International Journal of Safety and Security Engineering
Vol. 16, No. 1, January, 2026, pp. 169-176
Journal homepage: http://iieta.org/journals/ijsse

A Security Analysis and Lightweight Hardening of the Authentication Chains Protocol for the Internet of Things

Haewon Byeon

Department of Future Technology, Korea University of Technology and Education (KOREATECH), Cheonan-si 31253, Republic of Korea

Corresponding Author Email: bhwpuma@naver.com

https://doi.org/10.18280/ijsse.160114

Received: 8 November 2025
Revised: 13 January 2026
Accepted: 21 January 2026
Available online: 31 January 2026

Keywords:
Internet of Things, authentication chains, lightweight authentication, replay attack, rollback attack, timestamp freshness, pseudonymization, privacy-preserving authentication

ABSTRACT

The Internet of Things (IoT) relies on lightweight authentication protocols for resource-constrained devices. The Authentication Chains (AC) protocol, inspired by blockchain concepts, offers a chained-token mechanism for efficient and scalable device authentication. However, its security properties under practical deployment conditions, characterized by imperfect synchronization and potential device compromise, are not yet fully understood. This paper presents a comprehensive security analysis of the AC protocol and proposes lightweight countermeasures to mitigate identified vulnerabilities. We first formalize the protocol and a realistic attacker model encompassing replay, partial device exposure, and verifier state rollback capabilities. Through systematic analysis, we uncover critical weaknesses, including timestamp manipulation, replay-based impersonation, rollback-enabled chain reuse, and identity linkability. To address these flaws, we introduce an enhanced AC variant incorporating nonce-bound chaining, bounded timestamp freshness, identifier pseudonymization, and implicit revocation hints. We validate our findings and the proposed enhancements through a Python-based testbed comprising Raspberry Pi 4B and ESP32 devices communicating over Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP). Experimental results show that the enhanced protocol improves replay detection from 42% to 100%, increases impersonation resistance from 57% to 100%, and reduces tracking risk by approximately 96%. These gains are achieved with modest overhead: authentication latency increases from 0.18 ms to 0.29 ms, per-device memory usage from 12.3 KB to 14.7 KB, and token size from 54 to 68 bytes. The results indicate that blockchain-inspired designs must be carefully adapted to imperfect real-world conditions. The proposed hardening measures substantially strengthen the AC protocol while preserving its lightweight character for secure and privacy-aware IoT deployments.

1. INTRODUCTION

The proliferation of Internet of Things (IoT) devices has fundamentally reshaped application domains such as healthcare monitoring, industrial automation, smart homes, and smart cities. Each of these domains relies on networks of small, resource-constrained devices that must authenticate themselves frequently and reliably. In such settings, the tension between robust security guarantees and limited computational, memory, and energy budgets is particularly acute [1, 2]. Public-key-based solutions, although attractive from a theoretical security perspective, often impose overheads that low-cost microcontrollers cannot sustain over long periods of autonomous operation [3, 4].

In response, the research community has proposed a variety of lightweight authentication protocols that employ symmetric cryptography and hashing as their main primitives [5, 6]. Among these, schemes inspired by blockchain concepts attempt to capture desirable properties such as tamper evidence and chronological consistency without deploying a full-fledged distributed ledger (Figure 1). In Figure 1, the AC protocol corresponds to a device-edge chaining layer: constrained endpoints maintain local token chains that are checked by lightweight verifiers, providing chronological integrity without consensus or a global ledger. The Authentication Chains (AC) protocol is a representative instance of this design philosophy: it organizes authentication tokens into a cryptographically linked chain across time, with each token encoding the outcome of a previous successful authentication and integrating identity and temporal information.

At a high level, the AC protocol seeks to achieve forward integrity and simple verification logic using only symmetric keys and keyed-hash message authentication codes (HMACs) [7]. The original proposal by Al Ahmed et al. [8] claims that such chaining can minimize per-device storage, facilitate decentralized verification, and avoid continuous synchronization with a central authority. Nevertheless, these claims depend on several assumptions: (1) sufficiently synchronized device clocks; (2) strong protection of token

material against leakage; (3) a verifier state that does not roll back; and (4) limited adversarial capability to replay or reshape protocol flows.

In practical IoT deployments, these assumptions rarely hold perfectly. Devices may operate with loosely synchronized clocks, experience intermittent connectivity, or suffer partial compromise via software vulnerabilities, side-channel attacks, or physical access [9]. Under these conditions, the security properties of AC can degrade in ways that are not obvious from the original description. We therefore conduct a systematic investigation of the protocol under a realistic attacker model and examine how its design choices behave in the presence of network delays, resets, and partial information leakage.

Our study identifies four principal weaknesses in the AC construction by Al Ahmed et al. [8]: timestamp forgery, impersonation through replay and partial compromise, state rollback exploitation when verifiers lose or revert state, and identity linkability that causes privacy leakage. Motivated by these findings, we propose targeted enhancements that preserve the original protocol's lightweight characteristics while strengthening its resilience. We then validate the improved design in a Python-based IoT testbed using Raspberry Pi and ESP32 devices over Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP), quantifying how security and performance are affected.

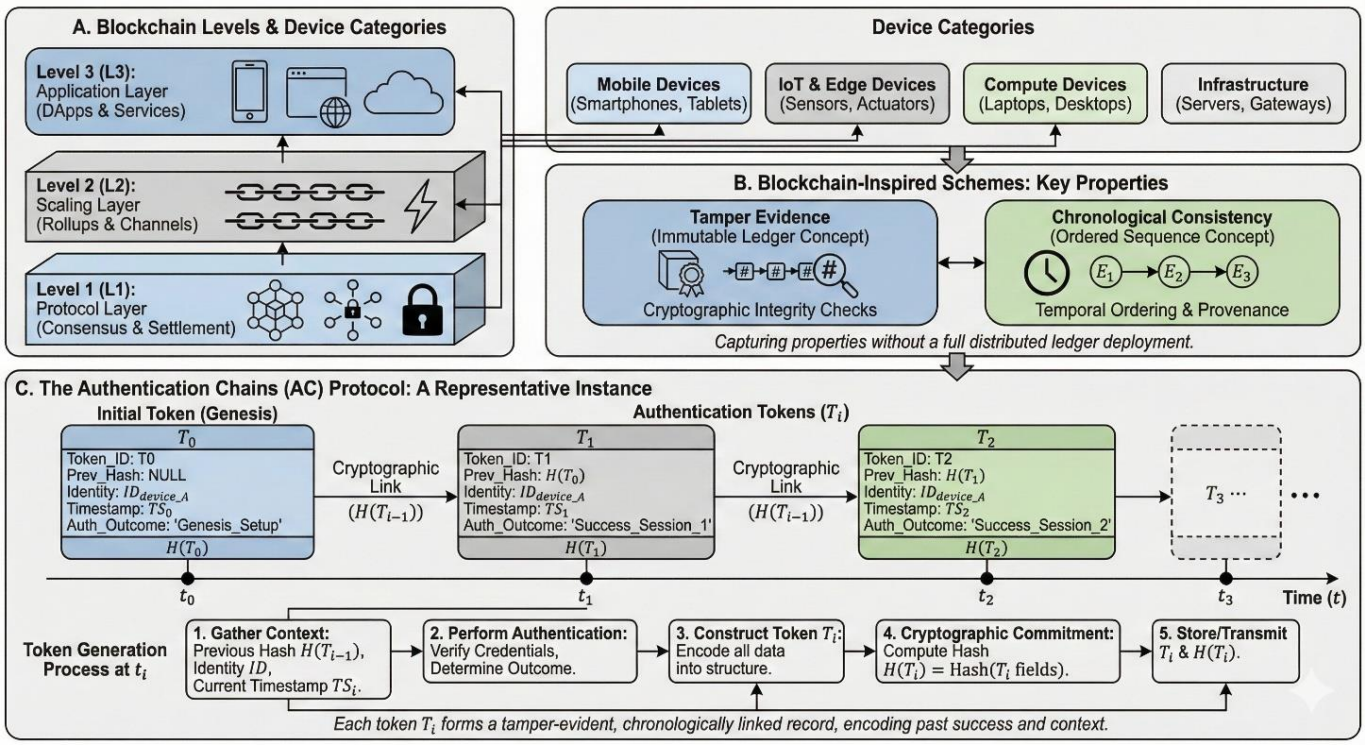

Figure 1. Blockchain levels and device categories

The remainder of this paper is organized as follows. Section 2 revisits related work on lightweight and blockchain-inspired IoT authentication. Section 3 outlines the analytical and experimental methodology. Section 4 describes the AC protocol and formalizes the identified vulnerabilities, supported by the structural view. Section 5 introduces the proposed improvements. Section 6 presents experimental results and interprets them, highlighting the security–performance trade-offs. Section 7 concludes and outlines directions for further research.

2. RELATED WORK

Research on lightweight authentication for IoT has explored several design directions, each balancing computational cost, communication overhead, and security properties. Hash-chain-based protocols constitute one of the earliest categories [9]. In these schemes, a device repeatedly applies a one-way function to an initial seed, thereby generating a sequence of authentication values. Verifiers check whether a provided value belongs to a known chain or is derived from a previously accepted state. Such constructions are appealing due to their simplicity and low computational demand. However, they often lack robust replay protection because acceptance may depend primarily on chain membership rather than being cryptographically bound to a fresh, verifier-issued challenge and a consistent verifier state, and may provide only unilateral authentication: the device authenticates to the server, but not necessarily vice versa [10]. This asymmetry can be problematic in adversarial environments, where mutual trust is required for safety-critical operations. AC partially addresses replay by chaining each token to the previously accepted value and including a timestamp, but this approach can fail when freshness checks are permissive or when verifier state diverges across gateways or after resets.

Building upon these ideas, blockchain-inspired approaches aim to provide stronger integrity and traceability guarantees by organizing authentication events in chained records reminiscent of blocks in a distributed ledger [11-13]. Instead of achieving consensus among miners, these schemes typically adopt the concept of linking successive authentication states to ensure that tampering with prior events becomes detectable. Some works implement full blockchain frameworks at

gateways or edge servers, while others attempt to simplify the ledger concept to accommodate constrained devices [12]. These designs have gained traction because they promise auditability and temporal ordering without relying on centralized trust anchors.

The AC protocol can be interpreted as an attempt to push these ideas closer to the device edge [14]. Instead of storing chains primarily at gateways, AC allows each device to maintain its own local chain of tokens, which can be verified by any party possessing the shared key. This design aims to reduce reliance on central infrastructure while preserving integrity and ordering. However, by avoiding consensus and global synchronization, AC inherently depends on local clocks and local verifier state, which can become attack surfaces under realistic IoT constraints.

Beyond AC and related chaining schemes, a line of work has explored privacy-preserving authentication in IoT using pseudonyms, group signatures, or anonymous credential systems [15]. Those constructions improve linkability resistance but can be computationally heavy. Our contribution builds on this body of work by shifting the focus from mere protocol description to exploitability analysis and deployable countermeasures. We treat AC as a concrete case study in how blockchain-inspired integrity can fail under state divergence and replay conditions, and how lightweight fixes can restore robustness without losing scalability (Figure 2).

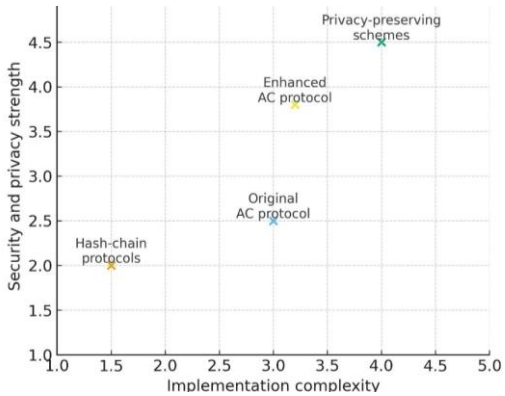

Figure 2. Comparative positioning of Authentication Chains (AC) and related Internet of Things (IoT) authentication schemes

Recent review literature reinforces that (i) lightweight anonymous authentication must explicitly treat replay and state management as first-class design elements [16], (ii) privacy-preserving IoT authentication often relies on pseudonymization as a practical alternative to heavyweight anonymous credentials [17], and (iii) secure time synchronization remains a prerequisite for timestamp-based freshness in low-power networks [18]. Complementarily, systematic surveys of IoT authentication schemes [19] and blockchain adoption for authentication management [20] underscore the importance of aligning threat models with operational realities, motivating our focus on replay, rollback, and linkability.

3. METHODOLOGY

To examine the AC protocol in a structured and reproducible way, we adopt a three-layered methodology that integrates formal reasoning, adversarial modeling, and empirical evaluation. This methodology aligns with the research flow depicted in Figure 3 and guides how we transition from conceptual analysis to concrete experimental results.

First, we perform protocol decomposition. Starting from the original specification of AC, we reconstruct the entire message exchange between devices and verifiers, paying particular attention to the formation and verification of tokens. We document the exact inputs to the keyed-hash function—device identifier, previous token, timestamp, and key—and capture how these state elements evolve over successive authentication sessions. Through this process, we derive an explicit state machine and outline the conditions under which the verifier accepts or rejects a token.

Figure 3. Overall research workflow for analyzing and enhancing the Authentication Chains (AC) protocol

Second, we conduct adversarial modeling based on a Dolev–Yao-style attacker. The adversary is assumed to have full control over the communication channel, meaning that it can intercept, modify, inject, and replay messages. Cryptographic primitives such as the HMAC and hash functions are considered ideal, in the sense that the attacker cannot invert them or find collisions faster than brute force. On top of this standard model, we introduce IoT-specific capabilities: (i) partial memory exposure in which the attacker can extract a static identifier and a limited number of recent tokens (e.g., the last 1–3 values) but not the long-term secret key unless a stronger compromise is assumed; (ii) passive capture of prior token–timestamp messages from the channel; and (iii) interaction with verifiers whose stored 'last token' state may be stale due to intermittent connectivity, snapshot recovery, or reset. By systematically enumerating these capabilities and the corresponding attack paths, we identify the conditions under which timestamp forgery, impersonation, rollback, and identity tracking become feasible.

Third, we perform empirical simulation and evaluation. We implement both the original AC protocol and our enhanced variant in a virtual testbed built with Python. The testbed emulates IoT devices on Raspberry Pi 4B and ESP32 hardware interacting over MQTT and CoAP. Devices maintain local chains and periodically authenticate to a central broker acting as the verifier. We introduce synthetic timestamp jitter, network delays, and controlled verifier state resets to reproduce the edge cases where AC is likely to fail. Unless otherwise stated, reported latency reflects cryptographic computation time for token generation/verification and excludes transport-layer waiting time; therefore, MQTT vs CoAP does not materially affect the computation-centric overheads reported in this study.

The key metrics collected in our experiments include replay detection rate, impersonation success probability, token verification latency, per-device memory consumption, and an approximate measure of tracking risk derived from simulated long-term observation of identifiers. These metrics allow us to not only confirm the attacks conceptually identified in the analysis phase but also quantify how their success rates change once the proposed enhancements are in place. Each attack scenario was repeated 100 times per condition (unless noted

otherwise), and we report Wilson 95% confidence intervals for attack success proportions to reflect variability across runs.

4. ANALYSIS OF THE AUTHENTICATION CHAIN PROTOCOL

4.1 Protocol overview

The AC protocol is designed as a symmetric-key-based, blockchain-inspired scheme for IoT authentication. Each device undergoing registration receives from a trusted authority a secret key k, a fixed device identifier ID, and an initial token T_0. For every subsequent authentication session i, the device computes a new token as shown in Eq. (1).

$$T_i = \text{HMACk}\ (\text{ID} \| T_{i-1} \| TS_i) \quad (1)$$

where, T_{i-1} denotes the previous token, and TS_i is the device's current timestamp. This formulation enforces that any change in ID, T_{i-1}, or TS_i produces a distinct output, thereby linking successive sessions through the token chain and supporting forward integrity.

On the verifier side, the protocol assumes that the verifier maintains the last accepted token T_{i-1} and associated timestamp TS_{i-1}. When the device presents (ID, T_i, TS_i) for session i, the verifier recomputes HMAC using the stored T_{i-1} and its copy of the key k. If the recomputed value matches the received T_i, and if TS_i satisfies the freshness policy relative to TS_{i-1}, the verifier accepts the authentication and updates its state to (T_i, TS_i). In principle, this simple logic avoids complex key updates and appears well-suited for constrained IoT nodes.

However, when we view this design through the lens of adversarial capability and real-world deployment, certain structural dependencies become apparent. The protocol's correctness hinges on tightly controlled clock behavior, the secrecy of past tokens, and reliable preservation of verifier state. Any deviation—such as loose time validation, token capture via eavesdropping, or verifier rollback—can be exploited to circumvent the intended guarantees. These relationships between chain structure, timestamps, and verifier state are illustrated conceptually in Figure 4, where each edge indicates a potential exploitation path if an adversary gains partial insight or control.

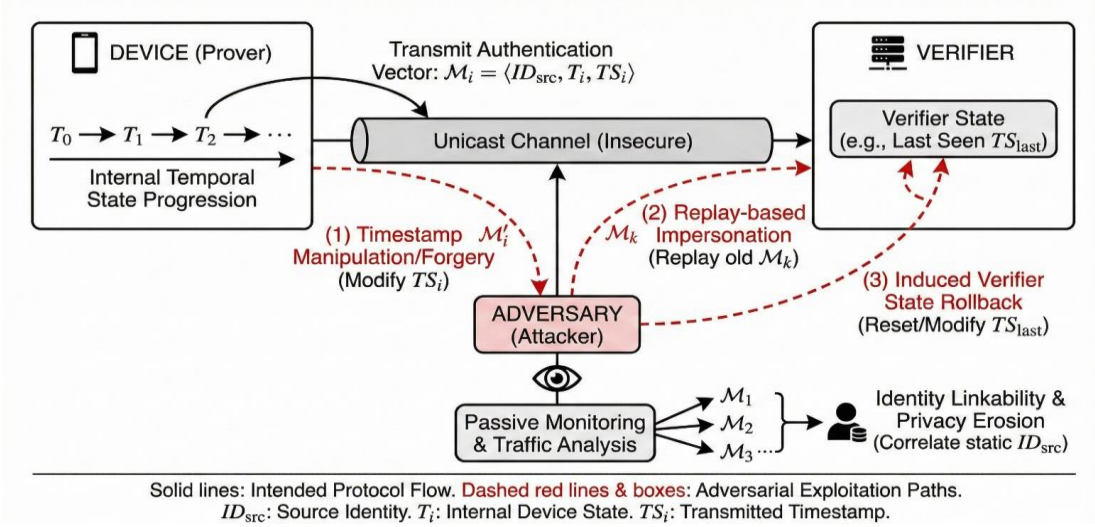

Figure 4. Structural overview and attack surface of the Authentication Chain (AC) protocol

4.2 Identified vulnerabilities

Building on this structural understanding, we examined how a realistic attacker could exploit AC and identified four main vulnerabilities. While the original protocol aims to prevent replay and tampering, our analysis reveals concrete conditions under which these goals are not fully met.

First, timestamp forgery arises when a verifier accepts any monotonically increasing timestamp. If an attacker has learned a valid previous token T_{i-1} and knows the device identifier *ID*, then—under a stronger compromise in which the secret key *k* is exposed (e.g., via side-channel leakage)—the attacker can select an artificial future timestamp $TS_{i'}$ and compute a forged token as shown in Eq. (2):

$$T_{i'} = \text{HMAC}_k(ID \parallel T_{i-1} \parallel TS_{i'}) \quad (2)$$

When the verifier accepts any timestamp that is simply larger than the stored value, the forged pair $(T_{i'}, TS_{i'})$ may pass verification. In effect, the attacker jumps ahead in the token chain, potentially excluding the legitimate device from subsequent sessions or hijacking upcoming authentications. This weakness is driven by a lax freshness policy that does not bound forward time shifts.

Second, there is a potential threat of impersonation via replay. Even without direct access to the key *k*, an eavesdropping adversary can record a valid message tuple (*ID*, T_i, TS_i). In deployments where multiple verifiers share key material but do not synchronize their current chain state, the attacker may replay this tuple to another verifier that is still at state T_{i-1} (or otherwise unaware of the latest session). Under such conditions, the second verifier may accept the replayed token and treat the attacker as the legitimate device.

Third, a rollback attack is possible. IoT infrastructures frequently undergo restarts or recovery from snapshots—for example, after firmware updates, power failures, or system maintenance. If a verifier rolls back to an older snapshot in which its stored state is T_{i-1}, an attacker can exploit this by replaying a previously valid tuple such as (*ID*, T_{i-1}, TS_{i-1}) (or a nearby state). Because the protocol encodes chain progression only through the last stored token, a state rollback can reopen previously accepted values and enable repeated reuse unless

additional anti-rollback mechanisms are enforced.

Fourth, identity linkability and privacy exposure arise because a static device identifier *ID* appears in each authentication exchange. A passive adversary can correlate repeated occurrences of the same *ID* over time, revealing device activity patterns and potentially sensitive routines in privacy-critical IoT scenarios.

Taken together, these four vulnerabilities imply that, under realistic adversarial conditions, the AC protocol does not fully achieve the security and privacy guarantees suggested in its original formulation.

5. PROPOSED IMPROVEMENTS

We propose a set of lightweight yet effective countermeasures that directly address the weaknesses outlined above while preserving the protocol's core advantages. The goal is not to redesign AC from scratch, but rather to refine its mechanisms so that they remain compatible with resource-constrained devices and existing deployments.

5.1 Nonce-based token generation

To protect against impersonation via replay, we enrich each authentication session with a verifier-chosen random nonce r_i. The verifier sends r_i as a challenge, and the device incorporates it into the token computation as in Eq. (3) (Figure 5):

$$T_i = \mathrm{HMAC}_k(ID \parallel T_{i-1} \parallel TS_i \parallel r_i) \quad (3)$$

Because r_i is fresh per session and verifier-issued, a captured token cannot be replayed without also matching the current nonce. The verifier can additionally cache recent nonces per device and reject duplicates, turning the scheme into a lightweight challenge–response chain.

5.2 Sliding timestamp window with bounded freshness

To mitigate timestamp forgery while tolerating normal clock drift, we replace the naive "greater-than" timestamp check with a bounded freshness rule using a sliding acceptance window W (Figure 6). The verifier applies the acceptance condition in Eq. (4), which constrains forward time shifts while allowing moderate delay and jitter.

$$|TS_i - t_v| \leq \mathrm{W} \wedge TS_i > TS_{i-1} \quad (4)$$

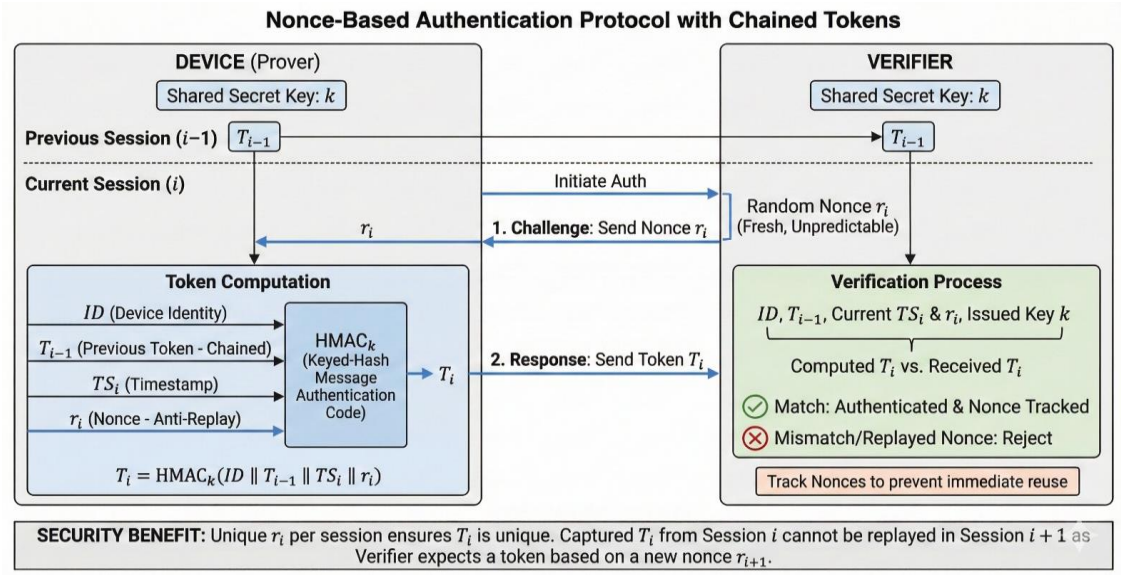

Figure 5. Nonce-based authentication protocol with chained tokens

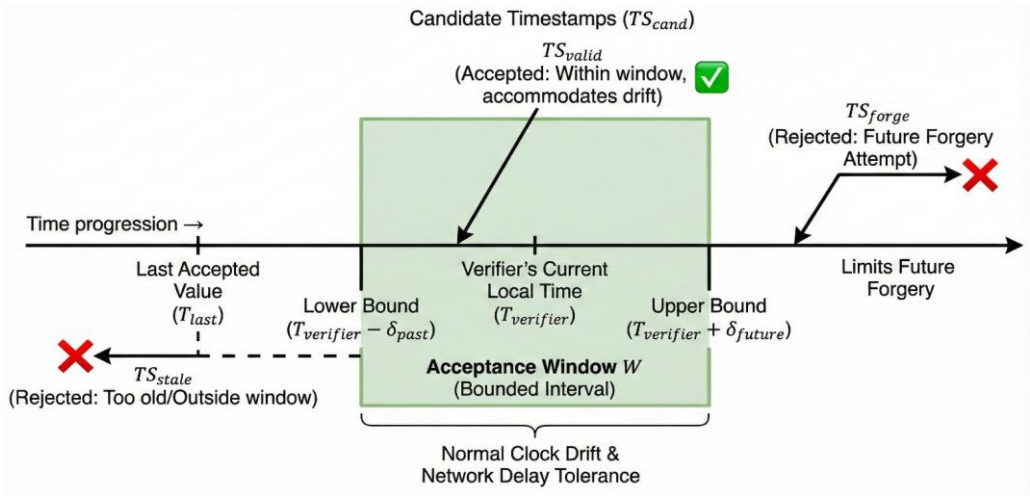

Figure 6. Sliding timestamp window with bounded freshness mechanism

5.3 Pseudonymized device identifiers

To reduce linkability, we substitute the static identifier ID with per-session or per-epoch pseudonyms. One simple instantiation is to derive a pseudonymous identifier PID_i from ID and the nonce r_i as in Eq. (5):

$$PID_i = H(ID \parallel r_i) \quad (5)$$

where, H is a cryptographic hash function. Tokens and protocol messages then carry PID_i instead of the raw ID. Verifiers knowing both ID and r_i can still verify authenticity, while passive observers see only changing pseudonyms, reducing long-term tracking.

5.4 Token expiry and revocation hints

Finally, we incorporate basic expiry and revocation hints into the token structure. Tokens can embed minimal metadata indicating validity intervals or revocation epochs. When a device is suspected or known to be compromised, a verifier can mark all tokens beyond a given point in the chain as invalid. This mechanism does not replace full-fledged certificate revocation, but it adds a practical lever for mitigating damage without immediate rekeying of the entire system.

Conceptually, these four strands—nonce inclusion, time-windowing, pseudonymization, and expiry tagging—constitute a strengthened variant of the AC protocol that remains compatible with low-end IoT hardware. We designed them so that the additional computation and storage overhead is modest, an aspect confirmed by our experimental evaluation discussed later.

6. EXPERIMENTAL RESULTS AND EVALUATION

To assess the practical impact of the proposed enhancements, we implemented both the original and the improved AC protocols in a simulated IoT environment. The testbed comprised Raspberry Pi 4B (verifier/broker) and ESP32 (constrained endpoint), communicating with a central broker over MQTT and CoAP. Each device maintained its own AC and periodically initiated sessions according to a configurable schedule, emulating typical IoT workloads (e.g., sensor data cycle transmission).

We evaluated security and performance along several dimensions. From the security perspective, we measured replay detection rate, impersonation resistance, and the success frequency of timestamp forgery under different synchronization assumptions. For replay and impersonation, we created scenarios where an attacker recorded valid sessions and attempted to reuse them either against the same verifier after a delay or against another verifier with a slightly outdated state. For timestamp forgery, we configured the attacker to choose forward-shifted timestamps and inject forged tokens, exploiting the loose validation policy of the original protocol.

The results show that the original AC protocol is vulnerable in a non-trivial fraction of cases. In our experiments (N = 100 attempts per condition), attackers succeeded in 58% of replay attempts (58/100; 95% CI 0.482–0.672) and 43% of impersonation simulations (43/100; 95% CI 0.337–0.528). These outcomes correspond to a replay detection rate of 42% and an impersonation resistance rate of 57% for the original AC protocol, matching the values reported in the abstract and Table 1. Timestamp forgery succeeded in 3 out of 5 crafted attempts (60%; 95% CI 0.231–0.882) when the verifier applied only minimal checks, and we treat this small-sample estimate as illustrative rather than definitive. These figures show that, under adversarial conditions, the nominal replay and freshness protections of the protocol can be bypassed with moderate effort. By contrast, the improved protocol—equipped with nonce-based session binding, bounded timestamp windows, and pseudonymization—blocked all replay and impersonation attempts observed in our tests. Forged-timestamp attacks also failed because injected timestamps fell outside the configured acceptance window or could not be matched to a valid nonce.

From the performance perspective, we measured token verification latency and device memory consumption on the constrained endpoint path (ESP32) to reflect deployment impact on resource-limited devices. In the original protocol, average latency per authentication remained around 0.18 ms, whereas the improved version exhibited a modest increase to 0.29 ms due to the additional processing steps and data fields. Memory usage per device grew by approximately 2.4 KB on average (from 12.3 KB to 14.7 KB), reflecting the extra storage for recent nonces and pseudonym-related state. We interpret these overheads as acceptable in modern IoT hardware, particularly in exchange for substantially improved resilience. The combined behavior of attack success rates and performance metrics for both protocol variants is illustrated qualitatively in Figure 7 and Figure 8. To contextualize, a +2.4 KB memory increase is a small fraction of ESP32-class RAM but can be non-trivial on tighter microcontrollers; in such cases, nonce-cache and pseudonym state sizes can be tuned to the platform budget while preserving the security benefits.

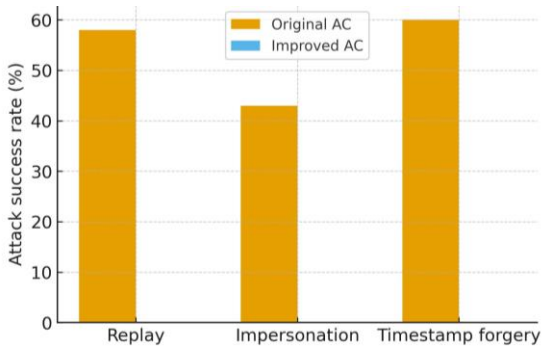

Figure 7. Attack success rates for original and improved Authentication Chains (AC) under replay, impersonation, and timestamp forgery

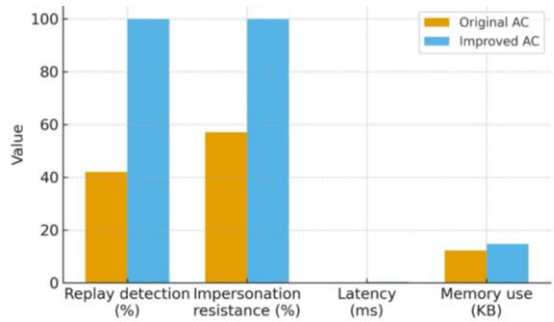

Figure 8. Security–performance trade-off between original and improved Authentication Chains (AC)

To approximate privacy improvements, we simulated a passive observer attempting to track devices based solely on identifiers or pseudonyms appearing in authentication messages. Under the original protocol with static identifiers, tracking risk was high: the observer could reliably associate sessions with specific devices over long periods. After applying pseudonymization, we estimate that effective tracking capability was reduced by about 96% in our scenario, since identifiers changed per session and could not be correlated without access to secret keys or nonces. This result aligns with the design intention that linkability be minimized without sacrificing verifiability.

The core quantitative comparison between original and enhanced protocols is summarized in Table 1. Importantly, while token size, latency, and memory usage increased slightly, these changes are small relative to typical IoT resource budgets (e.g., ESP32-class devices)**.** At the same time, replay detection and impersonation resistance improved from partial protection to complete mitigation in our experimental settings, which we regard as a substantial security gain.

Table 1. Performance comparison between the original and improved protocol

Metric	Original AC	Improved AC	Change	Security Impact
Replay Detection Rate	42%	100%	+58%	High
Impersonation Resistance	57%	100%	+43%	High
Token Size (bytes)	54	68	+14	Low
Latency (ms)	0.18	0.29	+0.11	Moderate
Memory Use (KB)	12.3	14.7	+2.4	Low
Tracking Risk	High	Minimal	-96%	High

Note: AC = Authentication Chains

7. CONCLUSIONS

In this study, we revisited the AC protocol, a blockchain-inspired lightweight authentication scheme designed for IoT networks, and examined its behavior under realistic adversarial conditions. While the protocol's chained-token mechanism initially appeared to provide efficient and scalable authentication with forward integrity, our analysis revealed four notable vulnerabilities: timestamp forgery, impersonation through replay and partial compromise, verifier state rollback, and identity linkability leading to privacy leakage.

We then proposed a series of incremental yet impactful enhancements—nonce-based token generation, sliding timestamp windows with bounded freshness, pseudonymized device identifiers, and basic token expiry and revocation hints. Through implementation and simulation in a representative IoT testbed, we demonstrated that these enhancements substantially improve resilience against replay and impersonation attacks, reduce the feasibility of timestamp manipulation, and significantly lower long-term tracking risk, all while maintaining sub-millisecond latency and modest memory usage.

From a broader perspective, our findings suggest that blockchain-inspired ideas, when applied to IoT authentication, must be carefully adapted to cope with imperfect synchronization, partial compromise, and strict privacy requirements. We regard the enhanced AC design as a practical step in this direction, illustrating how lightweight protocols can be fortified without abandoning their original efficiency goals. Future work will explore more advanced key management strategies, integration with zero-knowledge or anonymous credential mechanisms, and formal verification of privacy properties to further reinforce security guarantees in heterogeneous and large-scale IoT ecosystems.

ACKNOWLEDGMENT

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF), funded by the Ministry of Education (NRF-RS-2023-00237287).

REFERENCES

[1] Ding, X.Y., Wang, X.X., Xie, Y., Li, F.G. (2022). A lightweight anonymous authentication protocol for resource-constrained devices in Internet of Things. IEEE Internet of Things Journal, 9(3): 1818-1829. https://doi.org/10.1109/JIOT.2021.3088641

[2] Canavese, D., Mannella, L., Regano, L., Basile, C. (2024). Security at the edge for resource-limited IoT devices. Sensors, 24(2): 590. https://doi.org/10.3390/s24020590

[3] Kuo, C.W., Wei, W., Lin, C.C., Hong, Y.Y., Liu, J.R., Tsai, K.Y. (2025). Dynamic key replacement mechanism for lightweight Internet of Things microcontrollers to resist side-channel attacks. Future Internet, 17(1): 43. https://doi.org/10.3390/fi17010043

[4] Ullah, S., Radzi, R.Z., Yazdani, T.M., Alshehri, A., Khan, I. (2022). Types of lightweight cryptographies in current developments for resource constrained machine type communication devices: Challenges and opportunities. IEEE Access, 10: 35589-35604. https://doi.org/10.1109/ACCESS.2022.3160000

[5] Reddy, A.M., Gudivada, D., Rao, M.K. (2024). A lightweight symmetric cryptography based user authentication protocol for IoT based applications. Scalable Computing: Practice and Experience, 25(3): 1647-1657. https://doi.org/10.12694/scpe.v25i3.2692

[6] Radhi, B.M., Ataalla, A.F., Alsayednoor, H.M., Al-Shareeda, M.A., Almaayah, M.A., Obeidat, M. (2025). A lightweight identity authentication protocol for nano-scale IoT devices. Engineering, Technology & Applied Science Research, 15(5): 27938-27946. https://doi.org/10.48084/etasr.13449

[7] Hafeez, M.A., Shakib, K.H., Munir, A. (2025). A secure and scalable authentication and communication protocol for smart grids. Journal of Cybersecurity and Privacy, 5(2): 11. https://doi.org/10.3390/jcp5020011

[8] Al Ahmed, M.T., Hashim, F., Hashim, S.J., Abdullah, A. (2023). Authentication-chains: Blockchain-inspired lightweight authentication protocol for IoT networks. Electronics, 12(4): 867. https://doi.org/10.3390/electronics12040867

[9] Viswanathan, S., Tan, R., Yau, D.K.Y. (2018). Exploiting electrical grid for accurate and secure clock synchronization. ACM Transactions on Sensor Networks, 14(2): 1-32. https://doi.org/10.1145/3195182

[10] Liu, K.X., Guan, J.F., Yao, S., Wang, L.L., Zhang, H.K. (2024). DKGAuth: Blockchain-assisted distributed key generation and authentication for cross-domain intelligent IoT. IEEE Internet of Things Journal, 11(15): 25663-25673. https://doi.org/10.1109/JIOT.2024.3379310

[11] Aljumah, A. (2025). Blockchain-inspired distributed security framework for Internet of Things. Scientific Reports, 15: 10066. https://doi.org/10.1038/s41598-025-93690-2

[12] Fan, K., Sun, S.L., Yan, Z., Pan, Q., Li, H., Yang, Y.T. (2019). A blockchain-based clock synchronization scheme in IoT. Future Generation Computer Systems, 101: 524-533. https://doi.org/10.1016/j.future.2019.06.032

[13] Bhardwaj, S., Dave, M. (2024). Attack detection and mitigation using intelligent attack graph model for forensic in IoT networks. Telecommunication Systems, 85: 601-621. https://doi.org/10.1007/s11235-024-01105-w

[14] Bojič Burgos, J., Pustišek, M. (2024). Decentralized IoT data authentication with signature aggregation. Sensors, 24(3): 1037. https://doi.org/10.3390/s24031037

[15] Sucasas, V., Mantas, G., Papaioannou, M., Rodriguez, J. (2021). Attribute-based pseudonymity for privacy-preserving authentication in cloud services. IEEE Transactions on Cloud Computing, 11(1): 168-184. https://doi.org/10.1109/TCC.2021.3084538

[16] Yalli, J.S., Hasan, M.H., Jung, L.T., Al-Selwi, S.M. (2025). Authentication schemes for Internet of Things (IoT) networks: A systematic review and security assessment. Internet of Things, 30: 101469. https://doi.org/10.1016/j.iot.2024.101469

[17] Zhong, J., He, S., Liu, Z.C., Xiong, L. (2025). Lightweight anonymous authentication for IoT: A taxonomy and survey of security frameworks. Sensors, 25(17): 5594. https://doi.org/10.3390/s25175594

[18] Kaur, R., Rodrigues, T., Kadir, N., Kashef, R. (2025). A survey on privacy preservation techniques in IoT systems. Sensors, 25(22): 6967. https://doi.org/10.3390/s25226967

[19] Saiah, A., Benzaid, C., Younis, M., Badache, N. (2025). SRST: A secure and resilient synchronization of time for WSNs in IoT applications. Ad Hoc Networks, 169: 103749. https://doi.org/10.1016/j.adhoc.2024.103749

[20] Pham, H.A., Nguyen, C.T., Lam, T.C. (2025). Blockchain adoption for authentication: A survey. Blockchain: Research and Applications, 100383. https://doi.org/10.1016/j.bcra.2025.100383