Temporary Identification Management System Using UNIX Time for IoT Device Privacy Protection

Koki Mizoguchi
Department of Informatics, The Graduate University for Advanced Studies
2–1–2, Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430, JAPAN.
e-mail: mizoguchi-koki@nii.ac.jp

Somchart Fugkeaw
Sirindhorn International Institute of Technology, Thammasat University
131 M.5 Tiwanont Rd., Bangkadi, PathumThani, 12000, THAILAND.
e-mail: somchart@siit.tu.ac.th

Masahito Kumazaki, Hirokazu Hasegawa, Hiroki Takakura
Center for Strategic Cyber Resilience R&D, National Institute of Informatics
2–1–2, Hitotsubashi, Chiyoda-ku, Tokyo, 101–8430, JAPAN.
e-mail: {kumazaki,hasegawa,takakura}@nii.ac.jp

***Abstract*—With the rapid growth of Internet of Things (IoT) devices, privacy concerns regarding device identifiers have become increasingly significant. Authentication and Key Exchange (AKE) protocols are essential for securing IoT environments, but many implementations transmit device identifiers in plaintext or hashed form, which can lead to privacy issues for device users. On the other hand, session-based ID anonymization systems, which change device identifiers every time authentication occurs, require reading and writing on Non-Volatile Memory (NVM), such as flash memory, which consumes more energy and has limited write endurance. This paper proposes a novel ID management system for generating temporary device identifiers using UNIX time, which does not require reading and writing on NVM. The system is considered a communication and update process that is delay resilient. The effectiveness of the proposed system is also demonstrated in terms of computational and communication costs compared to three baseline systems. The proposed system is concluded to be one of the effective solutions for protecting the privacy of resource-constrained IoT devices and their users.**

Keywords-IoT; Device Identifier; Privacy; ID anonymization; UNIX time.

I. Introduction

The global share of the Internet of Things (IoT) is increasing at an accelerating rate. Transforma Insights estimates that the number of IoT connections will reach 40.6 billion by 2034 [1]. Authentication and Key Exchange (AKE) protocols are essential to securing IoT environments. In most AKE protocols, IoT devices must provide their IDs to the authentication server for identification. IoT device IDs contain manufacturing information, such as the maker name, product model, and firmware version; hardware IDs, such as Media Access Control address (Mac address), International Mobile Equipment Identity (IMEI), and serial number; owner information, such as owner registered information and relationship with owner's account; and location information, such as installed location and local network IDs.

Until now, many AKE protocols have been proposed, and the ways of providing IDs of IoT devices are classified as plaintext, hashed, and session-based ID anonymization systems.

Messages repeatedly sent from the same devices result in the same hash value, even if the device identifier is hashed. It enables eavesdroppers to link different communications originating from the same devices. Eavesdroppers can recognize communication between specific IoT devices and authentication servers, observe the frequency and intervals of communication, and infer device usage patterns by observing the amount or size of transmitted data. Since many IoT devices exhibit regular and identifiable communication patterns based on user interaction, this information can be used to infer device usage patterns. For instance, a smart lock sends its hashed ID at a specific time every day, and eavesdroppers can infer the user's typical leave or return home time and the user's behavioral patterns. If it is easy to estimate original IDs, such as MAC address, phone number, and other regular and short IDs, a rainbow table attack and a dictionary attack are enabled. According to Choudhary [2], privacy concerns regarding device identifiers arise not only from the exposure of raw identifiers but also from the ability to associate seemingly anonymous data with behavioral patterns. It is emphasized that even anonymized or encrypted transmissions can leak sensitive information when analyzed over time.

Overall, plaintext and hashed IDs are not sufficient to protect the privacy of IoT devices and their users due to the risk of eavesdropping and inference of device usage patterns.

To address to prevent inference of device usage patterns using hashed IDs, session-based ID anonymization systems are proposed. In this systems, IDs are changed every time authentication occurs. The IoT device and the authentication server generate temporary IDs based on the state. This approach prevent inference of device usage patterns through

ISBN: 978-1-68558-295-1

eavesdropping on IDs, like plaintext and hashed IDs. However, it requires reading and writing operation on Non-Volatile Memory (NVM), such as flash memory, to store the state. Flash memory is representative of NVM, which is widely used in IoT devices. There are two concerns about using it. First, reading and writing on flash memory consumes more energy than volatile memory, such as DRAM [3][4]. Second, NVM exhibits limited write endurance, meaning it can only sustain a finite number of write operations before experiencing failure or degradation [5]. Ferroelectric RAM (FRAM/FeRAM) is a promising alternative to flash memory for non-volatile storage, offering higher endurance and lower energy consumption, though it remains more expensive and less widely available [4][6].

In summary, the plaintext and hashed IDs are not sufficient to protect the privacy of IoT devices and their users due to the risk of eavesdropping and inference of device usage patterns. Session-based ID management systems can protect the privacy of IoT devices and their users by changing IDs every time authentication occurs, but they require reading and writing on NVM, which consumes more energy and has limited write endurance.

To address these issues, this research proposes a new system to generate temporary IDs using UNIX time, which does not require reading and writing on NVM, and IoT devices' IDs are changed every certain times. UNIX time is used despite using state, which is synchronized between IoT devices and the authentication server. The generated temporary IDs are changed at certain times.

The paper is organized as follows: Section II describes related work and classification of ID management system. Section III describes the proposed system. Section IV provides a comparative evaluation of the proposed system against three baseline approaches, focusing on computational cost, communication overhead, and the achieved privacy level. Section V discusses the limitations of the proposed system and these potential solutions. Section VI concludes the paper.

II. Related Work

This section describes related work on ID anonymization systems for IoT devices.

Braeken proposed a PUF-based P2P (Peer-to-Peer) AKE protocol for IoT devices [7]. In this protocol, the IoT devices' IDs are not anonymized but are in plaintext.

Badhib et al. proposed a robust CSS (Client Server System) AKE protocol for IoT devices [8]. This protocol adopts a session-based ID anonymization system. The IoT device sends its ID, which is masked with the shared key and track sequence, to the authentication server. They are changed every time authentication occurs. However, the protocol requires reading and writing on NVM to store them in the IoT device.

For large-scale smart IoT applications, Chen et al. proposed a novel authentication scheme that models and supports the entire lifecycle of IoT device authentication, from manufacturing to daily use and resetting [9]. However, the IoT device ID (denoted as the smart device's unique identity) is transmitted in plaintext.

Alizadeh et al. proposed anonymous ticket-based authentication protocol for the IoT [10]. In this protocol, the IoT device ID is anonymized using arias ID. The combination of a hashed ID, secret, and nonce is a critical component of the proposed protocol's approach to sensor anonymity. Specifically, each alias ID is meticulously constructed from a hashed ID, the Sensor Node's (SN) secret value (denoted as ID_{SN}), and a nonce. This intricate composition significantly impedes the identification of an object's true identity, as malicious actors would be required to possess knowledge of the object's secret to ascertain its real ID. Employing a one-way hash function for each object's ID renders its decoding practically unfeasible. However, provided with the context in which fixed IDs are transmitted, it becomes possible to collect information, such as the communication time and interval of specific devices.

Nimmy et al. proposed a PUF-based CSS AKE protocol for IoT devices [11]. This protocol also adopts a session-based ID anonymization system. The IoT device sends its ID, which is masked with the state, to the authentication server. The state is stored in the IoT device's NVM and changed every time authentication occurs.

Tun and Mambo proposed a PUF-based secure AKE protocol for IoT devices [12]. In this protocol, the IoT devices' IDs are not anonymized but are in plaintext.

III. Proposed System

This section provides a detailed description of the proposed system. The notation used in this paper is shown in Table I.

TABLE I. Notation and Description

Notation	Description
$H(x)$	Apply hash function H to x
$a \leftarrow b$	Assign b to a
$a \parallel b$	Bitwise concatenate a and b
$\lfloor x \rfloor$	Round down x to the nearest integer
`UNIX TIME`	Current UNIX time

A. System Overview

The proposed system assumes an environment where many IoT devices are connected to an authentication server. Figure 1 shows the structural overview of the process. It denotes the authentication server possesses two types of processes: generation of temporary ID and identification by temporary ID. Figure 2 shows an example of the authentication server's database.

The authentication server stores the original ID, three types of temporary IDs, and data, such as the authentication information, in its database. When an IoT device requests authentication, it sends its temporary ID to the authentication server. The temporary ID is generated based on the IoT device original ID and the current UNIX time as follows:

$$\mathtt{id}_T \leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME}}{x} \right\rfloor\right) \tag{1}$$

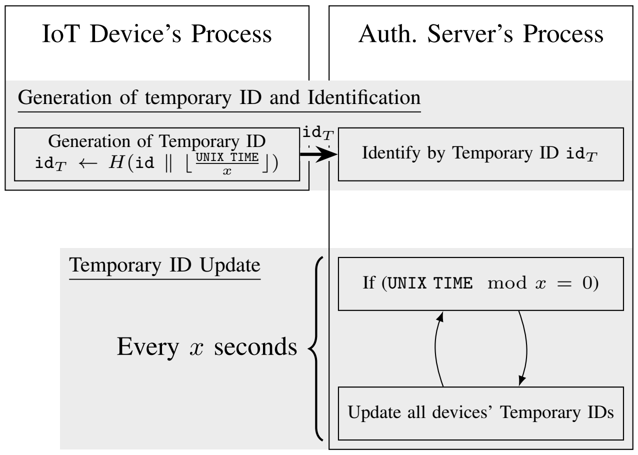

IoT Device's Process
Auth. Server's Process
Generation of temporary ID and Identification
Generation of Temporary ID
$\mathtt{id}_T \leftarrow H(\mathtt{id} \parallel \lfloor \frac{\mathtt{UNIX\ TIME}}{x} \rfloor)$
$\mathtt{id}_T$
Identify by Temporary ID $\mathtt{id}_T$
Temporary ID Update
Every x seconds
If ($\mathtt{UNIX\ TIME} \mod x = 0$)
Update all devices' Temporary IDs

Figure 1. Structural overview of the process.

id	$\mathtt{id}_{T1}$	$\mathtt{id}_{T2}$	$\mathtt{id}_{T3}$	data
0x82...	0x21...	0x70...	0x4F...	0x93...
0xEA...	0x45...	0x82...	0x18...	0xA7...
		⋮		
0xD6...	0x89...	0xA3...	0x28...	0xC1...

Figure 2. Authentication server database example.

where $\mathtt{id}$ is the IoT device original ID, $\mathtt{id}_T$ is the temporary ID, and x is the ID update interval constant shared between the IoT device and the authentication server. UNIX time is a system for tracking time, defined as the number of seconds that have elapsed since the Unix epoch, which is 00:00:00 UTC on 1 January 1970. The temporary ID $\mathtt{id}_T$ is changed every x seconds. This mechanism follows a principle similar to that of the Time-based One-Time Password (TOTP) algorithm [13]. Assuming the current UNIX time is synchronized between the IoT device and the authentication server. The condition on the value x and consideration of the communication and update process delay are described in Section III-D.

B. Initialization Phase

The IoT device original ID and ID update interval constant x are shared in advance. The authentication server generates three types of temporary IDs in the initialization phase and saves its database:

$$\begin{aligned} \mathtt{id}_{T1} &\leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME} - x}{x} \right\rfloor\right) && \text{Previous time-step,} \\ \mathtt{id}_{T2} &\leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME}}{x} \right\rfloor\right) && \text{Current time-step,} \\ \mathtt{id}_{T3} &\leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME} + x}{x} \right\rfloor\right) && \text{Next time-step.} \end{aligned} \tag{2}$$

C. Temporary IDs Update Phase

The authentication server updates the temporary IDs in its database every x seconds. The temporary IDs are updated as follows:

$$\begin{aligned} &\mathtt{id}_{T1} \leftarrow \mathtt{id}_{T2}, \qquad \mathtt{id}_{T2} \leftarrow \mathtt{id}_{T3}, \\ &\mathtt{id}_{T3} \leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME} + x}{x} \right\rfloor\right). \end{aligned} \tag{3}$$

To execute this operation every x seconds, the condition

$$\mathtt{UNIX\ TIME} \mod x = 0 \tag{4}$$

is used. To enhance the performance of the authentication server, the previous time-step and current time-step temporary IDs are substituted with the current time-step and the next time-step temporary ID, respectively, instead of generating new temporary IDs as in Equation (2).

D. Communication and Update Process Delay

Communication delay and the temporary ID update process delay on the authentication server should be considered.

The communication delay is the time it takes for the IoT device to send its temporary ID to the authentication server and for the authentication server to process it. The update delay is the time it takes for the authentication server to update its temporary IDs in its database.

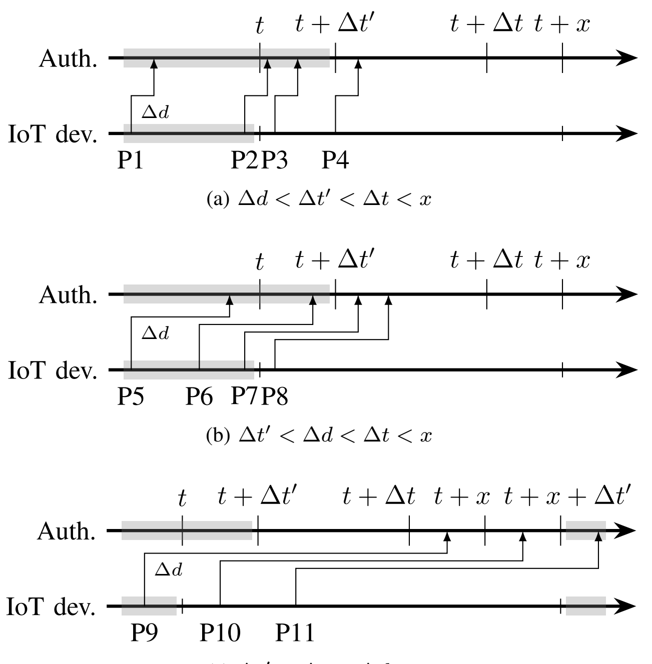
In the following, the effectiveness of the proposed system in addressing these delays and the condition of ID update constant x are discussed. The notation of the communication and update process delays is defined in Table II.

TABLE II. NOTATION OF COMMUNICATION AND UPDATE DELAY

Notation	Description
Δd	The communication delay. The IoT device sends $\mathtt{id}_T$ to the authentication server, and it takes Δd seconds to reach the authentication server.
Δt	Time required to update all temporary IDs on the authentication server.
$\Delta t'$	Time required to update a certain temporary ID on the authentication server. Assuming that the authentication server has 10,000 records on its database, the 5,000th record could be updated approximately $\Delta t' = \frac{1}{2}\Delta t$ seconds later.

Three cases are considered based on the relationship between Δd, $\Delta t'$, and Δt. Figure 3 shows the three cases. In (a), the update delay is larger than the communication delay. In (b), the communication delay is larger than the update delay. In (c), the communication delay is larger than the update delay and update time of all temporary IDs. The gray area represents the time when the temporary ID is updated. This process assumes that, for a certain ID $\mathtt{id}$, the authentication server takes $\Delta t'$ seconds to update the temporary ID $\mathtt{id}_T$ in its database.

Consider the case where the time required to update a certain temporary ID exceeds the communication delay (see Figure 3 (a)). Regarding points P1, P2, and P4, $\mathtt{id}_T$ sent from the IoT devices will match the current time-step temporary ID $\mathtt{id}_{T2}$ in the authentication server's database. Regarding point P3, the IoT device sends $\mathtt{id}_T$ using the current UNIX time, but the authentication server has not updated its temporary ID $\mathtt{id}_{T2}$ yet. Thus, $\mathtt{id}_T$ sent from the IoT device matches



(a) $\Delta d < \Delta t' < \Delta t < x$

(b) $\Delta t' < \Delta d < \Delta t < x$

(c) $\Delta t' < \Delta t < \Delta d < x$

Figure 3. Update and communication delay.

the next time-step temporary ID $\mathtt{id}_{T1}$ in the authentication server's database.

Consider the case where the communication delay exceeds the time required to update a certain temporary ID. The time required to update all temporary IDs exceeds the communication delay (see Figure 3 (b)). Regarding points P5, P6, and P8, $\mathtt{id}_T$ sent from the IoT devices will match the current time-step temporary ID $\mathtt{id}_{T2}$ in the authentication server's database. Regarding point P7, the IoT device sends $\mathtt{id}_T$, but the authentication receives it after the update process is completed due to the communication delay. Thus, $\mathtt{id}_T$ sent from the IoT device matches the next time-step temporary ID $\mathtt{id}_{T3}$ in the authentication server's database.

Finally, consider the case where the communication delay exceeds the time required to update a certain temporary ID (see Figure 3 (c)). Regarding point P10, $\mathtt{id}_T$ sent from the IoT devices will match the current time-step temporary ID $\mathtt{id}_{T2}$ in the authentication server's database. Regarding points P9 and P11, the authentication server has not updated the IoT device temporary ID. Thus, $\mathtt{id}_T$ sent from the IoT device matches the previous next-step temporary ID $\mathtt{id}_{T3}$ in the authentication server's database.

Consequently, the authentication server can identify the IoT device by matching the temporary ID sent from the IoT device with the temporary ID in its database.

Concerning all cases, the ID update interval constant x should be set to a value larger than the communication delay Δd and the time required to update all temporary IDs Δt.

IV. EVALUATION

This section evaluates the computational and communication overhead of the proposed system and privacy level in comparison to three traditional ID management systems, incorporating authentication schemes commonly used in IoT systems. The comparison highlights the efficiency of the proposed method in terms of lightweight operations and minimal data exchange. Table IV summarizes the computational, communication costs and privacy level of the proposed system and three baseline as follows:

- **Baseline 1**: The IoT device sends its ID in plaintext.
- **Baseline 2**: The IoT device sends its ID in hashed form.
- **Baseline 3**: Assuming that the IoT device derives a masked ID using hash function from its original ID and stored mask value in its NVM. The mask value (32 bytes) is generated by the authentication server and sent to the IoT device every time authentication occurs in plaintext.

This process assumes that the hash function is a cryptographic hash function Secure Hash Algorithm 256-bit (SHA-256), which produces a 32 bytes output.

Table III defines the privacy levels regarding the ID management systems. The privacy level is defined based on the ability to track the IoT device over time and the reuse of the ID.

TABLE III. PRIVACY LEVEL DEFINITION

Privacy Level	Description
None	The IoT device ID is sent in plaintext, allowing easy identification of the device.
Weak	The IoT device ID is not disclosed, but the ID is reused for multiple authentication sessions, making it possible to track the device over time.
Strong	The IoT device ID is changed every time authentication occurs or periodically, and the ID is not reused, making it difficult to track the device over time. The ID is masked with a secret value, enhancing privacy protection.

TABLE IV. COMPARISON OF COMPUTATIONAL AND COMMUNICATION COSTS, AND PRIVACY LEVELS

System	Computational Cost		Communication Cost	Privacy Level
	IoT device	Auth. Server		
Proposed	1 hash, R `UNIX TIME`	n hash every x seconds, ≤ 3 comparison	1 message (32 bytes)	Strong
Baseline 1	No cost	1 comparison	1 message (32 bytes)	None
Baseline 2	1 hash	1 hash, 1 comparison	1 message (32 bytes)	Weak
Baseline 3	1 hash, receive mask value, R/W NVM	1 hash, generates mask value	2 messages (64 bytes)	Strong

Abbreviations:
n is the number of IoT devices connected to the authentication server.
R/W is read and write, respectively.

As shown in Table IV, the proposed system offers the lowest computation and communication costs by utilizing a single hash function and unidirectional message transmission on the IoT device side. This design is highly suitable for resource constrained IoT environments. Although Baseline 1 incurs no computational cost and Baseline 2 has a similar cost to the proposed system on the IoT device side, their privacy protections can be described as none and weak, respectively, due to the reuse of static identifiers. On the other hand, Baseline 3 provides enhanced privacy by masking the ID, which can be described as strong, but incurs additional costs due to the need for generating and transmitting mask values.

Consequently, the proposed system achieves a balance between privacy protection and resource efficiency, making it a compelling choice for IoT applications where both computational and communication resources are limited.

V. Discussion

This section discusses the limitations of the proposed system and these potential solutions.

A. RTC Clock Drift

The proposed system uses UNIX time to generate temporary IDs and assumes that the IoT device and the authentication server have synchronized UNIX time. Most computers adopt the Real-Time Clock (RTC) to keep track of time. However, the RTC is not always accurate [14], and the clock drift–the offset between the actual time and the time kept by the RTC drift–can affect the correctness of generation of temporary ID.

Two approaches can be considered to address the clock drift issue. First, the IoT device can periodically synchronize its RTC with the authentication server's time using a time synchronization protocol, such as NTP (Network Time Protocol). In NTP, the IoT device and the authentication server communicate to NTP servers to synchronize their clocks. This approach helps maintain the accuracy of the IoT device's and the authentication server's RTC clocks. However, there are some concerns regarding the overhead and security of NTP. In terms of overhead, NTP imposes the need for IoT devices to perform write operations to the RTC registers in order to update the time values, in addition to incurring the communication overhead associated with the protocol. In terms of security, NTP does not ensure the authenticity of the time source, which may lead to the injection of falsified time information. According to Martin et al. [15], authentication in the context of NTP does not imply that the time is correct. Secure NTP [16] is a protocol that provides authentication and integrity protection for NTP messages, but it requires additional complexity and overhead for digital signatures and certificates.

Second, measure the offset between the IoT device's RTC and the authentication server's time denoted as Δp, and adjust the generation of temporary ID accordingly. The abstract of generation of temporary ID considering the clock drift is as follows:

1) Measure the round trip time (RTT) between the IoT device and the authentication server and obtain the average RTT denoted as $\overline{R}$.
2) The IoT device sends its RTC clock to the authentication server.
3) The authentication server calculates the offset between its RTC clock as follows:
$$\Delta p \leftarrow \text{IoT device's RTC} - \text{Auth. Server's RTC} - \frac{\overline{R}}{2}. \tag{5}$$
If $\Delta p > 0$, the IoT device's RTC is ahead of the authentication server's RTC, and if $\Delta p < 0$, the IoT device's RTC is behind the authentication server's RTC.
4) The authentication server stores the Δp value associated with the IoT device's ID.
5) The authentication server generates and updates the temporary ID as follows:
$$\begin{aligned} &\mathtt{id}_{T1} \leftarrow \mathtt{id}_{T2}, \qquad \mathtt{id}_{T2} \leftarrow \mathtt{id}_{T3}, \\ &\mathtt{id}_{T3} \leftarrow H\left(\mathtt{id} \parallel \left\lfloor \frac{\mathtt{UNIX\ TIME} + x + \Delta p}{x} \right\rfloor\right). \end{aligned} \tag{6}$$

This algorithm is executed every certain time interval. This approach only stores the offset of the time drift, eliminating the need for time synchronization via external servers such as NTP server. However, since the integrity of the RTC values transmitted by the IoT device cannot be guaranteed, it is necessary to incorporate mechanisms to ensure the integrity of the time information. The authentication server also requires the additional storage of the offset value Δp for each IoT device.

Both approaches incur additional communication and processing overhead, resulting in increased energy consumption of IoT devices. This overhead depends on the frequency of RTC synchronization. Therefore, it is necessary to examine whether this overhead can be reduced in comparison with the Baseline 3 presented in Section IV, which relies on read and write operations to NVM.

B. RTC Energy Consumption

There is a concern that the energy consumption of the RTC. The RTC consumes energy to keep track of the time. According to Nisshinbo Micro Devices Inc.[17], the RTC (C2051S01) is active for 10 years with a 3V CR2032 coin cell battery. RTC is designed to consume low power, and its energy consumption is negligible compared to the IoT device's energy consumption. However, it is necessary to measure the RTC's energy consumption and compare it with that of reading and writing to NVM, such as flash memory, in order to store temporary IDs.

C. Scalability of the Authentication Server

The authentication server should be able to process the generation of temporary ID and identification by temporary ID. In the proposed system, the authentication server generates one next time-step temporary ID and two substitution operations every x seconds for each device connected to the authentication server. Assuming that 100,000 IoT devices are connected to the authentication server, the authentication server needs to generate 100,000 next time-step temporary IDs and perform 200,000 substitution operations every x seconds. It is predicted that this protocol requires the authentication server to have sufficient processing capacity to handle these operations efficiently.

VI. Conclusion and Future Work

To address privacy concerns in IoT environments, this paper proposes an ID management system that leverages UNIX time to generate temporary device identifiers. In conventional systems, device IDs are often transmitted either in plaintext or in hashed form. This allows adversaries to monitor communication patterns and infer usage behavior over time, thereby introducing significant privacy risks. Moreover, session-based ID management schemes typically require frequent updates to NVM, resulting in increased energy consumption and reduced memory lifespan due to repetitive write operations.

In the proposed system, a temporary device ID is dynamically generated by computing a hash of the original device ID concatenated with the current UNIX timestamp. This temporary ID is updated every x seconds, where x is a shared constant known to both the IoT device and the authentication server. Since the identifier changes periodically, even successive communications from the same device appear to originate from different sources. This makes long-term tracking by eavesdroppers significantly more difficult. Furthermore, because ID updates are computed in memory without requiring writes to NVM, the scheme mitigates the energy and endurance issues inherent to session-based ID management systems. In the analysis conducted, the proposed system demonstrates lower computational and communication costs compared to traditional ID management systems, making it particularly suitable for resource-constrained IoT devices.

By integrating the system into the AKE protocol, the smart lock can be identified without revealing its original or static ID. Eavesdroppers cannot track the smart lock over time based on the temporary ID.

Future work will focus on two directions. First, the system will be implemented and evaluated on resource-constrained IoT devices (e.g., MCU, Micro Controller Unit) and authentication servers, measuring computational cost, energy consumption, and scalability under large device populations. These results will also inform the development of methods for determining the optimal update interval x, which is essential for balancing security and performance. Second, RTC clock drift will be investigated, comparing mitigation techniques to quantify their impact on the integrity and efficiency of the proposed approach.

Overall, these efforts will refine the system design and confirm its practicality for real-world IoT deployments, particularly in scenarios where energy efficiency and privacy-preserving authentication are critical.

Acknowledgments

This research would not have been possible without the valuable guidance and continuous support of Professor Akihiro Shimizu of Kochi University of Technology, Japan. The authors are deeply grateful for his insightful advice throughout the course of this work. This work was partially supported by JST K Program Grant Number JPMJKP24K3, Japan.

References

[1] Transforma Insights, “Current IoT Forecast Highlights,” Accessed: Jun. 25, 2025. [Online]. Available: https : / / transformainsights.com/research/forecast/highlights.

[2] A. Choudhary, “Internet of Things: a comprehensive overview, architectures, applications, simulation tools, challenges and future directions,” *Discover Internet of Things*, vol. 4, no. 1, p. 31, 2024.

[3] B. C. Lee, E. Ipek, O. Mutlu, and D. Burger, “Architecting phase change memory as a scalable dram alternative,” in proceedings of *the 36th annual international symposium on Computer architecture*, 2009, pp. 2–13.

[4] M. Kim, J. Lee, Y. Kim, and Y. H. Song, “An analysis of energy consumption under various memory mappings for FRAM-based IoT devices,” in proceedings of *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, 2018, pp. 574–579.

[5] S. Bennett and J. Sullivan, “NAND flash memory and its place in IoT,” in proceedings of *2021 32nd Irish Signals and Systems Conference (ISSC)*, 2021, pp. 1–6.

[6] J. Boukhobza, S. Rubini, R. Chen, and Z. Shao, “Emerging NVM: A survey on architectural integration and research challenges,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 23, no. 2, pp. 1–32, 2017.

[7] A. Braeken, “PUF based authentication protocol for IoT,” *Symmetry*, vol. 10, no. 8, p. 352, 2018.

[8] A. Badhib, S. Alshehri, and A. Cherif, “A robust device-to-device continuous authentication protocol for the internet of things,” *IEEE Access*, vol. 9, pp. 124 768–124 792, 2021.

[9] F. Chen, Z. Xiao, T. Xiang, J. Fan, and H.-L. Truong, “A full lifecycle authentication scheme for large-scale smart IoT applications,” *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 3, pp. 2221–2237, 2022.

[10] M. Alizadeh, M. H. Tadayon, and A. Jolfaei, “Secure ticket-based authentication method for IoT applications,” *Digital Communications and Networks*, vol. 9, no. 3, pp. 710–716, 2023.

[11] K. Nimmy, S. Sankaran, and K. Achuthan, “A novel lightweight PUF based authentication protocol for IoT without explicit CRPs in verifier database,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 5, pp. 6227–6242, 2023.

[12] N. W. Tun and M. Mambo, “Secure PUF-based authentication systems,” *Sensors*, vol. 24, no. 16, p. 5295, 2024.

[13] D. M’Raihi, S. Machani, M. Pei, and J. Rydell, *Rfc 6238: Totp: Time-based one-time password algorithm*, 2011.

[14] R. Moravskyi and Y. Levus, “Using Stream Processing for Real-Time Clock Drift Correction in Distributed Data Processing Systems,” in proceedings of *2024 IEEE 19th International Conference on Computer Science and Information Technologies (CSIT)*, 2024, pp. 1–4.

[15] J. Martin, J. Burbank, W. Kasch, and P. D. L. Mills, *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC 5905, Jun. 2010. DOI: 10.17487/RFC5905.

[16] D. F. Franke, D. Sibold, K. Teichel, M. Dansarie, and R. Sundblad, *Network Time Security for the Network Time Protocol*, RFC 8915, Sep. 2020. DOI: 10.17487/RFC8915.

[17] Nisshinbo Micro Devices Inc., “Real Time Clock (RTC): Introduction,” Accessed: Jun. 25, 2025. [Online]. Available: https://www.nisshinbo-microdevices.co.jp/en/products/real-time-clock/introduction/.