct SK. Hence only legit IoT device and *SS* can compute the same SK, $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$.

Case 2: $\mathcal{A}$ computes correct SK with a negligible advantage.
For $\mathcal{A}$ to compute $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$, $\mathcal{A}$ needs to compute $DID_i = D_{K_{CS}}(EID_i)$, which requires the master key K_{CS} known to *CS* only, d_{ij-1} such that $d_{ij}=h(d_{ij-1})$ to get r_{i2} which is computationally infeasible basing on **definition 2.8** and **assumption 2.1** of hash function properties, d_{sl-1} to get r_{s2} which is computationally infeasible basing on **definition 2.8** and **assumption 2.1** of hash function properties and r_{cs} which requires $K_{CS\text{-}SS}$ known by *CS* and *SS* only. Since the hash function has a negligible advantage in finding the integers d_{ij-1} for the given $d_{ij} = h(d_{ij-1})$, i.e., $Adv_{\mathcal{A}}^{OW}(n) = Pr\left[\mathcal{A}\left(1^n, h(d_{ij-1})\right)\right] = d_{ij-1} \leq \omega(n)$ then $\mathcal{A}$'s advantage of computing correct SK in $M2M_{AKA}$-*FS* protocol, is negligible.

Proposition2.$M2M_{AKA}$-*FS*provides key control property of CK SK security.

Proof: In $M2M_{AKA}$-*FS*, the session key SK is formulated by parameters $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$ which has parameters DID_i and r_{i2} contributed by IoT device, ID_{SS} and r_{S2} contributed by *SS* and r_{CS} contributed by *CS*. Hence no entity forces *SK* to be preselected.

Proposition 3. $M2M_{AKA}$-*FS*provides UKS resilience.

Proof*:* In $M2M_{AKA}$-*FS* the session key $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$ has an authentication binding between SK and the identifiers of legal entities. The parameters DID_i, ID_{SS} in EID_i and M_2 can only be computed by *CS* using the long-term key K_{CS}. Only after successful authentication of the IoT device, *SS* agrees a key with the IoT device. Hence, *SS* cannot be coerced to be sharing a key with $\mathcal{A}$ impersonating the IoT device.

Proposition 4. $M2M_{AKA}$-*FS* provides known key security property.
Proof*:* In $M2M_{AKA}$-*FS* each session key $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$ is formulated with independent random numbers r_{i2},$r_{CS}$and r_{S2} which ensure that each session key is independent from the other. Hence the compromise of the past session keys does not guarantee compromise of the future session keys.

Proposition 5. $M2M_{AKA}$-*FS*provides key compromise impersonation resilience.

Proof*:* In $M2M_{AKA}$-*FS*, if $\mathcal{A}$ gets the long-term key K_{CS} of *CS*$\mathcal{A}$ may use it to obtain the real identity of IoT Device, $\mathcal{A}$ will still not be able to impersonate IoT Device to *CS*, due to the parameters d_{id} and $d_{id\text{-}1}$ in M_3 and M_4 that require inverse computation of hash function which is impossible as in **assumption 2.1**of the hash function. Similarly, the compromise of the long-term key $K_{CS\text{-}SS}$ will not be enough for $\mathcal{A}$ to impersonate *SS* to *CS* since $\mathcal{A}$ will not be able to compute M_{11} and M_{12} which requires the hash chain secret values of *SS*. Hence, $M2M_{AKA}$-*FS* provides key compromise impersonation resilience.

Proposition 6. $M2M_{AKA}$-*FS* provides mutual entity authentication.

Proof: In $M2M_{AKA}$-*FS*, *CS* is a trusted party and a bridge of communication between the IoT device and *SS*, and the mutual authentication among the three parties is achieved explicitly. Particularly when receiving login request message $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$, *CS* first restores DID_i using secret key K_{CS}. Then *CS* retrieves r_{i1}, ID_{SS}, r_{i2}, $d_{ij\text{-}1}$ and $M_5$and verifies the validity of IoT device by checking $M_5' ? = M_5$. In step 3 of section 3.4, when receiving message $\{M_6, M_7, M_8, M_9\}$ from *CS*, *SS* first retrieves DID_i, r_{cs} and r_{i2} by using $K_{CS\text{-}SS}$ and verifies the validity of the message by checking $M_9 ?= h(DID_i''' \| K_{CS\text{-}SS} \| d_{S1} \| r_{i2}'' \| r_{CS}')$. In step 4 when getting the response message $\{M_{10}, M_{11}, M_{12}, M_{13}\}$ from *SS*, *CS* retrieves r_{s1} and r_{s2} using $K_{CS\text{-}SS}$, calculates SK_{CS} and $M_{13}$and authenticates *SS* by checking M_{13} $?= h(K_{CS\text{-}SS}' \| SK_{CS} \| r_{S1}')$. Similarly, when obtaining message $\{M_{14}, M_{15}, M_{16}, M_{17}\}$ from *CS*, the IoT device retrieves r_{cs} and r_{s2} and calculates SK_i and M_{17}. Finally, the validity of the message can be affirmed by the IoT device if M_{17} $?= h(DID_i' \| SK_i \| r_{CS}'' \| r_{S2}'' \| M_{16})$.

Proposition 7. $M2M_{AKA}$-*FS* provides IoT device anonymity.
Proof: In $M2M_{AKA}$-*FS*, the plaintext of IoT device real identity DID_i is not contained in any message and $\mathcal{A}$ cannot get IoT device identity from the communication messages directly. IoT device real identity DID_i is implied in message $\{EID_i, M_6\}$. When receiving the login request message $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$ from the IoT device, with the master key K_{CS}, *CS* can recover real identity by computing $DID_i = D_{K_{CS}}(EID_i)$. When receiving message $\{M_6, M_7, M_8, M_9\}$

from *CS*, *SS* recovers $DID_i = M_6 \oplus K_{CS\text{-}SS}$ by using $K_{CS\text{-}SS}$. Without knowing K_{CS} and $K_{CS\text{-}SS}$, $\mathcal{A}$ cannot reveal the IoT device's real identity from the communication messages.

Proposition *8*. *M*2*M*

Proposition *8*. $M2M_{AKA}$*-FS* provides unlinkability.

Proof: In $M2M_{AKA}$*-FS*, the random numbers r_{i1} and r_{i2} are generated by the IoT device for each session, which makes the login request message $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$ of one session different from those of other sessions. Furthermore, $M2M_{AKA}$*-FS* uses encrypted amplified dynamic identities EID_i with the one-way hash function. Only *CS* can get the real identity of IoT device hence there is no way for $\mathcal{A}$ to link any relationship between different sessions even if $\mathcal{A}$ could capture the messages $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$, $\{M_6, M_7, M_8, M_9\}$, $\{M_{10}, M_{11}, M_{12}, M_{13}\}$ and $\{M_{14}, M_{15}, M_{16}, M_{17}\}$ during the protocol run of $M2M_{AKA}$*-FS* due to the computation hardness of a one-way has function.

Proposition 9. $M2M_{AKA}$*-FS* provides a SK agreement with FS.

Proof: To prove this proposition we will prove it in two parts. Firstly, we show that $M2M_{AKA}$*-FS* provides a SK agreement secondly we show that the SK agreed is FS secure.

[SK agreement] Proof: In the authentication and login process of $M2M_{AKA}$*-FS*, the IoT device and *SS* establish a SK, $SK = h(DID_i \| ID_{SS} \| r_{i2} \| r_{CS} \| r_{S2})$ with the help of *CS*, which is used for future communication. The SK contains the IoT device's contribution to DID_i and r_{i2} and *SS*'s contribution to ID_{SS} and r_{S2}. Without these private values, any third party cannot predetermine the SK. Therefore, $M2M_{AKA}$*-FS* provides a SK agreement.

[Forward Secrecy] Proof: FS is provided by ensuring one-time use of the secret parameters employed in the login and AKA phase. In $M2M_{AKA}$*-FS*, OTS values are d_{ij} and d_{sl}. For $\mathcal{A}$ who captures d_{ij} and d_{sl} will not be able to use these secret values to get future SKs, since in the next login the protocol will use $d_{ij\text{-}1}$ and $d_{sl\text{-}1}$ as secret authentication credentials. But $d_{ij\text{-}1}$ and $d_{sl\text{-}1}$ cannot be computed in polynomial time since it requires computing $d_{ij\text{-}1}$ and $d_{sl\text{-}1}$ such that $h(d_{ij\text{-}1}) = d_{ij}$ and $h(d_{sl\text{-}1}) = d_{sl}$ which is computationally infeasible as per **definitions 2.8** and **2.9**. Thus the probability of $\mathcal{A}$ getting future SKs having known the present secret parameters is negligible.

Proposition 10. $M2M_{AKA}$*-FS* resists against replay attacks.

Proof: Suppose $\mathcal{A}$ eavesdrops valid messages and then resends it at his discretion. In $M2M_{AKA}$*-FS*, if $\mathcal{A}$ resends $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$, *CS* checks $d_{id\text{-}1}$ from M_4 and checks whether $d_i ?= h\ (d_{ij\text{-}1})$. Since $d_{ij\text{-}1}$ is new in each session as per the design of the FS framework, it will not be equal, hence the session will be terminated. Similarly, the random numbers r_{i1}, r_{i2},r_{cs}, r_{s1} and r_{s2} used in first, second, and third messages, are always new in each session, if $\mathcal{A}$ replays the recorded message, it will not match with the message freshness support M_5, M_9, M_{13}, and M_{17}. Thus $M2M_{AKA}$*-FS* resists against replay attacks.

Proposition 11. $M2M_{AKA}$*-FS* resists MITM attack.

Proof*:* $M2M_{AKA}$*-FS* is secure against MITM attacks as the $\mathcal{A}$ cannot fake the IoT device, *CS* and *SS* without the knowledge of K_{CS}, $K_{CS\text{-}SS}$, d_{ij} and d_{sl}. If $\mathcal{A}$ can capture the first message $\{EID_i, M_1, M_2, M_3, M_4, M_5\}$ to impersonate the IoT device, $\mathcal{A}$ will still not be able, since it requires knowledge of true IoT device identity which is secured by a master secret key K_{CS}. Furthermore, $\mathcal{A}$ needs to know the one-time hash chain secret value d_{ij} which is not possible due to **definition 2.8** FS property of a hash function. Similarly, if $\mathcal{A}$ captures the second message $\{M_6, M_7, M_8, M_9\}\mathcal{A}$ still will not be able to impersonate *CS* since only one who has the secret material $K_{CS\text{-}SS}$ can successfully compute the second message. If $\mathcal{A}$ captures the third message $\{M_{10}, M_{11}, M_{12}, M_{13}\}$, $\mathcal{A}$ will still not be able to impersonate *SS* since $\mathcal{A}$ will need the secret material $K_{CS\text{-}SS}$ which is only known to *SS* and *CS*, and d_{ij} and d_{sl} cannot be obtained due to **definition 2.9** FS property of a hash function.

4.3. Complexity Analysis

This section provides various comparisons among *M*2*M*$_{AKA}$-*FS* and well-known related protocols including Shuai et al.'s protocol, Xiong et al.'s protocol, Kapito et al.'s protocol, Yang et al.'s protocol and Li et al.'s protocol. First of all, we will focus on feature comparisons to know the distinctive feature differences among them. After that, computation and communication analysis follows, to show IoT environmental fitness of them.

4.3.1. Features Comparison

In this section, we give a detailed examination of the protocol features to see how much it satisfies the protocol design goals. The requirements for the design of *M*2*M*$_{AKA}$-*FS* are compared with those of the five earlier protocols, Shuai et al., Xiong et al., Kapito et al., Yang et al. and Li et al. as shown in Table 2.

4.3.2. Computational Overhead Analysis

In this subsection, we analyze the computational overheads in terms of time taken for each step in the protocol run. To facilitate the evaluation of computation costs, we use a scale provided by Shuai et al. [2]. They provided computation costs as in Table 3 .Table 4 shows the computational cost comparisons of *M*2*M*$_{AKA}$-*FS* and the related protocols. Results from Table 4 show that *M*2*M*$_{AKA}$-*FS* is more efficient than Shuai et al.'s protocol and Li et al.'s protocol. Kapito et al.'s protocol is slightly more efficient than *M*2*M*$_{AKA}$-*FS* but their protocol lacks SK agreement with FS feature which is very important in IoT environment. Although two protocols of Xiong et al. and Yang et al. are very efficient in computation cost, their protocols lack the security and privacy features as explained inTable 2. Yang et al.'s protocol does not provide anonymity and unlinkability and is also not resistant to various attacks whereas Xiong et al.'s protocol does not provide unlinkability.

Table 2. Features comparison

Feature / Protocol	AKA Goal 1	AKA Goal 2	AKA Goal 3	AKA Goal 4	AKA Goal 5	AKA Goal 6
Shuai et al.	No	Yes	Yes	Yes	Yes	Yes
Xiong et al.	No	Yes	Yes	Yes	Yes	Yes
Kapito et al.	Yes	Yes	No	Yes	Yes	Yes
Yang et al.	Yes	Yes	Yes	No	No	No
Li et al.	Yes	Yes	No	No	Yes	Yes
*M*2*M*$_{AKA}$-*FS*	Yes	Yes	Yes	Yes	Yes	Yes

AKA Goal 1: Lightweight property, AKA Goal 2: Mutual authentication, AKA Goal 3: SK agreement with FS, AKA Goal 4: Resilience to various attacks, AKA Goal 5: Anonymity, AKA

Goal 6: Unlinkability

Table 3. Computation cost

Entity / Protocol	IoT device	CS	SS	Total
Shuai et al.	$T_M + T_{QR} + 7T_h$ (1.17452)	$T_{QR} + 7T_h$ (1.17383)	$T_M + 5T_h$ (0.00414)	$2T_M + 2T_{QR} + 19T_h$ (2.3524)
Xiong et al.	$2T_{E/D} + 9T_h$ (0.00729)	$2T_{E/D} + 11\ T_h$ (0.00867+(N-1)*0.00069)	$4T_h$ (0.00276)	$4T_{E/D} + 24T_h$ (0.01872) + (N-1)*0.00069
Kapito et al.	$1T_{fe} + 9T_h$ (0.51421)	$2T_{E/D} + 9T_h$ (0.00729)	$4T_h$ (0.00276)	$T_{fe} + 2T_{E/D} + 22T_h$ (0.52426)
Yang et al.	$3T_{E/D} + 8T_h$ (0.00714)	$5T_{E/D} + 14T_h$ (0.01236)	$T_{E/D} + 7T_h$ (0.00537)	$8T_{E/D} + 27T_h$ (0.02295)
Li et al.	$2T_{\text{exp}} + 8T_h$ (1.02152)	$T_{exp} + 9T_h$ (0.51421)	$4T_h$ (0.00276)	$3T_{exp} + 21T_h$ (1.53849)
*M*2*M*$_{AKA}$-*FS*	$T_{fe} + 11T_h$ (0.51559)	$2T_{E/D} + 11T_h$ (0.00867)	$4T_h$ (0.00276)	$T_{fe} + 2T_{E/D} + 26T_h$ (0.52702)

Regardless of *M*2*M*$_{AKA}$-*FS* having computation cost slightly higher than the protocols of Xiong et al., Yang et al. and Kapito et al., *M*2*M*$_{AKA}$-*FS* is still the most suitable protocol for IoT environment since it provides the required features like SK agreement with FS, provides anonymity and unlinkability, and is also resilient to various attacks. Thus the amount of computation cost is compensated by the high security and privacy features that *M*2*M*$_{AKA}$-*FS* offers.

4.3.3. Communication Overhead Cost

To facilitate the analysis of communication overhead, we assume the length of the IoT device's identity, user's identity, pseudonym identity and the corresponding password are all 128 bits. The length of the secret key, the random number, the output of hash function and message authentication code (MAC) are all 160 bits. The length of the time stamp, the ECC point multiplication and the cipher text block in symmetric encryption/decryption are 32 bits, 320 bits and 256 bits respectively. To provide sufficient security, 1024-bits modulus is used for modular exponentiation and inversion operations. Therefore, the length of modular squaring is 1024-bits.
In *M*2*M*$_{AKA}$-*FS*, the transmitted messages {EID_i, M_1, M_2, M_3, M_4, M_5}, {M_6, M_7, M_8, M_9}, {M_{10}, M_{11}, M_{12}, M_{13}} and {M_{14}, M_{15}, M_{16}, M_{17}} require {256+160+160+160+160+160} = 1,056 bits, {160+160+160+160} = 640 bits, {160+160+160+160} = 640 bits and {160+160+160+256} = 736 bits, respectively. Therefore, the cumulative communication overhead of *M*2*M*$_{AKA}$-*FS* is 3,072 bits. The cumulative overheads of Shuai et al.'s protocol, Xiong et al.'s protocol, Kapito et al.'s protocol, Yang et al.'s protocol and Li et al.'s protocol are shown in Table 4.

Although there is advantage in the communication overhead of the other protocols, it is justifiable because *M*2*M*$_{AKA}$-*FS* offers better security and more functionality features as compared to these protocols shown in Table 2. We always believe that security is at least as important as efficiency for an AKA protocol and thus it is not advisable to significantly reduce security to increase marginal efficiency [46].

Table 4. Communicational cost comparison

Entity / Protocol	IoT device	CS	SS	Total
Shuai et al.	1,024+160+32 (1,216 bits)	1,024+4*160+32 (1,696 bits)	1,024+2*160 (1,344 bits)	(4,256 bits)
Xiong et al.	2*160+128+256 (704 bits)	2*256+2*160+128 (960 bits)	256*160 (416 bits)	(1,824 bits)
Kapito et al.	256+3*160 (736 bits)	7*160+256 (1,376 bits)	2*160 (320 bits)	(2,432 bits)
Yang et al.	2*160 (320 bits)	3*160 (480 bits)	(160 bits)	(960 bits)
Li et al.	5*160 (800 bits)	7*160 (1,120 bits)	2*160 (320 bits)	(2,240 bits)
$M2M_{AKA}$-FS	256+5*160 (1,056 bits)	7*160+256 (1,376 bits)	4*160 (640 bits)	(3,072 bits)

5. CONCLUSION

In this paper, we have designed a new machine-to-machine authenticated key agreement with FS for IoT environment $M2M_{AKA}$-FS . Firstly, we drew the lightweight property of IoT environment and the required features that authenticated key agreement protocols should satisfy by reviewing and analyzing some previous protocols. Secondly, we designed an FS framework based on hash chain OTS values. Thirdly, designed a machine-to-machine authenticated key agreement protocol with FS ($M2M_{AKA}$-FS) for IoT environment by adopting the FS framework. The building block of $M2M_{AKA}$-FS is based on the intractability of one-way hash function, symmetric cryptosystem, fuzzy commitment scheme, bitwise XOR and concatenation operations to complete the protocol successfully. We finally analyzed the design of $M2M_{AKA}$-FS in three ways, formal security analysis by using BAN logic, informal analysis by using cryptanalysis and complexity analysis by comparing computation and communication overhead with earlier protocols. It was determined that the complexity, security and privacy of $M2M_{AKA}$-FS was better than in earlier related protocols.

Future research should focus on the implementation of $M2M_{AKA}$-FS over the real IoT environment for optimization of the protocol. Furthermore, the proposed FS framework should be applied to the various AKA protocols by applying various requirements for the environments.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

ACKNOWLEDGEMENTS

The results in this paper are part of Mr. Batamu Anderson Chiphiko’s Master degree thesis. Corresponding author is Hyunsung Kim. This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1B04032598).

REFERENCES

[1] B.P. Beauchamp, L. Corneal, N. Kandalaft, Privacy of biofeedback human interfacing devices. Int J Adv Appl Sci ISSN, 2252(8814), 8814 (2022).

[2] Shuai, M., Xiong, L., Wang, C., & Yu, N. (2020). A secure authentication scheme with forward secrecy for industrial internet of things using Rabin cryptosystem. Computer Communications, 160, 215-227.

[3] Williams, P., Dutta, I. K., Daoud, H., & Bayoumi, M. (2022). A survey on security in internet of things with a focus on the impact of emerging technologies. Internet of Things, 19, 100564.

[4] Ren, Z., Liu, X., Ye, R., & Zhang, T. (2017, July). Security and privacy on internet of things. In 2017 7th IEEE international conference on electronics information and emergency communication (ICEIEC) (pp. 140-144). IEEE.

[5] Lanner. (2018). Examples of IoT devices in your next smart home. Available: https://www.lanner-america.com/blog/5-examples-iot-devices-next-smart-home. Accessed: October 10, 2020.

[6] Chaudhary, S., Johari, R., Bhatia, R., Gupta, K., & Bhatnagar, A. (2019, April). CRAIoT: concept, review and application (s) of IoT. In 2019 4th international conference on internet of things: Smart innovation and usages (IoT-SIU) (pp. 1-4). IEEE.

[7] Grizhnevich, A. (2018). IoT for smart cities: Use cases and implementation strategies. Science Soft.

[8] Chawla, R. (2021). Study of security threats and challenges in internet of things systems. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 12(2), 1154-1166.

[9] Thapa, A., Dhapola, C. S., & Saini, H. (2022, August). Security Analysis of User Authentication and Methods. In Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing (pp. 564-572).

[10] Chen, L., Wang, Z., Wu, J., Guo, Y., Li, F., & Li, Z. (2022). Dynamic threshold strategy optimization for security protection in Internet of Things: An adversarial deep learning-based game-theoretical approach. Concurrency and computation: practice and experience, e6944.

[11] Lundgren, M., & Padyab, A. (2021). Security and privacy of smart homes: issues and solutions. Security and Privacy in the Internet of Things: Architectures, Techniques, and Applications, 235-260.

[12] Schneller, L., Porter, C. N., & Wakefield, A. (2022). Implementing converged security risk management: drivers, barriers, and facilitators. Security Journal, 1-17.

[13] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P., & Sikdar, B. (2019). A survey on IoT security: application areas, security threats, and solution architectures. IEEE Access, 7, 82721-82743.

[14] Saleh, I. A., Kamal, M. A., Ibrahim, L. M., Alsaif, O., & Yahya, M. A. (2020, August). Using monkey optimization algorithm to detect neris botnet. In Journal of Physics: Conference Series.

[15] Yu, S., Jho, N., & Park, Y. (2021). Lightweight three-factor-based privacy-preserving authentication scheme for iot-enabled smart homes. IEEE Access, 9, 126186-126197.

[16] Pal, S., Hitchens, M., Rabehaja, T., & Mukhopadhyay, S. (2020). Security requirements for the internet of things: A systematic approach. Sensors, 20(20), 5897.

[17] Diffie, W., & Hellman, M. E. (1979). Privacy and authentication: An introduction to cryptography. Proceedings of the IEEE, 67(3), 397-427.

[18] Lee, D. H., & Lee, I. Y. (2020). A lightweight authentication and key agreement schemes for IoT environments. Sensors, 20(18), 5350.

[19] Alzahrani, B. A. (2021). Secure and efficient cloud-based IoT authenticated key agreement scheme for e-health wireless sensor networks. Arabian Journal for Science and Engineering, 46(4), 3017-3032.

[20] Kapito, B., Nyirenda, M., & Kim, H. (2021). Privacy-Preserving Machine Authenticated Key Agreement for Internet of Things. International Journal of Computer Networks and Communications, 13(2), 99-120.

[21] Zhang, J., Yan, Z., Fei, S., Wang, M., Li, T., & Wang, H. (2021). Is Today's End-to-End Communication Security Enough for 5G and Its Beyond?. IEEE Network, 36(1), 105-112.

[22] Seliem, M., Elgazzar, K., & Khalil, K. (2018). Towards privacy preserving iot environments: a survey. Wireless Communications and Mobile Computing, 2018, 1-15.

[23] Fitwi, A., & Chen, Y. (2021, July). Secure and privacy-preserving stored surveillance video sharing atop permissioned blockchain. In 2021 International Conference on Computer Communications and Networks (ICCCN) (pp. 1-8). IEEE.

[24] Imteaj, A., Thakker, U., Wang, S., Li, J., & Amini, M. H. (2021). A survey on federated learning for

resource-constrained IoT devices. IEEE Internet of Things Journal, 9(1), 1-24.

[25] Kornaros, G. (2022). Hardware-assisted machine learning in resource-constrained IoT environments for security: review and future prospective. IEEE Access, 10, 58603-58622.

[26] Gunnarsson, M., Brorsson, J., Palombini, F., Seitz, L., & Tiloca, M. (2021). Evaluating the performance of the OSCORE security protocol in constrained IoT environments. Internet of Things, 13, 100333.

[27] Lee, H., Kang, D., Ryu, J., Won, D., Kim, H., & Lee, Y. (2020). A three-factor anonymous user authentication scheme for Internet of Things environments. Journal of Information Security and Applications, 52, 102494.

[28] Hajian, R., ZakeriKia, S., Erfani, S. H., & Mirabi, M. (2020). SHAPARAK: Scalable healthcare authentication protocol with attack-resilience and anonymous key-agreement. Computer Networks, 183, 107567.

[29] Wang, J., Zhu, Y., & Maqbool, S. (2021). An efficient hash-based authenticated key agreement scheme for multi-server architecture resilient to key compromise impersonation. Digital Communications and Networks, 7(1), 140-150.

[30] Pardeshi, M. S., Sheu, R. K., & Yuan, S. M. (2022). Hash-chain fog/edge: A mode-based hash-chain for secured mutual authentication protocol using zero-knowledge proofs in fog/edge. Sensors, 22(2), 607.

[31] Das, M. L. (2009). Two-factor user authentication in wireless sensor networks. IEEE transactions on wireless communications, 8(3), 1086-1090.

[32] Yeh, H. L., Chen, T. H., Liu, P. C., Kim, T. H., & Wei, H. W. (2011). A secured authentication protocol for wireless sensor networks using elliptic curves cryptography. Sensors, 11, 4767-4779.

[33] Liu, Y., Peng, Y., Wang, B., Bai, X., Yuan, X., & Li, G. (2013). The Internet of Things security architecture based IBE integration with the PKI/CA. Proceedings of the Advanced Science and Technology Letters, Harbin, China, 18-20.

[34] Ndibanje, B., Lee, H. J., & Lee, S. G. (2014). Security analysis and improvements of authentication and access control in the internet of things. Sensors, 14(8), 14786-14805.

[35] Shivraj, V. L., Rajan, M. A., Singh, M., & Balamuralidhar, P. (2015, February). One time password authentication scheme based on elliptic curves for Internet of Things (IoT). In 2015 5th National Symposium on Information Technology: Towards New Smart World (NSITNSW) (pp. 1-6). IEEE.

[36] Xiong, L., Peng, D., Peng, T., Liang, H., & Liu, Z. (2017). A lightweight anonymous authentication protocol with perfect forward secrecy for wireless sensor networks. Sensors, 17(11), 2681.

[37] Li, X., Niu, J., Kumari, S., Wu, F., Sangaiah, A. K., & Choo, K. K. R. (2018). A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments. Journal of Network and Computer Applications, 103, 194-204.

[38] Aghili, S. F., Jolfaei, A. A., & Abidin, A. (2020). SAKE+: Strengthened Symmetric-Key Authenticated Key Exchange with Perfect Forward Secrecy for IoT. IACR Cryptol. ePrint Arch., 2020, 778.

[39] Avoine, G., Canard, S., & Ferreira, L. (2020). Symmetric-key authenticated key exchange (SAKE) with perfect forward secrecy. In Topics in Cryptology–CT-RSA 2020: The Cryptographers' Track at the RSA Conference 2020, San Francisco, CA, USA, February 24–28, 2020, Proceedings (pp. 199-224). Springer International Publishing.

[40] Yang, Z., He, J., Tian, Y., & Zhou, J. (2019). Faster authenticated key agreement with perfect forward secrecy for industrial internet-of-things. IEEE Transactions on Industrial Informatics, 16(10), 6584-6596.

[41] Ahmad, M., Singh, S., & Khurana, S. (2021). Cryptographic one-way hash function generation using twelve-terms 4D nonlinear system. International Journal of Information Technology, 13(6), 2295-2303.

[42] Underwood, R. G. (2022). Symmetric Key Cryptography. In Cryptography for Secure Encryption (pp. 139-171). Cham: Springer International Publishing.

[43] Dabbagh, Y. S., & Saad, W. (2019). Authentication of wireless devices in the Internet of Things: Learning and environmental effects. IEEE Internet of Things Journal, 6(4), 6692-6705.

[44] Bugeja, J., Jacobsson, A., & Davidsson, P. (2016, August). On privacy and security challenges in smart connected homes. In 2016 European Intelligence and Security Informatics Conference (EISIC) (pp. 172-175). IEEE.

[45] Alam, S., Siddiqui, S. T., Ahmad, A., Ahmad, R., & Shuaib, M. (2020). Internet of things (IoT) enabling technologies, requirements, and security challenges. In Advances in Data and Information Sciences: Proceedings of ICDIS 2019 (pp. 119-126). Springer Singapore.

[46] Wu, T. Y., Meng, Q., Kumari, S., & Zhang, P. (2022). Rotating behind security: A lightweight authentication protocol based on iot-enabled cloud computing environments. Sensors, 22(10), 3858.

AUTHORS

Batamu Anderson Chiphiko received a B.E. degree in Mathematical Science Education from Polytechnic of the University of Malawi and is currently a Masters Degree student with the Department of Mathematics, Chancellor College, University of Malawi. He was worked as a part-time lecturer at Domasi College, Malawi. He has been a Mathematics teacher at Ntcheu CDSS. His research interest is in Cryptography, Information Security, Cryptographic Protocol, Privacy, Internet of Things and Cryptanalysis. His Master thesis proposal is "M2M$_{AKA}$-FS Machine to Machine Authenticated Key Agreement with Forward Secrecy for Internet of Things."

Hyunsung Kim received the M.Sc. and Ph.D. degrees in computer engineering from Kyungpook National University, Korea, in 1998 and 2002, respectively. He is a Professor at the School of Computer Science, Kyungil University, Korea from 2012. Furthermore, he is currently a visiting professor at the Department of Mathematical Sciences, Chancellor College, University of Malawi, Malawi from 2015. He also was a visiting researcher at Dublin City University in 2009. From 2000 to 2002, he worked as a senior researcher at Ditto Technology. He had been an associate professor from 2002 to 2012 with the Department of Computer Engineering, Kyungil University. His research interests include cryptography, VLSI, authentication technologies, network security, ubiquitous computing security, and security protocol.