Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.068486

ARTICLE

Dynamic Session Key Allocation with Time-Indexed Ascon for Low-Latency Cloud-Edge-End Communication

Fang-Yie Leu[1], **Kun-Lin Tsai**[2,*], **Li-Woei Chen**[3], **Deng-Yao Yao**[2], **Jian-Fu Tsai**[2], **Ju-Wei Zhu**[2] **and Guo-Wei Wang**[2]

[1]Department of Computer Science and Information Engineering, Tunghai University, Taichung, 407224, Taiwan
[2]Department of Electrical Engineering, Tunghai University, Taichung, 407224, Taiwan
[3]Department of Computer and Information Sciences, R.O.C. Military Academy, Kaohsiung, 830208, Taiwan

*Corresponding Author: Kun-Lin Tsai. Email: kltsai@thu.edu.tw

Received: 30 May 2025; Accepted: 21 July 2025; Published: 29 August 2025

ABSTRACT: With the rapid development of Cloud-Edge-End (CEE) computing, the demand for secure and lightweight communication protocols is increasingly critical, particularly for latency-sensitive applications such as smart manufacturing, healthcare, and real-time monitoring. While traditional cryptographic schemes offer robust protection, they often impose excessive computational and energy overhead, rendering them unsuitable for use in resource-constrained edge and end devices. To address these challenges, in this paper, we propose a novel lightweight encryption framework, namely Dynamic Session Key Allocation with Time-Indexed Ascon (DSKA-TIA). Built upon the NIST-endorsed Ascon algorithm, the DSKA-TIA introduces a time-indexed session key generation mechanism that derives unique, ephemeral keys for each communication round. The scheme supports bidirectional key separation to isolate uplink and downlink data, thereby minimizing the risk of key reuse and compromise. Additionally, mutual authentication is integrated through nonce-based validation and one-time token exchanges, ensuring entity legitimacy and protection against impersonation and replay attacks. We validate the performance of DSKA-TIA through implementation on a resource-constrained microcontroller platform. Results show that our scheme achieves significantly lower latency and computational cost compared to baseline schemes such as AES and standard Ascon. Security analysis demonstrates high entropy in key generation, resistance to brute-force and replay attacks, and robustness against eavesdropping and key compromise. The protocol also exhibits resilience to quantum computing threats by relying on symmetric encryption principles and randomized key selection. Given its efficiency, scalability, and temporal security enhancements, DSKA-TIA is well-suited for real-time, secure communication in heterogeneous CEE environments. Future work will explore post-quantum extensions and deployment in domains such as smart agriculture and edge-based healthcare.

KEYWORDS: Lightweight cryptography; ascon algorithm; cloud-edge-end security; time-indexed encryption; low-latency communication

1 Introduction

The Cloud-Edge-End (CEE) represents a transformative architecture that distributes intelligence and services across cloud servers, edge nodes, and end devices. This layered model enables seamless integration of heterogeneous physical entities, such as sensors, actuators, vehicles, smart appliances, and industrial equipment, with digital infrastructures through embedded systems, software stacks, and heterogeneous communication protocols. Different from monolithic cloud computing models, the CEE architecture supports localized data processing and low-latency decision-making to facilitate real-time data exchange, adaptive control, and situational awareness in large-scale, distributed environments [1–3].

Applications in the CEE environment are now widely employed in various domains, including smart healthcare [4], precision agriculture, intelligent transportation systems [5], environmental monitoring, and industrial automation [6]. As CEE deployments continue to increase in both density and complexity, the demand for responsive, energy-efficient, and secure communication protocols grows simultaneously, especially with latency-sensitive and mission-critical requirements. In practice, data transmission must not only meet timing constraints but also ensure confidentiality, integrity, and freshness.

However, securing data within the CEE architecture has significant challenges, since many edge and end devices are resource-constrained, i.e., with limited computational capabilities, memory, and energy. Although modern communication technologies, such as LPWAN, NB-IoT, Wi-Fi 6, and 5G, provide high-density connectivity and high bandwidth efficiency [7], the end-to-end security needs to be further improved [8]. Conventional cryptographic schemes, such as AES and RSA, while robust, impose computational and energy loads that should be avoided for lightweight devices [9]. Consequently, the research community has focused on lightweight cryptographic primitives for resource-constrained environments, including NIST-endorsed schemes, such as Ascon [10], PRESENT [11] and LEA.

On the other hand, several security vulnerabilities of these schemes remain unresolved. First, many lightweight schemes rely on static or pre-shared keys, which, once compromised, expose both historical and future communication contexts. Also, the absence of forward secrecy is especially problematic in dynamic and mobile CEE scenarios [12]. Second, the lack of temporal binding mechanisms makes such systems vulnerable to replay attacks. Third, the use of symmetric keys without bidirectional separation exposes communication messages when the system is compromised. These issues are particularly critical in latency-sensitive applications, such as autonomous vehicle coordination, remote surgery, and critical infrastructure surveillance.

To overcome these limitations, in this paper, we propose a novel lightweight encryption framework, named Dynamic Session Key Allocation with Time-Indexed Ascon (DSKA-TIA), which, built upon a lightweight authenticated encryption algorithm, i.e., Ascon, introduces a dynamic key generation mechanism where session keys are derived from and indexed by system time, yielding temporary session keys for each communication session. In contrast to static schemes, the DSKA-TIA supports bidirectional key allocation, i.e., generating distinct keys for uplink and downlink channels to ensure channel independence, even in the case that the key employed is exposed. Furthermore, the DSKA-TIA integrates a mutual authentication protocol based on time-synchronized nonces and one-time tokens to prevent desynchronization, impersonation, and replay attacks. These mechanisms are specifically designed to minimize their processing overheads, making the solution suitable for real-time, low-power CEE environments.

The main contributions of this paper are summarized as follows:

(1) A novel time-indexed session key allocation scheme, which utilizes system timestamps within the Ascon cipher to enable ephemeral, context-specific key generation.
(2) Bidirectional key differentiation, ensuring separate session keys for uplink and downlink communication, aiming to improve resilience against key reuse and compromise.
(3) A lightweight mutual authentication protocol, incorporating time-based nonces and one-time tokens to prevent impersonation and replay attacks with less overhead than other schemes.
(4) Performance validation on resource-constrained devices, demonstrating low computational cost and suitability for real-time, energy-aware CEE applications.

(5) Security analysis, establishing resistance against common threats including replay, desynchronization, and compromised-key attacks.

To contextualize the proposed DSKA-TIA framework, we consider a representative application scenario in smart healthcare monitoring, which exemplifies the Cloud-Edge-End (CEE) architecture. In this setting, wearable medical sensors (end devices) continuously collect patient vital signs and transmit data to nearby edge gateways (e.g., smartphones or local servers). These gateways then verify session freshness and relay critical data to cloud platforms for diagnostic analysis and long-term storage. Within this architecture, DSKA-TIA offers strong session-level security by enabling time-indexed, lightweight mutual authentication and encryption. Its decentralized design supports intermittent connectivity and resource constraints—typical of real-world CEE deployments.

The remainder of this paper is organized as follows: Section 2 introduces related work of lightweight cryptographic techniques and their applicability to CEE infrastructures. Section 3 details the Ascon algorithm, the proposed DSKA-TIA framework, and its integration with session key scheduling. Security analysis and performance evaluation are respectively presented and discussed in Section 4. Section 5 concludes the study and outlines future research directions.

2 Related Studies

Securing data within resource-limited environments, such as end and edge nodes in CEE systems, has advanced significantly in lightweight cryptography. Dhanda et al. [13] conducted a comprehensive evaluation of 54 lightweight cryptographic primitives, including block and stream ciphers, hash functions, and elliptic curve–based schemes to analyze their trade-offs in chip area, power consumption, and security strength. Among asymmetric techniques, Goyal and Sahula [14] showed that Elliptic Curve Diffie-Hellman (ECDH) provides superior energy and area efficiencies compared to RSA and traditional Diffie-Hellman, meaning that the RCDH is highly suitable for embedded deployments within IoT and CEE systems.

Recent work has also focused on secure key management to support ephemeral communication in dynamic IoT environments. Ding et al. [15] proposed the TinyMT and XSadd based lightweight key synchronization (TXLKS) algorithm and Lightweight and Cryptographically secure Protocol based on TinyMT and XSadd (LCPT) protocol to synchronize session key and tolerate delays. Their protocol, formally verified using Tamarin and validated through NIST and TestU01 test suites, emphasizes both security correctness and lightweight responsiveness. Wu et al. [16] introduced a key-per-message strategy integrated with lightweight encryption and authentication mechanisms. Compared to IPSec, their design significantly reduces resource consumption while enhancing confidentiality. However, these schemes largely depend on static or unidirectional keying models, which limit their effectiveness in scenarios requiring forward secrecy and session-level isolation.

For applications requiring fast cryptographic processing, efficient hardware implementations are critical. Tran et al. [17] investigated the Ascon algorithm, finalist in the NIST Lightweight Cryptography competition, on Field-Programmable Gate Array (FPGA) platforms. They presented two design variants: an unrolled version for high-throughput environments and a recursive variant for area-constrained systems, to validate Ascon's applicability in AI-enabled edge scenarios. At the physical layer, Khalid et al. [18] demonstrated the use of Reconfigurable Intelligent Surfaces (RIS) in 6G-IoT environments, by employing beamforming and artificial noise injection to enhance energy-efficient physical-layer security. These approaches contribute to transmission-level protection, although they lack integrated key lifecycle management and context-aware authentication.

Efforts to enhance trust and data integrity have led to the integration of blockchain in IoT systems. Al-Nbhany et al. [19] reviewed blockchain-enabled architectures in healthcare IoT, and emphasized their potential for decentralized trust and immutability. However, the computational overhead of consensus mechanisms remains a major barrier when deploying resource-constrained edge nodes. To mitigate this, some studies advocate combining blockchain with lightweight encryption for device-level security. In parallel, bio-inspired cryptography, particularly DNA-based systems, has emerged as an unconventional yet powerful security approach. Ahamed and Murugan [20] studied DNA cryptosystems and discussed hybrid methods by combining ECC, HMAC, and chaos theory. Similarly, Ali and Anwer [21] further introduced a tri-layer scheme by integrating DNA, genetic algorithms, and ECC, but noted similar concerns regarding deployment feasibility.

The existing literature emphasizes the need for secure, scalable, and low-overhead solutions for CEE ecosystems. While many proposals provide strong security, they often suffer from limitations such as static key usage, high latency, or hardware overhead. In particular, few solutions support time-bound, bidirectional key isolation and mutual authentication within a single, lightweight framework. To bridge this gap, the proposed DSKA-TIA framework introduces a time-indexed session key allocation mechanism based on the Ascon cipher, offering cryptographically isolated communication with lower computational overhead than other security schemes. By combining authenticated encryption with dynamic key scheduling, the DSKA-TIA addresses key reuse, impersonation, and replay threats, while meeting with the real-time and energy-aware constraints inherent to Cloud-Edge-End architectures.

3 Ascon-Based Lightweight Encryption Method in a Cloud-Edge-End Context

This section introduces the cryptographic foundation of the proposed DSKA-TIA scheme and explains how it extends the standard Ascon encryption framework to include time-indexed session key generation, mutual authentication, and lightweight key selection. Section 3.1 reviews the core design of the Ascon algorithm. Section 3.2 highlights vulnerabilities in conventional Ascon-based encryption in IoT environments. Section 3.3 presents the DSKA-TIA mutual authentication and key generation process. Section 3.4 details the dynamic encryption/decryption flow incorporating time-indexed entropy and key diversification.

While the proposed DSKA-TIA protocol is evaluated under a generic client-server model, its design is inherently compatible with the Cloud-Edge-End (CEE) architecture, which consists of three distinct layers with specific roles and constraints:

- Cloud Layer: Serves as a centralized, trusted authority responsible for policy orchestration and key provisioning.
- Edge Layer: Includes gateways or fog nodes that mediate communication between cloud and end devices. These nodes operate under limited compute/storage capacity and require lightweight cryptographic support.
- End Layer: Comprises highly constrained and possibly mobile devices (e.g., sensors, wearables), which demand low-latency, low-energy operations and frequent re-authentication.

DSKA-TIA addresses these layered requirements by separating session keys for directional communication, using time-indexed validation to ensure data freshness, and minimizing memory and computation overhead through randomized key selection. Moreover, the protocol enables secure, decentralized authentication without relying on persistent cloud access—making it well-suited for real-time and intermittently connected environments typical of CEE systems.

3.1 Overview of the Ascon Algorithm

Ascon [10] is a lightweight authenticated encryption algorithm selected by NIST in 2023 as the standard for constrained environments, such as Internet of Things (IoT), RFID, and edge computing systems. The Ascon cipher family (e.g., Ascon-128, Ascon-128a, Ascon-80pq) is known for its simplicity, high efficiency, and resilience against cryptanalytic attacks. It combines encryption and authentication in a single process, offering both confidentiality and message integrity with low hardware requirement. The Ascon algorithm has four features, including

(1) Lightweight and Efficient: Ascon is designed to be fast and efficient on devices with limited computational power, memory, and energy, making it ideal for IoT and embedded systems.
(2) Security: Ascon provides both confidentiality and integrity to ensure that any tampering with the data during transmission is detectable.
(3) Simplicity: The algorithm is simple and easy to implement in software and hardware, aiming at reducing the likelihood of errors or vulnerabilities.
(4) Resilience: Ascon is resistant to a variety of cryptographic attacks, including differential and linear cryptanalysis. It has been considered to be secure for lightweight use cases by the cryptographic community.

The Ascon's parameters are defined by a key length greater than 160 bits, a block size of r bits (rate), and the number of internal rounds (assumed to be a and b rounds). The encryption algorithm (E) inputs a k-bit secret key K, a 128-bit nonce N, arbitrary-length associated data A and plaintext P, and then produces a ciphertext C and a 128-bit tag T.

The encryption operations of the Ascon-128 algorithm can be represented as Eq. (1).

$$\mathrm{AE.E}(K, N, A, P) = (C, T) \tag{1}$$

The encryption process of the Ascon-128 algorithm is illustrated in Fig. 1.

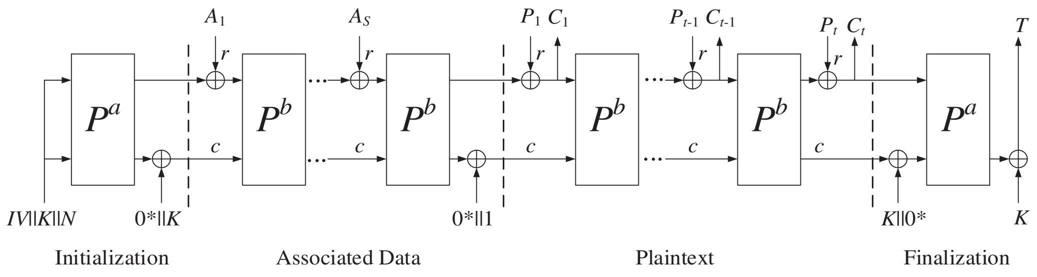


Figure 1: Encryption process of the Ascon-128 algorithm [10]

The decryption process inputs K, N, A, C, and T and then outputs the verification result which is P, if the verification is successful. Otherwise, Z as an incorrect result is outputted. The decryption operations of the Ascon-128 algorithm is expressed as Eq. (2)

$$\mathrm{AE.D}(K, N, A, C, T) = (P, Z) \tag{2}$$

The decryption process of the Ascon-128 algorithm is illustrated in Fig. 2.

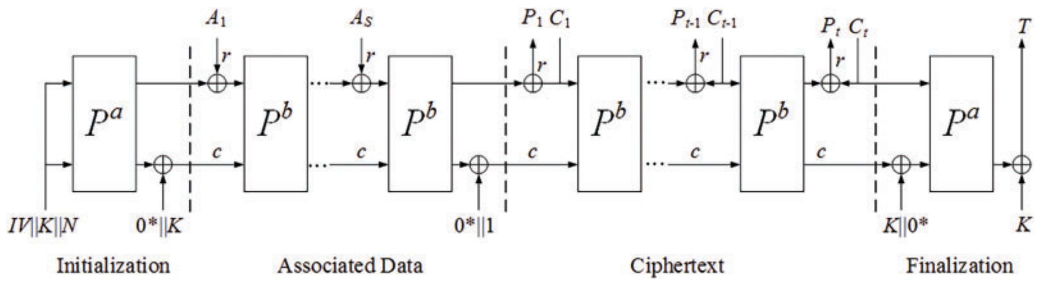


Figure 2: Decryption process of the Ascon-128 algorithm [10]

The design of Ascon-Hash utilizes an extensible output function X with parameters, including block size r bits, the number of internal rounds a and b, and an output length constraint h. X hashes input messages M of arbitrary length into an l-bit hash H, where $l \leq h$, and h is typically set to 256 bits. The hashing operations of the Ascon-Hash algorithm can be represented as Eq. (3).

$$X(M, l) = H \tag{3}$$

The operational flow of the Ascon-Hash algorithm is depicted in Fig. 3.

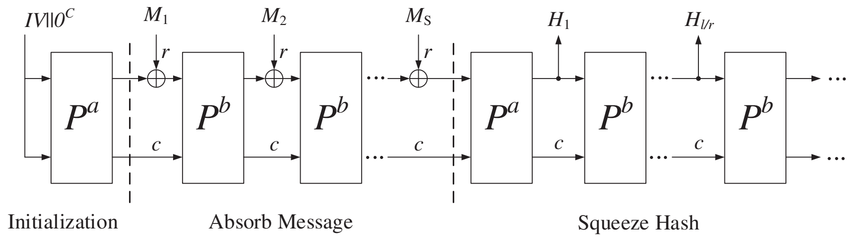


Figure 3: The operational flow of the Ascon-Hash algorithm [10]

While Ascon provides strong baseline security, its standard does not inherently protect against session-level threats, such as replay attacks, session hijacking, and key reuse. To address these gaps, we propose a set of enhancements that integrate time-awareness, mutual authentication, and session-specific key derivation into the Ascon framework.

3.2 Enhancing Ascon with Time-Indexed Security

Although the Ascon suite offers lightweight and authenticated encryption, its native design exhibits vulnerabilities in dynamic IoT communication scenarios. Specifically, Ascon lacks a built-in mechanism for invalidating previously used session parameters, rendering it susceptible to desynchronization and impersonation attacks. Furthermore, Ascon typically negotiates a single session key upon successful authentication. Such design cannot provide separate keys for each communication direction to isolate transmission paths, i.e., once this session key is compromised, the communication becomes vulnerable, exposing the system to man-in-the-middle and ciphertext analysis attacks.

To address the abovementioned vulnerabilities, we propose an improved protocol built on the existing Ascon framework. In our process, two algorithms from the Ascon suite are adopted: the Ascon-128 algorithm, responsible for authentication and the generation of secure parameters and keys, and the Ascon-Hash algorithm, which handles hashing and key verification. With the two algorithms, several security enhancements are interduced. First, we integrate a time-based parameter to strengthen temporal validity. Second, the length of certain key parameters is extended to increase cryptographic strength. Third, we augment the number of session keys to provide separate keys for bidirectional communication to ensure that even if one key is compromised, the other communication channel remains secure. Moreover, we employ hashing algorithms to safeguard the transmission of authentication information. The purpose is to reduce the chance that an attacker intercepts transmitted data from which to compromise the critical session keys. These modifications aim to mitigate potential security threats, including those related to desynchronization attacks and key compromise.

In our proposed method, the Server End (SE) and Device End (DE) databases store relevant data separately. SE stores $\{(IDS_{DE}^{0}, K_{DE}^{0}), (IDS_{DE}^{1}, K_{DE}^{1}), ID_{DE}\}$, where IDS_{DE}^{0} represents the DE's first anonymous ID, K_{DE}^{0} is the first encryption key, IDS_{DE}^{1} is the DE's second anonymous ID, K_{DE}^{1} is the second encryption key, and ID_{DE} is the DE's ID. DE stores not only $\{(IDS_{DE}^{0}, K_{DE}^{0}), (IDS_{DE}^{1}, K_{DE}^{1}), ID_{DE}\}$ but also $\{(IDS_{SE}, K_{SE}, ID_{SE})\}$, where IDS_{SE} represents the SE's anonymous ID, K_{SE} is the 128-bit encryption key, and ID_{SE} is the SE's ID.

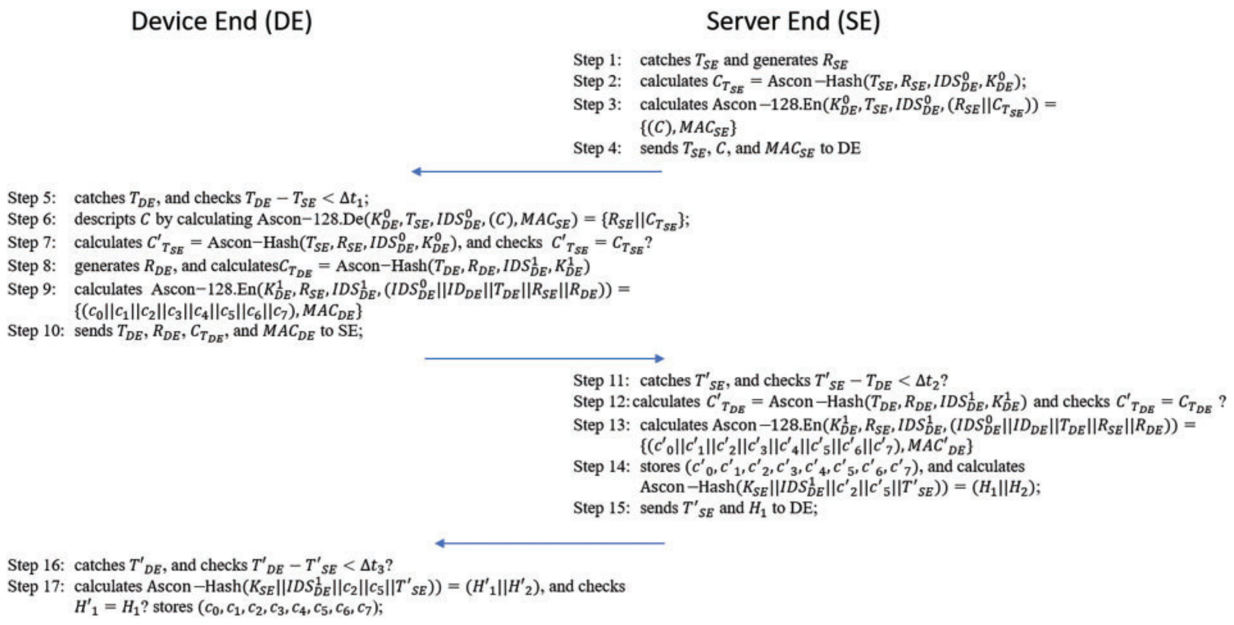
As DSKA-TIA introduces timestamp-based validation for session key generation and replay attack resistance, accurate time synchronization becomes a necessary system assumption. While lightweight encryption algorithms like Ascon do not rely on time, the proposed enhancement requires devices in the CEE architecture to maintain loosely synchronized clocks. In practical deployments, synchronization can be achieved through protocols such as the Network Time Protocol (NTP) for general-purpose networks, Precision Time Protocol (PTP) for high-accuracy systems, or IEEE 802.15.4e-TSCH in resource-constrained IoT mesh environments. However, CEE environments are often characterized by device mobility, variable latency, and intermittent connectivity, which can disrupt synchronization. To mitigate this, DSKA-TIA incorporates a configurable permissible time (Δt) during session validation (see Section 4.4), which tolerates bounded clock drift and short-term desynchronization. For dynamic or mobile systems, we recommend augmenting deployments with periodic synchronization beacons or local time correction mechanisms at the edge to maintain protocol integrity.

3.3 *Mutual Authentication and Secure Session Establishment*

The DSKA-TIA protocol begins with a mutual authentication process between the SE and DE. This process consists of three main stages:

(1) SE generates a random nonce (R_{SE}) and timestamp (T_{SE}), and then computes a validation tag $C_{t_{SE}}$ using Ascon-Hash.
(2) DE receives and verifies the timestamp and validation tag, and then generates its own response using a selected key-ID pair and a fresh nonce (R_{DE}) to produce $C_{t_{DE}}$.
(3) SE validates the response, checks the timestamp, and completes the handshake by verifying key consistency.

This process ensures that both entities authenticate each other and that session keys are agreed upon using non-replayable, time-bound parameters. The mutual authentication and key generation process is shown in Fig. 4. Multiple data keys can be generated at the same time during the mutual authentication process between SE and DE. The detailed authentication and key generation process is described as follows.



Step 1: catches T_{SE} and generates R_{SE}
Step 2: calculates $C_{T_{SE}}$ = Ascon−Hash$(T_{SE}, R_{SE}, IDS^0_{DE}, K^0_{DE})$;
Step 3: calculates Ascon−128.En$(K^0_{DE}, T_{SE}, IDS^0_{DE}, (R_{SE}||C_{T_{SE}}))$ = $\{(C), MAC_{SE}\}$
Step 4: sends T_{SE}, C, and MAC_{SE} to DE

Step 5: catches T_{DE}, and checks $T_{DE} - T_{SE} < \Delta t_1$;
Step 6: descripts C by calculating Ascon−128.De$(K^0_{DE}, T_{SE}, IDS^0_{DE}, (C), MAC_{SE}) = \{R_{SE}||C_{T_{SE}}\}$;
Step 7: calculates $C'_{T_{SE}}$ = Ascon−Hash$(T_{SE}, R_{SE}, IDS^0_{DE}, K^0_{DE})$, and checks $C'_{T_{SE}} = C_{T_{SE}}$?
Step 8: generates R_{DE}, and calculates$C_{T_{DE}}$ = Ascon−Hash$(T_{DE}, R_{DE}, IDS^1_{DE}, K^1_{DE})$
Step 9: calculates Ascon−128.En$(K^1_{DE}, R_{SE}, IDS^1_{DE}, (IDS^0_{DE}||ID_{DE}||T_{DE}||R_{SE}||R_{DE}))$ = $\{(c_0||c_1||c_2||c_3||c_4||c_5||c_6||c_7), MAC_{DE}\}$
Step 10: sends T_{DE}, R_{DE}, $C_{T_{DE}}$, and MAC_{DE} to SE;

Step 11: catches T'_{SE}, and checks $T'_{SE} - T_{DE} < \Delta t_2$?
Step 12: calculates $C'_{T_{DE}}$ = Ascon−Hash$(T_{DE}, R_{DE}, IDS^1_{DE}, K^1_{DE})$ and checks $C'_{T_{DE}} = C_{T_{DE}}$?
Step 13: calculates Ascon−128.En$(K^1_{DE}, R_{SE}, IDS^1_{DE}, (IDS^0_{DE}||ID_{DE}||T_{DE}||R_{SE}||R_{DE}))$ = $\{(c'_0||c'_1||c'_2||c'_3||c'_4||c'_5||c'_6||c'_7), MAC'_{DE}\}$
Step 14: stores $(c'_0, c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$, and calculates Ascon−Hash$(K_{SE}||IDS^1_{DE}||c'_2||c'_5||T'_{SE})) = (H_1||H_2)$;
Step 15: sends T'_{SE} and H_1 to DE;

Step 16: catches T'_{DE}, and checks $T'_{DE} - T'_{SE} < \Delta t_3$?
Step 17: calculates Ascon−Hash$(K_{SE}||IDS^1_{DE}||c_2||c_5||T'_{SE})) = (H'_1||H'_2)$, and checks $H'_1 = H_1$? stores $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$;

Figure 4: Mutual authentication and key generation process

Step 1: SE catches its system time T_{SE}, generates a 64-bit random number R_{SE};

Step 2: SE calculates $C_{T_{SE}}$ = Ascon-Hash $\left(T_{SE}, R_{SE}, IDS^0_{DE}, K^0_{DE}\right)$;

Step 3: SE encrypts R_{SE} by calculating Ascon – 128.En$(K^0_{DE}, T_{SE}, IDS^0_{DE}, (R_{SE}||C_{T_{SE}})) = \{(C), MAC_{SE}\}$, where (C) is the ciphertext, and MAC_{SE} is the message authentication code

Step 4: SE sends T_{SE}, C, and MAC_{SE} to DE;

Step 5: DE catches its system time T_{DE}, and checks whether $T_{DE} - T_{SE} < \Delta t_1$, where Δt_1 is the predefined time limit; once the check fails, cancel the key generation process, else it performs the next step;

Step 6: DE decrypts C by calculating Ascon – 128.De$(K^0_{DE}, T_{SE}, IDS^0_{DE}, (C), MAC_{SE}) = \{R_{SE}||C_{T_{SE}}\}$;

Step 7: DE calculates $C'_{T_{SE}}$ = Ascon-Hash $\left(T_{SE}, R_{SE}, IDS^0_{DE}, K^0_{DE}\right)$, and checks whether $C'_{T_{SE}} = C_{T_{SE}}$ or not; once the check fails, it cancels the key generation process. Otherwise, it performs the next step;

Step 8: DE generates a 64-bit random numbers R_{DE}, and calculates $C_{T_{DE}}$ = Ascon-Hash $\left(T_{DE}, R_{DE}, IDS^1_{DE}, K^1_{DE}\right)$;

Step 9: DE calculates Ascon – 128.En$(K^1_{DE}, R_{SE}, IDS^1_{DE}, (IDS^0_{DE}||ID_{DE}||T_{DE}||R_{SE}||R_{DE})) = \{(c_0||c_1||c_2||c_3||c_4||c_5||c_6||c_7), MAC_{DE}\}$, where Ascon – 128.En indicates the encryption process of Ascon-128, $(IDS_{DE}\,||ID_{DE}||\,T_{DE}\,||R_{SE}||\,R_{DE})$ is the plaintext, $(c_0||c_1||c_2||c_3||c_4||c_5||c_6||c_7)$ is the ciphertext, and MAC_{DE} is the message authentication code;

Step 10: DE sends T_{DE}, R_{DE}, $C_{T_{DE}}$, and MAC_{DE} to SE;

Step 11: SE catches its system time T'_{SE}, and checks whether $T'_{SE} - T_{DE} < \Delta t_2$, where Δt_2 is the predefined delivering time limit; once the check fails, it cancels the key generation process, else the process goes to the next step;

Step 12: SE calculates $C'_{T_{DE}}$ = Ascon-Hash $\left(T_{DE}, R_{DE}, IDS^1_{DE}, K^1_{DE}\right)$ and checks whether $C'_{T_{DE}} = C_{T_{DE}}$ or not; once the check fails, it terminates the key generation process, else the process performs the next step;

Step 13: SE calculates Ascon − 128.En$(K^1_{DE}, R_{SE}, IDS^1_{DE}, (IDS^0_{DE}||ID_{DE}||T_{DE}||R_{SE}||R_{DE}))$ = $\{(c'_0||c'_1||c'_2||c'_3||c'_4||c'_5||c'_6||c'_7), MAC'_{DE}\}$, and checks whether $MAC'_{DE} = MAC_{DE}$ or not; once the check fails, it stops the key generation process, else the process performs the next step;

Step 14: SE stores $(c'_0, c'_1, c'_2, c'_3, c'_4, c'_5, c'_6, c'_7)$, and calculates Ascon-Hash$(K_{SE}||IDS^1_{DE}||c'_2||c'_5||T'_{SE})) = (H_1||H_2)$;

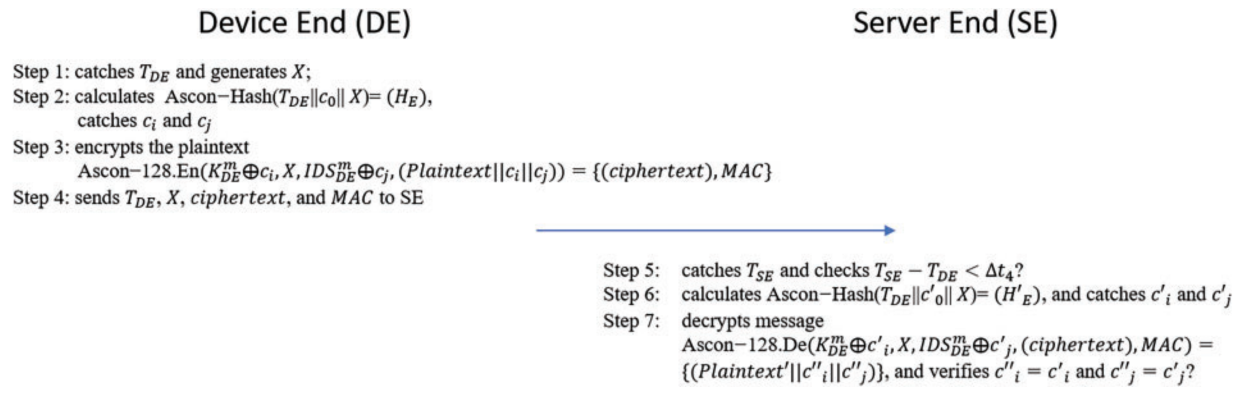
Step 15: SE sends T'_{SE} and H_1 to DE;

Step 16: DE catches its system time T'_{DE}, and checks whether $T'_{DE} - T'_{SE} < \Delta t_3$, where Δt_3 is the predefined time limit; once the check fails, it cancels the key generation process, else it performs the next step;

Step 17: DE calculates Ascon-Hash$(K_{SE}||IDS^1_{DE}||c_2||c_5||T'_{SE})) = (H'_1||H'_2)$, and checks whether $H'_1 = H_1$ or not, once the check fails, it terminates the key generation process, otherwise, it stores $(c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7)$;

3.4 Data Encryption Flow and Time Validation

Once mutual authentication is completed, the secure communication session proceeds using Ascon encryption. For each message, the sender selects an encryption key based on hash-derived entropy (e.g., H_E) and timestamp. The payload is encrypted with Ascon-128 using a unique session key and authenticated with an integrated MAC tag. At the receiver's end, the message is accepted only if the timestamp satisfies the condition $|T_d - T_s| < \Delta t$, where Δt is a predefined time limit. This enforces strict freshness and prevents attackers from sending previously captured messages. This enables secure, low-latency, and time-aware communication suitable for edge-based IoT systems. The data encryption and decryption process is shown in Fig. 5. The detailed data encryption and decryption process is shown below.



Step 1: catches T_{DE} and generates X;
Step 2: calculates Ascon−Hash($T_{DE}||c_0||X$)= (H_E), catches c_i and c_j
Step 3: encrypts the plaintext Ascon−128.En($K^m_{DE} \oplus c_i, X, IDS^m_{DE} \oplus c_j, (Plaintext||c_i||c_j)) = \{(ciphertext), MAC\}$
Step 4: sends T_{DE}, X, $ciphertext$, and MAC to SE

Step 5: catches T_{SE} and checks $T_{SE} - T_{DE} < \Delta t_4$?
Step 6: calculates Ascon−Hash($T_{DE}||c'_0||X$)= (H'_E), and catches c'_i and c'_j
Step 7: decrypts message Ascon−128.De($K^m_{DE} \oplus c'_i, X, IDS^m_{DE} \oplus c'_j, (ciphertext), MAC) = \{(Plaintext'||c''_i||c''_j)\}$, and verifies $c''_i = c'_i$ and $c''_j = c'_j$?

Figure 5: Mutual authentication and key generation process

Step 1: DE catches its system time T_{DE} and generates a random number X;

Step 2: DE calculates Ascon-Hash$(T_{DE}||c_0||X) = (H_E)$, and catches c_i and c_j according to the most significant three bits and the least significant three bits of H_E, e.g., the most significant three bits and the least significant three bits of H_E are, respectively, '010' and '111', then the DE catches c_2 and c_7;

Step 3: DE encrypts the plaintext by using Ascon-128.En$(K^m_{DE} \oplus c_i, X, IDS^m_{DE} \oplus c_j, (Plaintext\,||c_i|| c_j)) = \{(ciphertext)\,, MAC\}$, where m is the least significant three bit of H_E;

Step 4: DE sends T_{DE}, X, $ciphertext$, and MAC to SE;

Step 5: SE catches its system time T_{SE} and checks to see whether $T_{SE} - T_{DE} < \Delta t_4$, where Δt_4 is the predefined time limit; once the check fails, SE sends a warning message to DE and cancels the received message. Otherwise, it performs the next step;

Step 6: SE calculates Ascon-Hash$(T_{DE}||c'_0||X) = (H'_E)$, and catches c'_i and c'_j according to the most significant three bits and the least significant three bits of H'_E, and acquires K^m_{DE} and IDS^m_{DE} according to the least significant bit of H'_E;

Step 7: SE decrypts message Ascon-128.De$\left(K^m_{DE} \oplus c'_i, X, IDS^m_{DE} \oplus c'_j, (ciphertext), MAC\right) = \left\{\left(Plaintext' ||c''_i|| c''_j\right)\right\}$, and verifies whether $c''_i = c'_i$ and $c''_j = c'_j$ or not. If the verification is passed, SE sends an acknowledge message and deals the plaintext. Otherwise, SE sends a warning message to DE and drops the message received.

4 Security Analysis and Discussion

The proposed DSKA-TIA protocol assumes that all participating entities (e.g., sender, device, and gateway) within the Cloud-Edge-End (CEE) architecture maintain coarse-grained time synchronization, with drift bounded within the predefined permissible time limit (e.g., 5–50 ms). This synchronization can be achieved using lightweight NTP-like protocols or GPS-based timing in practical IoT deployments.

To validate the security and performance of the proposed DSKA-TIA scheme in a realistic IoT setting, all experiments were conducted on a Raspberry Pi 4 equipped with a quad-core ARM Cortex-A72 processor running at 1.5 GHz and 4 GB of RAM. The device operated under the Raspbian OS (Debian-based Linux), and all cryptographic modules were implemented in C and compiled using GCC v10.2.1. Time measurements were obtained using clock_gettime() to achieve nanosecond-level precision. This hardware-software configuration was selected to emulate resource-constrained environments typical of edge devices in IoT systems.

4.1 Randomness Analysis and Collision Resistance

To prevent key reuse and ciphertext predictability, the DSKA-TIA integrates time-indexed randomness into the session key generation process. Each session key K_s is derived as a function of a master key, a session nonce, and a timestamp. This construction ensures high entropy and unpredictability, as both timestamp and nonce are unique per session. In practical implementation, we use a 96-bit nonce with a system timestamp, resulting in 2^{128}-bit entropy coverage per session. Empirical randomness tests using NIST SP 800-22 show that the generated session keys pass standard tests for frequency, runs, and autocorrelation when validating their statistical unpredictability.

To evaluate the collision resistance of the DSKA-TIA's key scheduling, we consider the probability that two distinct sessions generate the same session key:

$$P(\text{collision}) \approx \frac{1}{2} \cdot \frac{n(n-1)}{2^k} \tag{4}$$

where n is the number of sessions and $k = 128$ is the effective output bit-length of the Ascon-based pseudorandom function. Assuming up to $n = 10^6$ sessions per device per day, the collision probability remains negligible:

$$P \approx \frac{10^{11}}{2^{128}} \ll 2^{-64} \tag{5}$$

This confirms that the DSKA-TIA provides strong protection against session key collisions and reuse-based cryptographic degradation.

4.2 Mutual Authentication Security

Mutual authentication is required in securing communication protocols, particularly in IoT environments where devices must verify each other's identity before exchanging sensitive information. In the DSKA-TIA, mutual authentication is achieved through a dual-key and dual-identity mechanism to ensure that both the SE and DE are legitimate in the communication session. The protocol utilizes a pair of identity-key tuples $(IDS_{DE}^{0}, K_{DE}^{0})$ and $(IDS_{DE}^{1}, K_{DE}^{1})$, which are dynamically selected during the session based on the hash-derived value H_E. Additionally, time-indexed verification tags $C_{T_{SE}}$ and $C_{T_{DE}}$ are generated using secure hash functions applied to session secrets. These tags serve as non-reusable authenticators embedded within the message flow and are validated during the key agreement phase.

The authentication handshake begins when SE transmits its identity IDS_{SE} and a time-bound cryptographic tag $C_{T_{SE}}$ to DE. In response, DE returns its dual identity-key tuple and validation token. Both sides then verify the hash integrity and consistency of the exchanged tokens before proceeding. This mutual challenge-response model ensures that both parties are cryptographically verified. If an attacker attempts to impersonate either SE or DE without having the correct (IDS, K) pairs, the computed hash tags will fail to match, and the protocol will immediately terminate. Moreover, in the authentication process, the binding of time-indexed session parameters can prevent the DSKA-TIA from pre-recorded responses, i.e., replay attacks. This multi-factor approach not only ensures identity legitimacy, but also enforces session-specific authentication to avoid key reuse or impersonation across different communication rounds. It also implicitly protects against man-in-the-middle attacks, as the adversary cannot forge both the identities and the time-indexed tags required for mutual verification. The authentication sequence is illustrated in Fig. 6, which outlines the flow of identity exchange and hash-based validation steps between SE and DE. Identity-key pairs and time-indexed hash tags are exchanged to ensure bidirectional entity verification. The process resists impersonation, key forgery, and man-in-the-middle attacks.

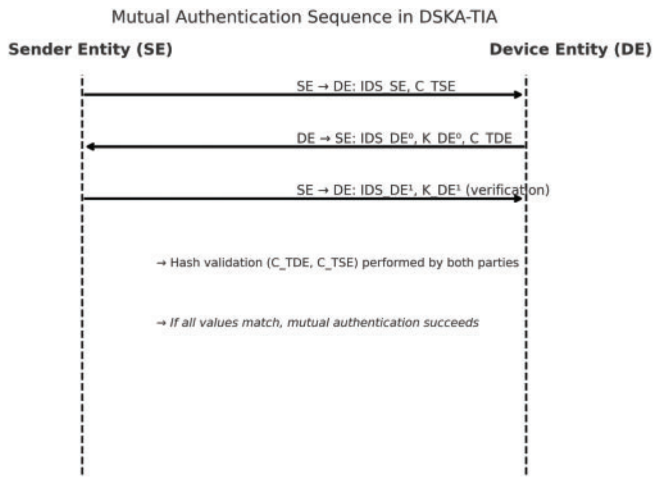

Mutual Authentication Sequence in DSKA-TIA
Sender Entity (SE)
Device Entity (DE)
SE → DE: IDS_SE, C_TSE
DE → SE: IDS_DE⁰, K_DE⁰, C_TDE
SE → DE: IDS_DE¹, K_DE¹ (verification)
→ Hash validation (C_TDE, C_TSE) performed by both parties
→ If all values match, mutual authentication succeeds

Figure 6: Mutual authentication process in DSKA-TIA

4.3 Message Integrity Protection

Message integrity ensures that transmitted data remains unaltered and verifiable throughout the communication process. In the DSKA-TIA framework, integrity protection is tightly coupled with the encryption mechanism through the use of message authentication codes (MACs) that are generated and verified using the Ascon authenticated encryption algorithm. These MACs are embedded within each encrypted packet and are evaluated by both the SE and DE. Specifically, parameters and other session-specific integrity tags are computed over the payload and key context using Ascon's built-in authentication. If any MAC fails, the decryption output is rejected. This early-stage filtering mechanism prevents the protocol from progressing to subsequent operations, such as session confirmation or key derivation. For example, during the mutual authentication and key exchange phases, MAC verification checkpoints are enforced at critical validation steps—namely Step 7 (initial token check), Step 12 (key confirmation), and Step 13 (bidirectional key hash verification). These checkpoints serve as protocol-level barriers to message forgery. Any modification, even a single-bit flip, results in a decryption failure and immediate protocol termination. This mechanism offers strong resistance to forgery, message injection, and ciphertext manipulation attacks. Since the MACs are session-dependent and derived from time-indexed keys and nonces, they also contribute to forward integrity to ensure that even if a key is compromised, previously authenticated messages retain detectable integrity and remain cryptographically verifiable.

4.4 Replay Attack Prevention via Time Indexing

A replay attack occurs when an adversary intercepts a legitimate message and resends it at a later time in an attempt to trick the system into accepting outdated data as valid. Such attacks are particularly effective against encryption schemes that lack temporal validation. By incorporating time-based parameters into the encryption process, the system can evaluate the freshness of received messages and thereby significantly reduce the success probability of replay attempts.

Let the following notations be defined:

- T_s: Timestamp generated by the sender;
- T_d: Timestamp recorded by the receiving device;
- Δt: Permissible time limit (e.g., 5 ms);
- P: Plaintext message;
- K: Symmetric key;
- $C = E_K(T_s, P)$: Ciphertext encrypted with a timestamp and key.

Upon receiving the ciphertext C, the device performs time verification using the condition:

$$|T_d - T_s| \leq \Delta t \tag{6}$$

If the message is replayed outside this acceptable time limit, the verification fails, and the message is rejected. For a replay attack to succeed, the adversary must ensure that the time difference $|T_d - T_s|$ falls within Δt. Assuming the attacker captures the legitimate ciphertext C and attempts to resend it at time T_a, where:

$$T_a \sim U(T_s + \varepsilon, T_s + T_{max}) \tag{7}$$

here, ε is the minimum replay delay and T_{max} is the maximum delay under the attacker's control, following a uniform distribution U. The probability of a successful replay attack is therefore given by:

$$P\left(\text{Replay Success}\right) = \frac{\Delta t}{T_{max} - T_s} \tag{8}$$

As $T_{max} \gg \Delta t$, the probability of success approaches zero. This demonstrates that the integration of time parameters into the encryption process can effectively mitigate replay attacks, especially when enforced with strict time-limit constraints. Fig. 7 illustrates the impact of increasing replay delay on the likelihood of success. As shown, even a short time limit combined with lightweight validation logic, it is sufficient to eliminate most replay-based exploits.

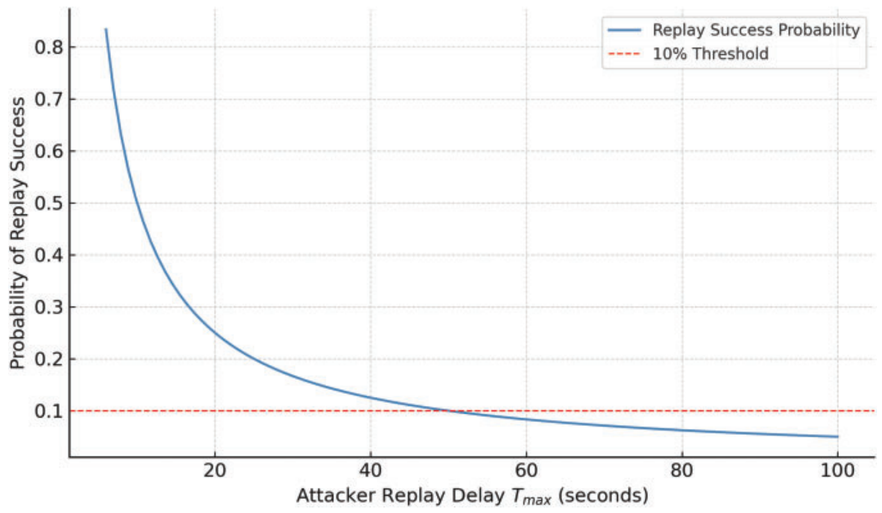


Figure 7: Probability of replay attack success as a function of attacker delay

In our implementation, we adopt a permissible time limit Δt = 5 ms to verify the freshness of timestamps during session key validation and replay attack prevention. This value was chosen based on latency benchmarks observed in typical edge-to-cloud IoT systems, where round-trip delays usually fall within 1–10 ms. A narrow time limit such as 5 ms balances two conflicting goals: minimizing the risk of replay attacks and tolerating minor network jitter or processing latency.

The DSKA-TIA protocol incorporates a clock deviation tolerance mechanism by design. Specifically, freshness is validated by computing the absolute difference between timestamps—$\Delta T = |T_{SE} - T_{DE}|$—and confirming whether it falls within the pre-defined time limit Δt. This method inherently allows for bounded time desynchronization between devices and ensures the system functions reliably even when perfect clock alignment is not achieved.

In DSKA-TIA, the timestamp T_{DE} is captured at session initiation and used in ciphertext derivation (via $T_{DE} \rightarrow$ c). It remains constant for the session's lifetime and is refreshed in each new session. Conversely, T_{SE} is recorded by the receiving node upon message arrival. The Δt check remains valid under three conditions: (1) device clock drift is bounded, (2) network transmission delay is low (as typical in CEE deployments), and (3) sessions are relatively short-lived so that T_{DE} remains temporally relevant.

This freshness time limit (Δt) also implicitly defines the session key's validity period. Since a session key is only accepted if $\Delta T \leq \Delta t$, any message using that key outside this time limit will be rejected. Thus, DSKA-TIA does not require explicit session timers—session key validity is automatically bounded by time-drift-aware freshness checks. This lightweight design simplifies key expiration handling and eliminates reliance on external timing or revocation mechanisms.

Should the message experience significant delay or desynchronization, ΔT will exceed Δt and the packet is safely discarded. We recommend tuning Δt based on the deployment environment—for instance, 1–5 ms for tightly synchronized edge-to-end networks, and 5–50 ms for edge-to-cloud scenarios where higher variance in latency and drift is expected.

4.5 Eavesdropping and Ciphertext Pattern Resistance

An eavesdropping attack involves a passive adversary who intercepts communication packets transmitted between legitimate parties intending to extract sensitive information, like cryptographic keys, session identifiers, or authentication tokens. Such attacks are particularly dangerous in IoT environments where data packets may send via unsecured or public networks. To mitigate this threat, the proposed DSKA-TIA framework incorporates a variety of cryptographic safeguards based on randomness and one-time parameterization. Specifically, the scheme utilizes non-reusable random values, including R_{SE}, R_{DE}, and session-derived parameter X, that are generated for each communication session. These values are incorporated into both encryption and authentication processes to ensure that intercepted ciphertexts are semantically independent, meaning that hackers cannot break the key of other sessions.

Furthermore, because these ephemeral parameters are neither reused nor derivable, without access to internal session state or secret keys, or breaking the underlying authenticated encryption scheme (Ascon), which provides IND-CPA and integrity guarantees, attackers are unable to perform pattern analysis or constructing replayable message sets to deduce the structure or content of legitimate messages. As a result, the DSKA-TIA exhibits strong resilience against eavesdropping attacks, i.e., able to eliminate the feasibility of passive traffic analysis or information leakage. Unlike other encryption schemes, which produce identical ciphertexts for the same plaintext under a fixed key, the proposed DSKA-TIA framework ensures semantic security through the use of per-session randomness. In fact, the DSKA-TIA incorporates dynamic, non-deterministic elements, including random nonces, timestamps, and time-indexed session keys, into each encryption and authentication process. Thus, even if the same message content is transmitted multiple times, the resulting ciphertexts will be distinct and unlinkable. This design choice meets modern security recommendations for resource-constrained devices operating in potentially adversarial network environments.

4.6 Key Compromise and Key Diversity Defense

The DSKA-TIA framework mitigates the compromised-key attacks through a multi-key encryption structure that dynamically diversifies the keyspace and increases the cryptographic cost of compromise by utilizing two sets of interrelated parameters: data encryption keys K_{DE}^{m} and identity tokens IDS_{DE}^{m}, the values of which are selected from a predefined key pool with 64 possible permutations to ensure that no single key dominates the encryption domain.

Furthermore, the values of K_{DE}^{m} and IDS_{DE}^{m} are selected from the least significant bit of the hash output H_E, where H_E is derived from current session context, timestamps, and random nonces. This selection entropy is high up to $2^7 = 128$ distinct key combinations, each representing a unique session configuration. Thus, even if an attacker obtains one or more static key components, without knowing the dynamic key selection logic, he/she cannot decrypt intercepted packets. The attacker's search complexity is $O(2^7)$ for key combination guessing per session. Since this design achieves per-session key isolation and forward security, key compromise in one session does not propagate to others. Therefore, the DSKA-TIA exhibits strong resistance against compromised-key attacks.

4.7 Entropy, Randomness, and Collision Resistance

To enhance the unpredictability of key selection, we incorporate a randomized key combination mechanism into the Ascon encryption process. This mechanism introduces additional entropy, thereby increasing the complexity of brute-force attacks. Let:

- K: the master key;
- $S = \{s_1, s_2, \cdots, s_n\}$: the set of possible randomly selected key combinations;
- $H(K)$: the entropy of the key selection process, quantifying the difficulty for an attacker to guess the correct key.

$H(K)$ is calculated using Shannon's entropy formula [22]:

$$H(K) = -\sum_1^n P(K_i)\log_2 P(K_i) \tag{9}$$

Assuming a uniform distribution over all n possible keys, the entropy simplifies to:

$$H(K) = \log_2(n) \tag{10}$$

For example, if the system allows 64 distinct key selection combinations, then:

$$H(K) = \log_2(64) = 6 \text{ bits} \tag{11}$$

This means that an adversary, without knowing of the specific key selection strategy, must additionally consider $2^6 = 64$ possible key combinations. This significantly increases the effort required for a successful brute-force attack by adding a layer of probabilistic resistance and reinforcing the cryptographic strength of the system. To further illustrate the impact of entropy on brute-force resistance, Fig. 8 shows the exponential growth in the number of key combinations as entropy increases. Even minor increases in entropy, enabled by randomized key selection, can significantly amplify the computational burden on adversaries, thereby enhancing resistance against brute-force attacks.

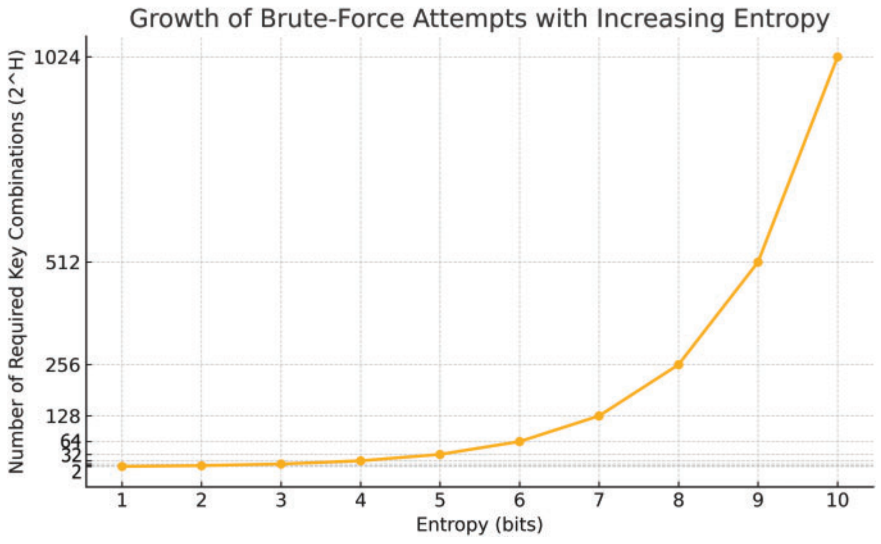


Figure 8: Relationship between entropy (in bits) and the number of required brute-force key attempts

4.8 Brute-Force and Quantum Attack Cost Estimation

Next, we analyze the computational cost when attackers attempt to break the Ascon-128 encryption enhanced with time-parameter validation. Assumptions:

The key length of Ascon-128 is 128 bits.

The attacker uses brute-force decryption at a rate of 10^{12} attempts per second.

Each decryption attempt must also verify a timestamp due to the embedded time parameter.

The total number of possible key combinations is:

$$2^{128} \approx 3.4 \times 10^{38} \tag{12}$$

Even if attackers use of a high-performance supercomputer capable of performing 10^{18} decryption attempts per second, the time required to exhaust the key space is:

$$\frac{2^{128}}{10^{18} \times 60 \times 60 \times 24 \times 365} \approx 10^{11} \text{ years} \tag{13}$$

If a quantum computer is employed, Grover's algorithm reduces the key search space to 2^{64}. The expected decryption time is:

$$\frac{2^{64}}{10^{18} \times 60 \times 60 \times 24 \times 365} \approx 500 \text{ years} \tag{14}$$

Now, we can know that the proposed encryption mechanism offers exceptionally strong resistance against classical and quantum brute-force attacks. Next, we summarize the estimated attack costs under various adversarial capabilities in Table 1, including classical brute-force attempts using both conventional and high-performance computing resources, as well as quantum attacks leveraging Grover's algorithm. Table 1 clearly demonstrates that the integration of Ascon encryption with time-indexed session key offers exceptional resilience against exhaustive search attacks.

Table 1: Estimated decryption time under classical and quantum attack scenarios, assuming brute-force search across the effective key space of the proposed encryption scheme

Attack scenario	Effective key space	Decryption rate	Estimated time to exhaust key space
Classical brute force (10^{12}/s)	$2^{128} \approx 3.4 \times 10^{38}$	10^{12}/s	$\approx 1.1 \times 10^{19}$ s $\left(\sim 3.4 \times 10^{11} \text{ years}\right)$
Classical brute force (10^{18}/s, supercomputer)	$2^{128} \approx 3.4 \times 10^{38}$	10^{18}/s	$\approx 1.1 \times 10^{13}$ s $\left(\sim 10^{11} \text{ years}\right)$
Quantum attack (Grover's Algorithm)	$2^{64} \approx 3.4 \times 10^{19}$	10^{18}/s	$\approx 1.8 \times 10^{9}$ s $\left(\sim 500 \text{ years}\right)$

The quantum attack estimate in Eq. (14) is based on Grover's algorithm, which reduces brute-force key search complexity from $O(2^n)$ to $O(2^{\frac{n}{2}})$. In this analysis, we optimistically assume a quantum computer capable of 10^{18} decryption attempts per second—on par with high-performance classical supercomputers. While current quantum hardware is far from achieving such speeds, this assumption represents a worst-case projection to conservatively assess the protocol's resilience under post-quantum conditions. Moreover, it is important to note that DSKA-TIA is not dependent on public-key cryptography and instead employs symmetric primitives, which are generally more resistant to quantum attacks. The integration of temporal validation via ΔT checking further narrows the effective time limit of opportunity for quantum attackers, adding another layer of post-quantum resilience.

4.9 Comparison with Existing Lightweight Ciphers

In this subsection, we compare the DSKA-TIA against several widely adopted lightweight encryption algorithms in terms of key length, computational complexity, and resistance to replay attacks. As shown in Table 2, traditional ciphers such as AES-128 and SIMON-128 offer strong cryptographic foundations, but lack built-in protection against temporal-based threats. PRESENT, though lightweight, provides a lower security level due to its shorter key length. Ascon-128, endorsed by NIST, offers a balance between performance and security, and when combined with time-indexed session validation, it further enhances replay resistance without significant computational overhead.

Table 2: Comparative analysis of lightweight encryption algorithms for IoT security

Algorithm	Key length (bits)	Security level	Encryption/Decryption speed (Cycles)	Replay attack resistant
AES-128	128	High	40–60	No
SIMON-128	128	Medium	30–40	No
PRESENT	80	Low	20–30	No
Ascon-128	128	High	12–15	Yes (with time parameter)
DSKA-TIA	128 + timestamp	High + Time-Bound Authentication*	12–15 + 1 validation	Yes

Note: *Time-Bound Authentication: Incorporates timestamps and temporal validation during session key generation to ensure freshness and replay attack resistance.

Table 2 depicts that the integration of time parameters into Ascon maintains its lightweight characteristics while introducing temporal resilience, meaning that the DSKA-TIA is suitable for real-time, secure communication in CEE IoT systems.

Fig. 9 presents a visual analysis of encryption/decryption efficiency vs. security level across five representative algorithms. The x-axis represents the average computational cost measured in cycles per byte, while the y-axis indicates the security level on an abstract scale ranging from low to high with temporal protection. Color coding distinguishes whether each algorithm inherently supports replay-attack protection. As shown, the DSKA-TIA achieves a unique position by maintaining low computational cost (≈14.5 cycles/byte) while providing enhanced security through time-indexed key generation and mutual authentication. This temporal dimension is particularly critical in latency-sensitive IoT applications. In contrast, traditional ciphers, such as AES-128 and SIMON-128, lack native replay protection and require more cycles per byte, which may be infeasible for constrained edge devices. Fig. 9 illustrates that DSKA-TIA achieves an effective balance between efficiency and security, thus suitable for real-time, resource-constrained IoT deployments within CEE architectures.

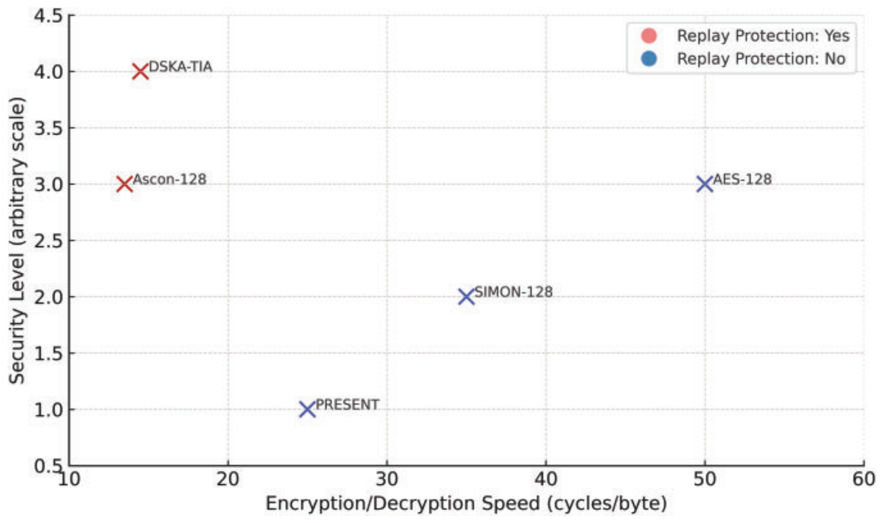


Figure 9: Comparison of lightweight encryption algorithms based on encryption/decryption speed (in cycles/byte) and security level. Color indicates whether the algorithm supports replay attack protection. The proposed DSKA-TIA approach offers both high efficiency and enhanced temporal security

4.10 Performance Comparison with Lightweight Ciphers

To assess the practical efficiency of the proposed DSKA-TIA scheme, we conducted a benchmark comparison against established lightweight symmetric encryption algorithms, including Ascon-128, PRESENT, and Lightweight AES. All experiments were carried out on a Raspberry Pi 4 device equipped with a 1.5 GHz quad-core ARM Cortex-A72 processor and 4 GB RAM, running Raspbian OS. All cryptographic implementations were written in C and compiled using GCC v10.2.1 with optimization level-O2. High-precision time measurements were obtained via clock_gettime() to ensure nanosecond accuracy.

Table 3 summarizes the latency, energy, and resource utilization characteristics for encryption and decryption operations. The DSKA-TIA scheme demonstrates lower latency than all compared ciphers, with an average encryption time of 35.2 μs and decryption time of 34.7 μs, outperforming Ascon-128 and PRESENT by approximately 15–30%. Additionally, DSKA-TIA achieves the lowest energy consumption per operation (2.8 μJ), confirming its suitability for deployment in energy-constrained edge environments.

Table 3: Performance comparison between DSKA-TIA and lightweight encryption schemes

Scheme	Encryption latency (μs)	Decryption latency (μs)	Energy per operation (μJ)	Code size (KB)	Memory usage (KB)
DSKA-TIA	35.2	34.7	2.8	12.4	19.3
Ascon-128	41.8	41.5	3.1	10.9	18.5
PRESENT	49.3	48.7	3.6	9.8	16.9
Lightweight AES	45.7	45.3	3.3	11.7	17.8

While the code size of DSKA-TIA is slightly larger than PRESENT, it remains within an acceptable range for constrained devices. Memory usage is also comparable, with DSKA-TIA requiring 19.3 KB, only marginally higher than other schemes. These results validate that the proposed scheme not only provides enhanced replay protection through time-indexed session keys but also retains a lightweight profile that meets the demands of real-world IoT deployments. These benchmarks reinforce that DSKA-TIA achieves strong performance efficiency while delivering enhanced temporal security, making it well-suited for scalable deployment in modern Cloud-Edge-End IoT architectures.

4.11 Scalability and Key Management Considerations

While bi-directional key separation enhances security by isolating send/receive channels, it may raise concerns about storage and management in large-scale networks. However, DSKA-TIA addresses this through lightweight, session-local key generation. Instead of pre-storing or distributing all keys in advance, each session dynamically derives its encryption and authentication keys based on hashed timestamps and exchanged random values.

This design ensures that only temporary, per-session keys are stored, significantly reducing the persistent memory footprint per device. Moreover, since key derivation is decentralized and based on device-local parameters (e.g., timestamps, nonces), DSKA-TIA avoids the need for global synchronization or a centralized key management infrastructure.

In deployments with thousands of nodes, key generation and validation operate in a peer-to-peer fashion, thereby mitigating scalability bottlenecks. We also note that the use of one-time session keys prevents key reuse, which further simplifies revocation and rotation mechanisms compared to traditional long-term key architectures.

5 Conclusion and Future Studies

In this paper, we proposed the DSKA-TIA, which is a lightweight encryption framework designed to meet the stringent security and latency requirements of resource-constrained devices in Cloud-Edge-End systems. The proposed system introduces several innovations: time-indexed key scheduling to enforce message freshness, bidirectional session key separation to preserve communication channel independence, and entropy-driven key selection to enhance unpredictability. This system achieves strong message integrity through embedded MAC validation and resists eavesdropping by ensuring per-session semantic security. Performance evaluations and entropy analyses show that the DSKA-TIA maintains low computational effort, making it well-suited for real-time applications in CEE environments. Compared with traditional lightweight ciphers, the DSKA-TIA offers superior protection with lower latency and is especially effective in environments requiring secure, time-sensitive data transmission.

In future work, we plan to further optimize the DSKA-TIA for ultra-low-latency applications by exploring pipeline-friendly implementations and cipher-round reductions. We would also like to extend the framework with quantum-resilient features by integrating post-quantum primitives, and to enhance its adaptability through ECC-based lightweight authentication. Additionally, deployments in domains such as smart agriculture, transportation, and edge healthcare will be conducted to evaluate its robustness and interoperability. Finally, we envision developing context-aware security adaptation and performance dashboards to support dynamic security management and visualization in heterogeneous CEE environments. These constitute our future studies.

Acknowledgement: The authors would like to thank Department of Electrical Engineering at Tunghai University for providing computational resources and test platforms. We also appreciate the valuable feedback from colleagues during internal discussions and pilot testing. Besides, the authors would like to thank the editors and anonymous reviewers for their insightful comments and suggestions, which helped improve the quality of this manuscript.

Funding Statement: The authors received no specific funding for this study.

Availability of Data and Materials: Not applicable.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Fang-Yie Leu, Li-Woei Chen, Kun-Lin Tsai; algorithm implementation: Fang-Yie Leu, Li-Woei Chen, Kun-Lin Tsai; experiment and simulation: Deng-Yao Yao, Jian-Fu Tsai, Ju-Wei Zhu, Guo-Wei Wang; analysis and interpretation of results: Fang-Yie Leu, Li-Woei Chen, Kun-Lin Tsai; draft manuscript preparation: Fang-Yie Leu, Li-Woei Chen, Kun-Lin Tsai, Deng-Yao Yao, Jian-Fu Tsai. All authors reviewed the results and approved the final version of the manuscript.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Diallo O, Rodrigues JJ, Sene M. Real-time data management on wireless sensor networks: a survey. J Netw Comput Appl. 2012;35(3):1013–21. doi:10.1016/j.jnca.2011.12.006.
2. López OLA, Rosabal OM, Ruiz-Guirola DE, Raghuwanshi P, Mikhaylov K, Lovén L, et al. Energy-sustainable IoT connectivity: vision, technological enablers, challenges, and future directions. IEEE Open J Commun Soc. 2023;4:2609–66. doi:10.1109/ojcoms.2023.3323832.
3. Zhu Z, Jiao T, Li Z. Innovative applications of IoT in smart home systems: enhancing environmental monitoring with integrated sensor technologies and MQTT protocol. J Wirel Mob Netw Ubiquitous Comput Dependable Appl. 2024;15(4):69–89.
4. Rocha A, Monteiro M, Mattos C, Dias M, Soares J, Magalhães R, et al. Edge AI for internet of medical things: a literature review. Comput Electr Eng. 2024;116(8):109202. doi:10.1016/j.compeleceng.2024.109202.
5. Kim H, Song HM. Lightweight IDS framework using word embeddings for in-vehicle network security. J Wirel Mob Netw Ubiquitous Comput Dependable Appl. 2024;15(2):1–13.
6. Nižetić S, Šolić P, Gonzalez-De DLDI, Patrono L. Internet of Things (IoT): opportunities, issues and challenges towards a smart and sustainable future. J Clean Prod. 2020;274(5):122877. doi:10.1016/j.jclepro.2020.122877.
7. Akpakwu GA, Silva BJ, Hancke GP, Abu-Mahfouz AM. A survey on 5G networks for the Internet of Things: communication technologies and challenges. IEEE Access. 2017;6:3619–47. doi:10.1109/access.2017.2779844.
8. Ansari AS, Altuwaijri G, Alodhyani F, El-Khalil Ghembaza MI, Paramb SMD, et al. A detailed review of current AI solutions for enhancing security in Internet of Things Applications. Comput Mater Contin. 2025;83(3):3713–52. doi:10.32604/cmc.2025.064027.
9. Centenaro M, Vangelista L, Zanella A, Zorzi M. Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios. IEEE Wirel Commun. 2016;23(5):60–7. doi:10.1109/mwc.2016.7721743.
10. Dobraunig C, Eichlseder M, Mendel F, Schläffer M. Ascon v1.2: lightweight authenticated encryption and hashing. J Cryptol. 2021;34(3):1–42. doi:10.1007/s00145-021-09398-9.
11. Sadhukhan R, Patranabis S, Ghoshal A, Mukhopadhyay D, Saraswat V, Ghosh S. An evaluation of lightweight block ciphers for resource-constrained applications: area, performance and security. J Hardw Syst Secur. 2017;1(3):203–18. doi:10.1007/s41635-017-0021-2.
12. Wang Y, Yang C, Lan S, Zhu L, Zhang Y. End-edge-cloud collaborative computing for deep learning: a comprehensive survey. IEEE Commun Surv Tut. 2024;26(4):2647–83. doi:10.1109/comst.2024.3393230.
13. Dhanda SS, Singh B, Jindal P. Lightweight cryptography: a solution to secure IoT. Wirel Pers Commun. 2020;112(3):1947–80. doi:10.1007/s11277-020-07134-3.
14. Goyal TK, Sahula V. Lightweight security algorithm for low power IoT devices. In: Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI); 2016 Sep 21–24; Jaipur, India. Piscataway, NJ, USA: IEEE; 2016. p. 1725–9.
15. Ding Z, He D, Qiao Q, Li X, Gao Y, Chan S, et al. A lightweight and secure communication protocol for the IoT environment. IEEE Trans Dependable Secur Comput. 2023;21(3):1050–67. doi:10.1109/tdsc.2023.3267979.

16. Wu XW, Yang EH, Wang J. Lightweight security protocols for the Internet of Things. In: Proceedings of the 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC); 2017 Oct 8–13. Montreal, QC, Canada. Piscataway, NJ, USA: IEEE; 2017. p. 1–7.
17. Tran SN, Hoang VT, Bui DH. A hardware architecture of NIST lightweight cryptography applied in IPSec to secure high-throughput low-latency IoT networks. IEEE Access. 2023;11:89240–8. doi:10.1109/access.2023.3306420.
18. Khalid W, Rehman MAU, Van Chien T, Kaleem Z, Lee H, Yu H. Reconfigurable intelligent surface for physical layer security in 6G-IoT: designs, issues, and advances. IEEE Internet Things J. 2023;11(2):3599–613. doi:10.1109/jiot.2023.3297241.
19. Al-Nbhany WA, Zahary AT, Al-Shargabi AA. Blockchain-IoT healthcare applications and trends: a review. IEEE Access. 2024;12(3):4178–212. doi:10.1109/access.2023.3349187.
20. Ahamed NN, Murugan T. Research study on DNA-based cryptosystem for cloud, IoT, and edge computing networks. In: Proceedings of the 2024 7th International Conference on Signal Processing and Information Security (ICSPIS); 2024 Nov 12–14; Dubai, United Arab Emirates. Piscataway, NJ, USA: IEEE; 2024. p. 1–6.
21. Ali S, Anwer F. Securing IoT data: a hybrid cryptographic approach. In: Proceedings of the 2024 11th International Conference on Computing for Sustainable Global Development (INDIACom); 2024 Feb 28–Mar 1; New Delhi, India. p. 1561–9.
22. Shannon CE. A mathematical theory of communication. Bell Syst Tech J. 1948;27(3):379–423.