**Issues in Information Systems**
Volume 27, Issue 2, pp. 163-177, 2026

DOI: https://doi.org/10.48009/2_iis_2026_112

# Bridging Cybersecurity Framework Deficiencies in IoT Systems: An Empirical Evaluation Using Arduino-Based Network Prototypes

**Robert Russey,** *Carson-Newman University, rcrussey@cn.edu*
**Seongyong Hong,** *Carson-Newman University, shong@cn.edu*

## Abstract

The rapid proliferation of IoT devices has fundamentally transformed both consumer and enterprise environments, with projections estimating 500 billion connected devices by 2030. This explosive growth has created an unprecedented expansion of the cybersecurity attack surface, which traditional security frameworks struggle to adequately protect. This research examines security gaps in current IoT deployments and evaluates the applicability of two prominent cybersecurity frameworks, the CKC and the NIST and Technology Cybersecurity Framework, to resource-constrained devices. Through a comprehensive literature review and framework analysis, this study identifies key challenges, including the CKC's linear attack model failing to account for simultaneous multi-endpoint compromise, the NIST framework's lack of device tier-specific guidance, and the absence of concrete controls for resource-constrained hardware. To address these gaps, this study implements and empirically validates two lightweight security controls on a model Arduino R4 WiFi-based Internet of Things network: a PUF authentication protocol leveraging SRAM startup behavior hashed with SHA-256 and a Vernam cipher-based encryption layer for protecting data payloads in transit. Wireshark packet capture analysis provides empirical evidence that the proposed controls eliminate plaintext credential exposure and provide replay attack resistance without requiring significant computational overhead. The experimental results of this study are anticipated to contribute to advancements in the field of physical cybersecurity.

**Keywords:** Internet of Things, Cybersecurity, Lightweight Cryptography, Physically Unclonable Functions

## Introduction

The Internet of Things (IoT) has fundamentally reshaped consumer and enterprise environments. Cisco projects 500 billion IoT devices globally by 2030, averaging 15 connected devices per person (Zikria et al., 2021). This scale has created an expanded attack surface that Ul Haq et al. (2023) characterize as severely under-secured, with most commercial IoT products lacking adequate security mechanisms. High-profile incidents illustrate the stakes: the 2020 SolarWinds compromise affected over 18,000 organizations, while a 2021 ransomware attack on a French hospital forced staff to revert to paper records (Djenna et al., 2021). Traditional frameworks such as the Cyber Kill Chain (CKC) and the NIST (National Institute of Standards and Technology) Cybersecurity Framework were designed for conventional IT infrastructure and carry significant limitations in IoT contexts: the CKC's linear seven-stage model does not account for simultaneous multi-endpoint compromise or physical device access; the NIST framework lacks device-tier specific guidance and concrete controls for resource-constrained hardware (NIST, 2015). This paper

identifies these gaps, proposes two lightweight controls, Physically Unclonable Functions (PUF) based encryption and Vernam cipher encryption, and validates them empirically on an Arduino-based testbed using Wireshark packet analysis. Recently, by the escalating concerns regarding IoT vulnerabilities, this research endeavor involves the meticulous evaluation of discrepancies across cybersecurity frameworks utilizing Arduino-based network prototypes. The primary objective of this study is to enhance the practical defenses, interoperability, and resilience of resource-constrained, real-world connected device systems.

## Literature Review

IoT systems face a more complex threat landscape than traditional IT, with attacks spanning three layers: device (hardware exploitation, session hijacking), network (TCP/UDP floods, RPL routing attacks, DNS amplification), and application (injection, cross-site scripting, ransomware) (Djenna et al., 2021, Tariq et al., 2023). High-profile incidents confirm the stakes: BlackEnergy (2015) disrupted Ukrainian power infrastructure; Mirai (2016) conscripted consumer IoT devices into large-scale Distributed Denial of Service (DDoS) operations; WannaCry (2017) propagated through unpatched systems globally; and SolarWinds (2020) exposed interconnected software supply chain risks, collectively causing operational and financial harm across critical sectors.

### Security Frameworks and Risk Management

The CKC (Garba, 2019) models attacks as seven sequential stages, reconnaissance, weaponization, delivery, exploitation, installation, command and control, and action on objectives, exploiting a useful asymmetry: an attacker must complete all seven, while a defender only needs to stop one. However, its linear structure maps poorly onto IoT deployments where attacks may target many endpoints simultaneously, progress non-sequentially, or exploit physical device access in ways the model does not capture. The NIST Cybersecurity Framework organizes security activities into five functions, Identify, Protect, Detect, Respond, Recover, and has seen broad adoption across federal agencies and enterprises (Aljumaiah et al., 2025). Defensive effort clusters around Identify (28%) and Protect (28%), with Recover receiving only 5% despite being essential to operational resilience. Like the CKC, NIST lacks IoT-specific guidance for resource-constrained devices, diverse protocols, or patch management for deployed endpoints, and provides no concrete cost-benefit analysis guidance for security budget allocation. The Gordon-Loeb Model partially addresses the NIST budget gap by factoring information value, breach probability, and marginal security returns to identify cost-effective investment levels (Gordon et al., 2020). Lee (2020) extends this with a four-layer IoT risk model using linear programming across ecosystem monitoring, infrastructure evaluation, risk assessment, and performance measurement. Framework gaps are compounded by human factors, 12 categories including operational errors, training gaps, and social engineering susceptibility (Aljumaiah et al., 2025). Though in automated IoT environments, social engineering is less likely; adversaries focus on the volume of technically weak, unattended endpoints, underweighting the CKC linear model and NIST's Recover function (only 5% of observed organizational effort).

### Authentication and Technical Solutions

Many IoT devices lack the processing power for conventional authentication (Delgado et al., 2023). Zhu et al. (2019) propose Physical Unclonable Functions (PUFs): instead of storing secret keys, PUF protocols derive device identity from microscopic manufacturing variations in SRAM, using only hash functions and

XOR operations with 128-bit storage overhead. The PUF response is hashed using SHA-256 (NIST FIPS 180-4); deterministic, one-way, collision-resistant, requiring ~46,248 clock cycles on constrained hardware, Figure 1 shows a diagram detailing the way the SHA-256 hash function works (Alfrhan et al., 2021; Manankova et al., 2023). Zhu et al. also demonstrates forward secrecy and untraces ability, confirming suitability for large-scale deployment. Djenna et al. (2021) argue that no single control is sufficient; defense-in-depth combining native device security, threat monitoring, network segmentation, and AI-assisted anomaly detection is required, so that failure of one control does not compromise the system.

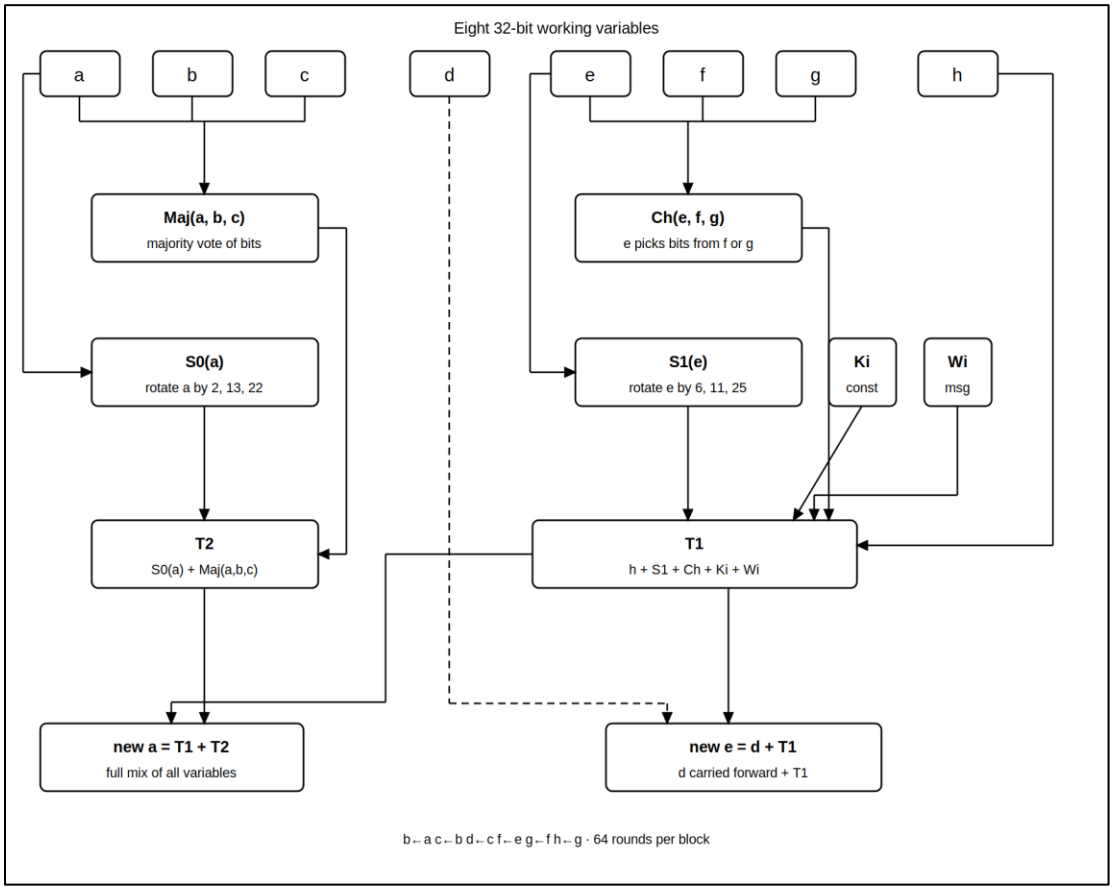



**Figure 1. The SHA-256 round function showing the six core formulas**

### Arduino Security and IoT Framework Development

The Arduino platform is widely used by hobbyists and students who prioritize functionality over security. Corno and Mannella (2023) found consistent failures across open-source Arduino projects: encryption rarely implemented for data in transit, Wi-Fi credentials hardcoded in plaintext, no replay attack defenses, and cascading resource vulnerabilities in 94% of analyzed projects. Alberca et al. (2016) demonstrate that heap buffer overflows on the Arduino Yun can escalate to root command execution across its internal bridge, exposing the platform to credential theft, rootkit installation, and firmware manipulation, risks the authors attribute to architectural memory safety deficiencies rather than model-specific flaws. Abdulghani et al. (2023) propose a Security Level Certificate (SLC) framework assigning tiered credentials based on device capability, from basic sensors (SLC1) to high-performance nodes (SLC5) with blockchain-based firmware management, concretely illustrating the device-specific guidance currently absent from NIST.

### Encryption Approaches for IoT Devices

Securing IoT data in transit requires balancing cryptographic strength against device constraints. Transport Layer Security / Datagram Transport Layer Security (TLS/DTLS) are standard, but a single handshake can transfer up to four kilobytes, far exceeding typical payloads, with latency up to 150 milliseconds (Carelli et al., 2022; Sokol et al., 2021). Sokol et al. propose replacing the TLS data phase with Vernam cipher XOR encryption after a one-time master key exchange, reducing total data volume below 10% of SSL/TLS, implementable on 8-bit microcontrollers. Thapar and Batra (2022) demonstrate ECC-ECDH key exchange on constrained hardware achieving approximately 10.8 million clock cycles, though tested only in simulation. Huo and Wang (2023) integrate AES with a chaotic authorization model for industrial IoT, tying key rotation to device health, addressing NIST's Recover function. For this testbed, Vernam encryption was selected for minimal overhead, no new library dependencies, and broadest device applicability.

### IoT Threat Modelling, Attack Graphs, and Embedded Security

Understanding how attacks propagate through IoT environments requires structured threat modelling, not merely identifying individual vulnerabilities. Salayma (2024) addresses this challenge directly by developing a dynamic attack graph framework using the Neo4j graph database. Unlike conventional attack graph approaches designed for static enterprise networks, Salayma's model accommodates the defining characteristic of IoT deployments: continuous change, as devices join and leave the network. Modelled on a healthcare use case in which a patient's body sensor network dynamically interconnects with clinic systems, the framework generates and maintains separate topology, reachability, and attack graphs, updating only the portions affected by a topology change rather than regenerating the entire graph. The result is a rapid, query able security model that can identify new attack paths within sub-second query times when systems merge or disconnect. Critically for the current study, Salayma demonstrates that even a device considered secure in isolation can expose entirely new attack paths once it connects to a second network, a dynamic analogous to an Arduino sensor joining an enterprise Wi-Fi segment and thereby becoming reachable from an external attacker. Attack-graph-based metrics including shortest path length and total path count provide quantitative risk scores that can guide countermeasure prioritization, directly informing the framework-alignment analysis conducted in this paper. At the hardware layer, the root cause of many IoT authentication weaknesses is the absence of a reliable entropy source for cryptographic key generation. Kietzmann et al. (2023) address this through a comprehensive integration of physically unclonable functions into the RIOT operating system, evaluated across 708 real-world IoT nodes. Their large-scale field study of uninitialized SRAM confirms a measurable inter-device uniqueness (fractional hamming distance of approximately 48%) alongside a bimodal bit-alias distribution caused by the physical layout of SRAM cells, a static bias that reduces effective min-entropy to roughly 75% per block. To counteract entropy loss during fuzzy extraction, the authors configure a code offset of 24 bytes with five repetitions of a concatenated Golay-repetition error correction scheme, yielding 144 bits of remaining key entropy at a reconstruction failure rate of $6 \times 10^{-9}$. The study also identifies aging-induced wear-out in the first 26.5 kB of memory attributable to long-term firmware usage, providing concrete guidance for PUF memory region selection. These findings directly underpin the SRAM PUF authentication protocol implemented in the present study: the stability and uniqueness properties quantified by Kietzmann et al.

validate the feasibility of deriving a device fingerprint from uninitialized SRAM on commercial microcontrollers such as the Arduino R4 WiFi, while their entropy analysis informs the choice of SHA-256 as the compression function for the PUF response. Reaz and Wunder (2025) complement the device-level analysis of Kietzmann et al. by situating hardware security within a holistic, layered framework spanning silicon, firmware, communication, and application. Their recommendations align closely with the controls evaluated in this paper. For random number generation, Reaz and Wunder endorse NIST SP 800-90A-compliant generators and hardware-based TRNGs as the preferred entropy source, consistent with the SRAM PUF approach, which exploits intrinsic hardware randomness without dedicated TRNG circuitry. For key generation, they recommend elliptic curve cryptography with Edwards-curve signatures (Ed25519/Ed448) in current deployments and module-lattice-based algorithms (ML-KEM, ML-DSA) as post-quantum successors. For constrained devices unable to support full TLS 1.3, the framework endorses TLS AES 128 GCM SHA256 and notes the viability of lightweight alternatives when handshake overhead would be prohibitive, the same trade-off that motivates the Vernam cipher selection in this study. Reaz and Wunder also advocate compact CBOR-encoded certificates to reduce PKI overhead on resource-constrained nodes, addressing the same device-tier gap identified in the NIST framework analysis. Together, Salayma (2024), Kietzmann et al. (2023), and Reaz and Wunder (2025) establish the theoretical and empirical foundation for dual-control design, PUF-based authentication and lightweight payload encryption evaluated in this research.

## Research Methodology

This study used a mixed-methods approach, combining a systematic literature review with hands-on empirical testing using two Arduino R4 WiFi boards. The Arduino R4 WiFi's integrated ESP32-S3 WiFi module enables wireless communication, mirroring real-world IoT sensors and controllers. Its low cost and open-source design represent resource-constrained hardware that is most commonly targeted in IoT attacks. A laptop (Windows 10, Intel Celeron N5095, 12 GB DDR4 RAM) served as the server and ran Wireshark for all packet capture. Both Arduinos and the server communicated over the same WiFi network, with Wireshark monitoring traffic passively throughout all three test phases, Figure 2 depicts how the network for the testbed is set up.

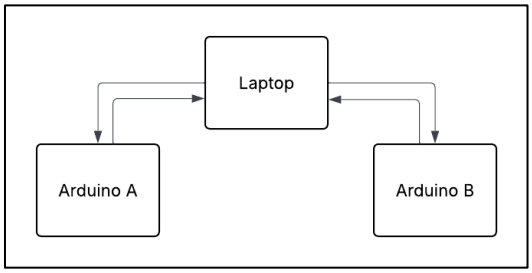



**Figure 2. Connections between the Arduinos and the laptop running the server and Wireshark**

The central security control evaluated in this study is the implementation of a Physical Unclonable Function (PUF) for device authentication, grounded in the lightweight RFID authentication protocols described by Zhu et al. (2019) and the vulnerability analysis conducted by Bendavid et al. (2018). As illustrated in Figure 3, PUFs exploit microscopic manufacturing variation in each Arduino's static random-access memory (SRAM) to generate a device fingerprint that is unique, reproducible, and impossible to extract or clone

without physically possessing the chip. When the Arduino powers on, its uninitialized SRAM cells settle into a pattern of 1s and 0s. This pattern is determined by transistor level imperfections introduced during fabrication. This raw pattern is hashed using SHA-256 (NIST, 2015) to produce a stable 32-byte PUF response. During a one-time enrollment phase shown in Phase 1 of Figure 3, this response is transmitted to the server, which stores it as the device's enrolled identity. The Arduino stores an ID which is used to ensure the server is comparing the XOR with the correct PUF. A single-device validation confirmed the Arduino R4 WiFi's SRAM produces a stable, reproducible startup pattern: the SHA-256 Hash of 64 bytes of uninitialized SRAM starting at address 0x20000000 remained consistent across multiple power cycles, validating the PUF approach before scaling to two boards as shown by Figure 4. As shown in phase 2 of Figure 3, each subsequent connection follows a three-step challenge-response protocol. First, the server sends a randomly generated number used once (nonce) to the Arduino.

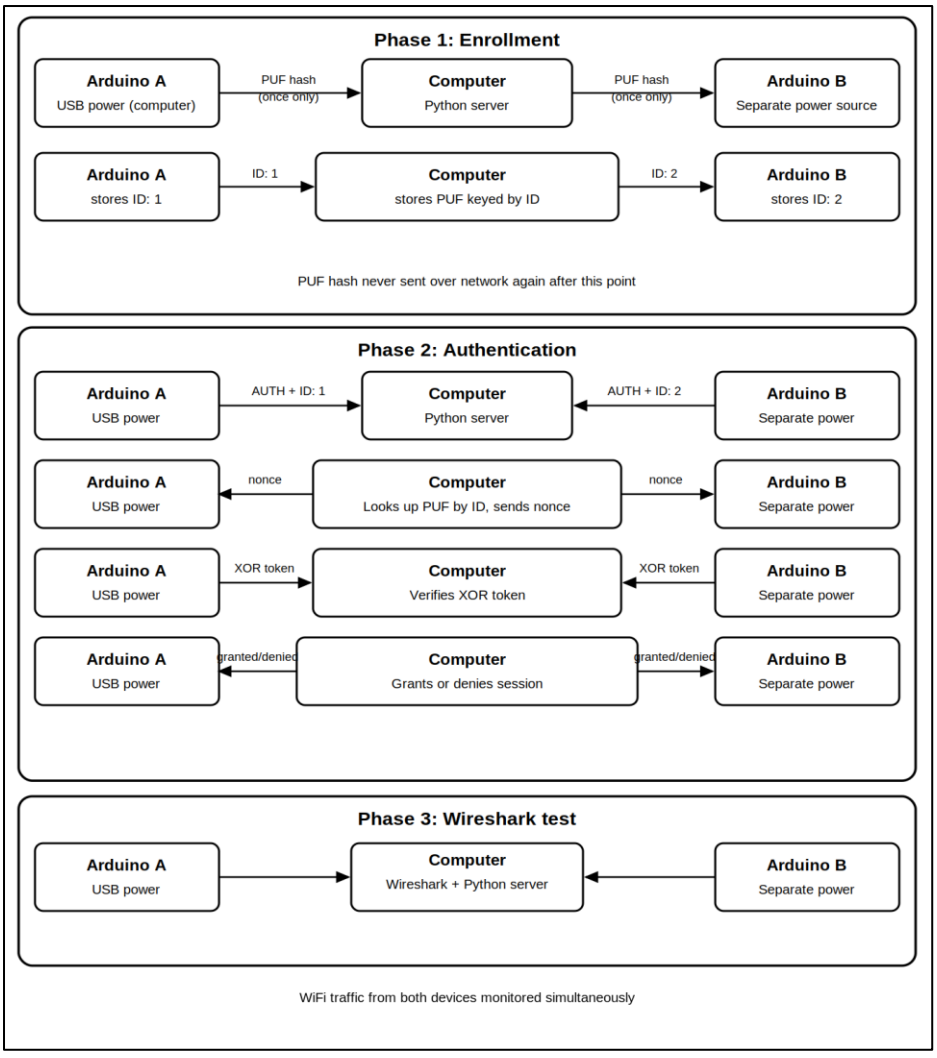



**Figure 3. PUF-based challenge-response authentication on two Arduino R4 WiFi microcontrollers**

Second, the Arduino computes the XOR (nonce, PUF response) and returns the result as an authentication token. Third, the server performs the same computation using the enrolled hash and nonce it issued and grants the session only if the two values match. Because the nonce is generated fresh for each session and

discarded after use, a token captured in a prior Wireshark session cannot be reused against a future connection. This provides inherent resistance to replay attacks without requiring any secrets to be stored on the sensor node.

```
raw sram
0A 00 00 00 FF 00 00 00 00 00 09 40 0A 00 00 00
01 12 01 00 02 02 02 01 40 41 23 6D 00 00 01 01
02 03 01 00 28 00 00 20 00 00 00 00 D0 0E 01 00
F0 0E 01 00 B0 0E 01 00 00 00 00 00 00 00 00 00
PUF hash
C85FEE7C733C1BDFDF7BD6F64C3B7CCA5FB3914FFDC8B246A0C7119CC9B9CD12
raw sram
0A 00 00 00 FF 00 00 00 00 00 09 40 0A 00 00 00
01 12 01 00 02 02 02 01 40 41 23 6D 00 00 01 01
02 03 01 00 28 00 00 20 00 00 00 00 D0 0E 01 00
F0 0E 01 00 B0 0E 01 00 00 00 00 00 00 00 00 00
PUF hash
C85FEE7C733C1BDFDF7BD6F64C3B7CCA5FB3914FFDC8B246A0C7119CC9B9CD12
```

**Figure 4. Results from the single Arduino PUF test**

Wireshark passively captured all traffic throughout all three phases. Figure 4 shows that the baseline captures the full plaintext payload visibility of the documents.

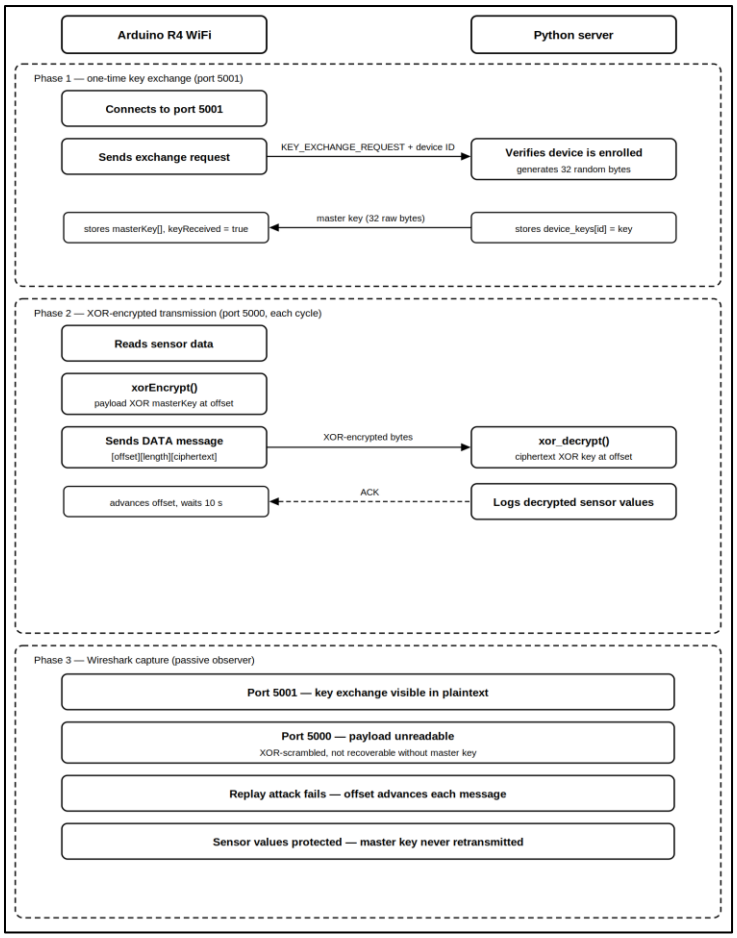



**Figure 5. Vernam cipher protocol diagram showing the three-phase implementation**

Post-implementation captures show that while the challenge-response exchange is visible, the token is cryptographically session-bound and unreadable without physically possessing the Arduino hardware. Following PUF authentication, a Vernam cipher encryption layer was added to protect sensor readings in transit. The Vernam approach was selected over ECC-based ECDH (requires the micro-ecc library with uncertain device compatibility, and approximately 10.8 million clock cycles for key exchange) and SRAM-AES (assumes deep-learning infrastructure far beyond the testbed scope) because it requires no additional Arduino libraries, imposes negligible computational overhead, and is implementable on basic 8-bit microcontrollers. In Phase 1 of Figure 5, the Arduino requests a 32-byte master key from the server; in Phase 2 of Figure 5, each sensor reading is XOR-encrypted using a rolling key slice with offset embedded in a two-byte header. The key exchange uses plain TCP rather than SSL due to the Arduino R4's rejection of self-signed certificates.

## Experimental Results

Testing was conducted in three phases using two Arduino R4 WiFi boards and a Python server, with Wireshark monitoring all network traffic throughout. In the base case, the Arduino transmitted a hardcoded authentication token over an unencrypted connection every five seconds. Wireshark confirmed the token was fully exposed in plaintext, any attacker with network access could intercept and replay the credential without specialized tools. The server accepted any device presenting the valid token with no mechanism to detect replay. Figures 6 and 7 show the Wireshark captures of packets from the base case authentication.

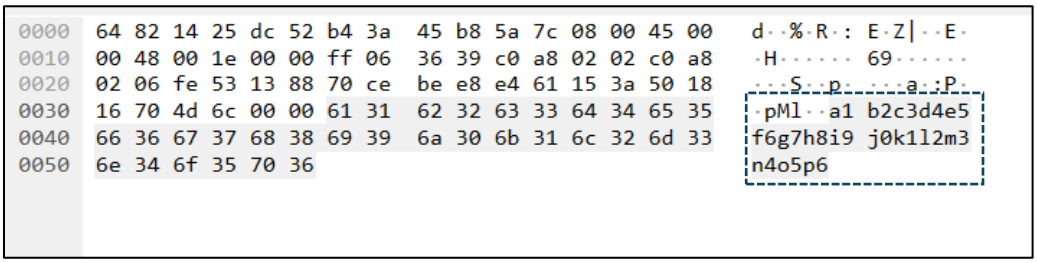



**Figure 6. Wireshark captures of first packet from base case authentication**

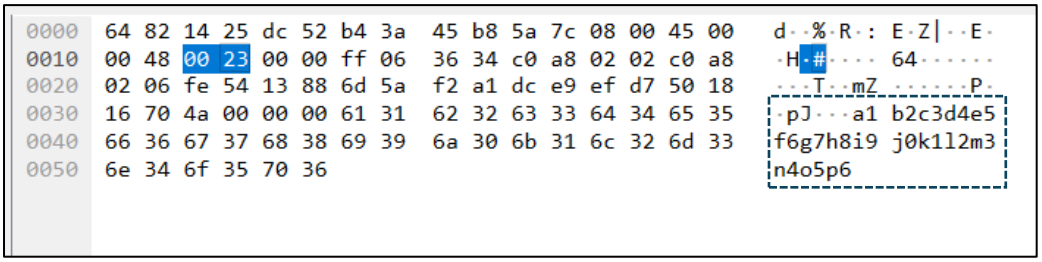



**Figure 7. Wireshark capture of second packet from base case authentication**

In the PUF-based implementation, Wireshark captures demonstrated a meaningful improvement over the base case. During enrollment, the PUF hash was transmitted once as raw data and does not reappear in subsequent exchanges. In each authentication session, the server issues a fresh 64-byte hex nonce, and the Arduino returns the XOR token. Because the nonce is regenerated each session and immediately discarded, a captured token cannot satisfy any future verification. These results confirm that PUF-based authentication eliminates plaintext credential exposure and provides replay attack resistance. Remaining limitations include Wi-Fi credentials stored in plaintext source code and the one-time enrollment hash exposure.

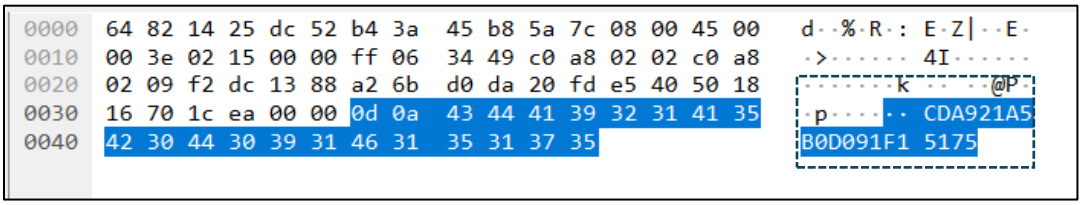



**Figure 8. Wireshark capture of PUF hash transmitted during the one-time enrollment phase**

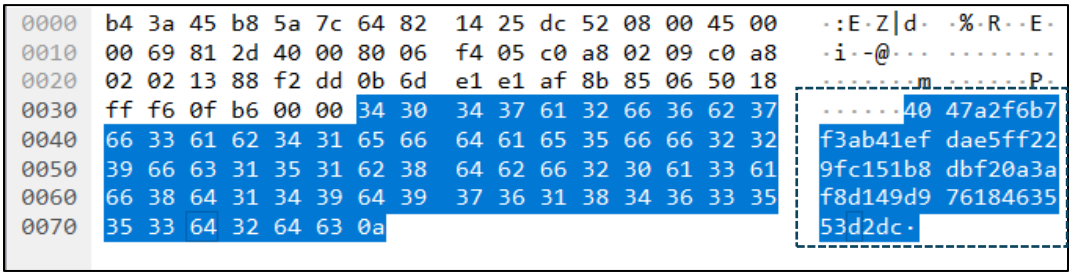



**Figure 9. Wireshark capture of server-issued nonce during PUF authentication**

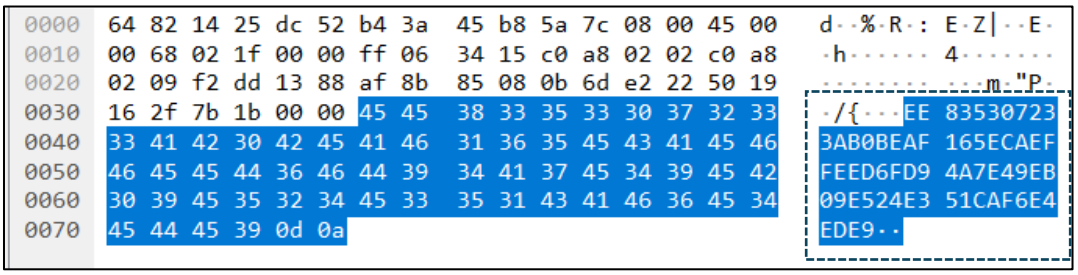



**Figure 10. Wireshark capture of XOR token during PUF authentication**

The token transmitted by the Arduino is derived by performing a bitwise XOR operation between the server-issued nonce and the device's PUF response. As demonstrated in the Wireshark captures in Figures 8 through 10, the nonce issued by the server began with the bytes 4047A2F6B7, and the PUF hash stored on the Arduino began with CDA921A5B0. XORing these values byte by byte produces a token beginning with 8DEE835307, which matches the token captured in Figure 9. This relationship can also be reversed given any two of the three values, the third can be derived. An attacker who captures both the nonce and the token from the network can recover the PUF hash by computing PUF = nonce XOR token. However, this does not constitute a practical vulnerability in the testbed implementation because the PUF hash is a hardware bound fingerprint derived from transistor level manufacturing variations in Arduino's SRAM, meaning that recovering the hash value does not allow an attacker to impersonate the device without physical possession of the chip. Therefore, the security of the protocol does not rely on the secrecy of the XOR operation itself but on the physical unclonability of the PUF response. As shown in Figure 11, when Vernam cipher encryption was enabled, Wireshark packet captures on port 5000 exhibited exclusively non-printable binary bytes in the payload field, with no recoverable or readable content derived from the packet data alone. The port 5001 key exchange remained visible in plaintext (a known testbed limitation). This before-and-after contrast across all three phases, namely the base case plaintext, PUF-authenticated plaintext, and PUF-authenticated encrypted payload, constitutes the primary empirical finding: lightweight controls eliminate the most critical IoT exposure vectors without additional hardware or library dependencies.

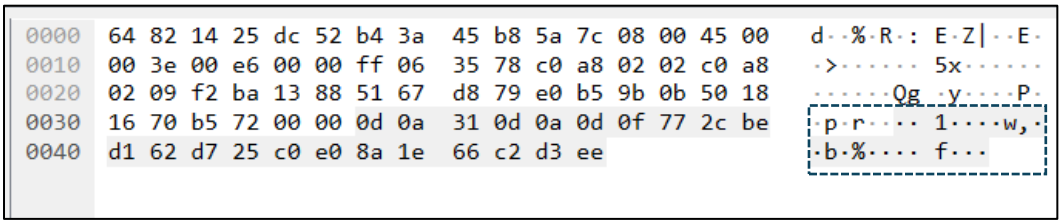



**Figure 11. Wireshark capture of XOR-encrypted payload on port 5000**

## Discussion

Packet-level analysis confirmed the three-phase security improvement. In the base case, Wireshark's Follow TCP Stream reproduced the fixed authentication token in full as human-readable text; the token appeared identically across every five-second interval, making a single interception sufficient for indefinite replay. In the PUF phase, three distinct events appear per session: a one-time 64-byte hex enrollment hash, a fresh 64-byte hex nonce per session, and a 64-byte XOR token, all of which fit within a single TCP segment with no fragmentation overhead. Because the nonce differs in every session, a captured token cannot satisfy any future verification. With Vernam encryption active, each data packet begins with a two-byte binary header (offset, length) followed by a non-printable ciphertext. Port 5001 key exchange remains the primary plaintext exposure, resolvable in production via TLS or pre-provisioned keys. Table 1 explains the comparative security properties across the three implementation phases.

**Table 1. Comparative Security Properties Across Three Implementation Phases**

| Security Property | Base Case (Figs. 6–7) | PUF Auth (Figs. 8–10) | PUF + Vernam (Fig. 11) | Notes |
|---|---|---|---|---|
| Credential visible in plaintext | Yes | No | No | Token replaced by session XOR token |
| Replay attack possible | Yes | No | No | Fresh nonce per session |
| Data payload confidentiality | None | None | XOR encrypted | Port 5000 payload fully scrambled |
| PUF hash/keys exposed on wire | N/A | Enrollment only | Enrollment only | Not repeated in subsequent sessions |
| Wi-Fi credentials in source code | Yes | Yes | Yes | Limitation across all phases |
| Extra library dependencies | None | SHA-256 | SHA-256 | Encryption adds no new dependencies |
| Single capture sufficient to auth | Yes | No | No | Nonce binding prevents reuse |

As illustrated in Table 2, the empirical results are directly applicable to the Cyber Kill Chain. In the base case, an attacker skips directly to Actions on Objectives by reading unencrypted packets, no Reconnaissance through Installation steps required. PUF authentication disrupts Exploitation through Command & Control by binding tokens to physical hardware; Vernam encryption closes the final gap by rendering intercepted payloads unreadable. Neither control addresses the early kill chain stages (Reconnaissance through Delivery) since the testbed uses a shared WiFi network without network-level access control.

**Table 2. Cyber Kill Chain Disruption Analysis Across Implementation Phases**

| Kill Chain Stage | Base Case | PUF Auth | PUF + Vernam | Notes |
|---|---|---|---|---|
| Reconnaissance | Not disrupted | Not disrupted | Not disrupted | Network traffic is still observable by passive attacker |
| Weaponization | Not disrupted | Not disrupted | Not disrupted | No control prevents offline attack preparation |
| Delivery | Not disrupted | Not disrupted | Not disrupted | Attackers can still reach the network |
| Exploitation | Not disrupted | Disrupted | Disrupted | PUF token cannot be forged without physical device |
| Installation | Not disrupted | Disrupted | Disrupted | Unauthenticated sessions rejected by server |
| Command & Control | Not disrupted | Disrupted | Disrupted | No valid session possible without enrolled PUF |
| Actions on Objectives | Not disrupted | Partial | Disrupted | Vernam cipher renders intercepted payloads unreadable |

As shown in Table 3, the implemented controls are mapped to the NIST framework. The implemented controls address the Protect function almost exclusively. They directly mitigate plaintext credential exposure, replay vulnerabilities, and unencrypted transmissions at the device level. The Identify, Detect, Respond, and Recover functions remain unaddressed, mirroring the real-world imbalance observed by Aljumaiah et al. (2025). This defines a clear design boundary: device-level controls can deliver immediate Protect-function gains on constrained hardware, while Detect, Respond, and Recover require supporting network infrastructure beyond individual microcontroller capability, a distinction that the current NIST framework does not make explicitly.

**Table 3. NIST Cybersecurity Framework Function Mapping**

| NIST Function | Gap from Literature Review | Addressed by Controls | Remaining Gap |
|---|---|---|---|
| **Identify** | No device-tier specific guidance for IoT hardware | Not addressed, controls do not classify device capability | Device capability tiering still absent |
| **Protect** | No concrete controls for resource constrained devices, plaintext credentials, unencrypted transmission | PUF auth eliminates plaintext credentials and replay attacks, Vernam cipher encrypts data payloads | Wi-Fi credentials are still hardcoded; key exchange unprotected |
| **Detect** | No IoT specific anomaly detection guidance | Wireshark passive monitoring demonstrates traffic visibility, no active detection implemented | No automated anomaly detection or alerting |
| **Respond** | No IoT specific incident response guidance | Not addressed by either control | No response capability implemented |
| **Recover** | Consistently underprioritized (5% of effort), no recovery guidance for IoT | Not addressed by either control | No recovery or key rotation mechanism implemented |

**Comparison with Prior Work**

The empirical findings of this study align with and extend several prior works. The SRAM startup stability observed across multiple power cycles on the Arduino R4 WiFi is consistent with the inter-device uniqueness results reported by Kietzmann et al. (2023), who found a fractional hamming distance of approximately 48% across 708 real-world IoT nodes, confirming that uninitialized SRAM is a reliable PUF source on commodity microcontrollers. The SHA-256 compression of the raw SRAM pattern follows directly from their entropy analysis, which showed that a min-entropy of approximately 75% per block is sufficient to derive unpredictable device-unique secrets. The forward secrecy and untraceability properties demonstrated by Zhu et al. (2019) in their lightweight RFID PUF protocol are reflected in this study's nonce-bound challenge-response design, where each session token is cryptographically distinct and cannot be reused, even when the nonce and token are both visible on the network. Bendavid et al. (2018) cautioned that PUF-based protocols may be susceptible to vulnerabilities when XOR relationships are exposed in traffic. This study acknowledges this limitation and confirms that the practical security guarantee is contingent on physical unclonability rather than protocol secrecy. In the domain of encryption, the findings of this study align with the assertions made by Sokol et al. (2021) that XOR-based encryption, following a one-time key exchange, can significantly reduce the total data volume to less than 10% of that of TLS, without necessitating additional library dependencies. This outcome is replicated in this testbed using standard Arduino hardware. The findings of the framework gap study are consistent with the real-world defensive imbalance documented by Aljumaiah et al. (2025), in which Protect-function activities accounted for 56% of the observed effort, while Recover received only 5%. This highlights the fact that device-level controls alone are insufficient to substitute for a comprehensive NIST function coverage strategy.

**Implications for Research and Practice**

For practitioners, this study demonstrates that meaningful IoT security improvements are achievable with zero additional hardware cost and minimal software overhead on commodity microcontrollers. The PUF authentication protocol requires no secure element, no pre-shared secret storage, and no extra libraries beyond a standard SHA-256 implementation, making it directly applicable to the low-margin, high-volume embedded devices that make up the majority of deployed IoT infrastructure. Organizations operating resource-constrained sensor networks should consider PUF-based device identity as an immediate drop-in improvement over hardcoded credential approaches, particularly in environments where physical device cloning is a realistic threat. The Vernam cipher layer similarly offers a practical path to payload confidentiality on devices too constrained for TLS, provided that a secure one-time key provisioning mechanism is established at enrollment. For researchers, the results highlight three productive directions. First, the key exchange and enrollment phases remain plaintext in this implementation; future work should explore lightweight authenticated key establishment protocols suitable for 8-bit and 32-bit microcontrollers. Second, scaling the testbed beyond two devices would allow evaluation of PUF uniqueness collision probability and nonce management at network scale. Third, the framework gap analysis points to an underexplored research area: translating the NIST Detect, Respond, and Recover functions into concrete, hardware-feasible controls for constrained IoT tiers, a gap that Abdulghani et al.'s (2023) Security Level Certificate framework begins to address but does not fully resolve.

### Limitations

Several limitations of this study should be noted. First, Wi-Fi network credentials remain hardcoded in plaintext source code across all three implementation phases, representing a persistent attack vector that the proposed controls do not address. Second, the PUF enrollment hash is transmitted once in plaintext during the initial enrollment phase; in a production deployment this exchange would need to occur over a secure, authenticated channel such as TLS or a pre-provisioned out-of-band mechanism. Third, the Vernam cipher key exchange on port 5001 also remains unprotected, as the Arduino R4's SSL implementation rejected self-signed certificates during testing. Fourth, the testbed is intentionally small at two devices; while this is sufficient to validate the protocol logic, it does not provide statistical evidence of PUF uniqueness collision probability at deployment scale, nor does it test nonce management under concurrent multi-device authentication. These limitations define a clear boundary for the current implementation and are resolvable in future work through secure provisioning models, automated key rotation, and larger-scale testbed evaluation.

## Conclusions

The present study corroborates the notion that the CKC and NIST Cybersecurity Framework exhibit substantial limitations when implemented in resource-constrained IoT environments. The CKC linear model is deficient in its inability to address simultaneous multi-endpoint compromise, whereas the NIST framework is deficient in its lack of device-tier-specific guidance and concrete controls for constrained hardware. The PUF-based authentication protocol, which is rooted in the SRAM startup behavior hashed with SHA-256, has been shown to eliminate plaintext credential exposure and provide inherent replay resistance. Wireshark confirmed that the tokens are cryptographically session-bound and that the PUF hash cannot be exploited without physical device possession. The Vernam cipher layer demonstrated that XOR-only encryption is sufficient to protect data payloads without the need for additional library dependencies. Collectively, these controls address three fundamental IoT vulnerabilities: plaintext credentials, replay attacks, and unencrypted transmissions. These controls were implemented entirely on commodity hardware with negligible computational overhead. Subsequent research should prioritize the development of secure provisioning models for the enrollment and key exchange phases, automated key rotation mechanisms, and the extension of the evaluation to encompass larger multi-device networks to assess collision probability and concurrent authentication performance.

## References

Abdulghani, H. A., Collen, A., & Nijdam, N. A. (2023). Guidance framework for developing IoT-enabled systems' cybersecurity. *Sensors, 23*(8), Article 4174. https://doi.org/10.3390/s23084174

Alberca, C., Pastrana, S., Suarez-Tangil, G., & Palmieri, P. (2016). Security analysis and exploitation of Arduino devices in the Internet of Things. In *Proceedings of the ACM International Conference on Computing Frontiers* (pp. 437–442). ACM. https://doi.org/10.1145/2903150.2911708

Alfrhan, A., Moulahi, T., & Alabdulatif, A. (2021). Comparative study on hash functions for lightweight blockchain in Internet of Things (IoT). *Blockchain: Research and Applications, 2*(4), 100036. https://doi.org/10.1016/j.bcra.2021.100036

Aljumaiah, O., Jiang, W., Reddy Addula, S., & Amin Almaiah, M. (2025). Analyzing cybersecurity risks and threats in IT infrastructure based on NIST framework. *Journal of Cyber Security and Risk Auditing, 2025*(2), 12-26. https://doi.org/10.63180/jcsra.thestap.2025.2.2

Bendavid, Y., Bagheri, N., Safkhani, M., & Rostampour, S. (2018). IoT device security: Challenging "A lightweight RFID mutual authentication protocol based on physical unclonable function." *Sensors, 18*(12), 4444. https://doi.org/10.3390/s18124444

Carelli, A., Palmieri, A., Vilei, A., Castanier, F., & Vesco, A. (2022). Enabling secure data exchange through the IOTA tangle for IoT constrained devices. https://doi.org/10.3390/s22041384

Corno, F., & Mannella, L. (2023). Security evaluation of Arduino projects developed by hobbyist IoT programmers. *Sensors, 23*(5), Article 2740. https://doi.org/10.3390/s23052740

Delgado, I., Sancristobal, E., Martin, S., & Robles-Gómez, A. (2023). Exploring IoT vulnerabilities in a comprehensive remote cybersecurity laboratory. *Sensors, 23*, Article 9279. https://doi.org/10.3390/s23229279

Djenna, A., Harous, S., & Saidouni, D. E. (2021). Internet of Things meet Internet of Threats: New concern cyber security issues of critical cyber infrastructure. *Applied Sciences, 11*(10), Article 4580. https://doi.org/10.3390/app11104580

Garba, F. A. (2019). The anatomy of a cyber attack: Dissecting the cyber kill chain (CKC). *Scientific and Practical Cyber Security Journal (SPCSJ), 3*(1), 29-44. https://journal.scsa.ge/wp-content/uploads/2019/04/3.1_5_faisal_ali_gabra2.pdf

Gordon, L. A., Loeb, M. P., & Zhou, L. (2020). Integrating cost-benefit analysis into the NIST Cybersecurity Framework via the Gordon-Loeb Model. *Journal of Cybersecurity, 6*(1), Article tyaa005. https://doi.org/10.1093/cybsec/tyaa005

Huo, X., & Wang, X. (2023). Internet of things for smart manufacturing based on advanced encryption standard (AES) algorithm with chaotic system. *Results in Engineering, 20*, Article 101589. https://doi.org/10.1016/j.rineng.2023.101589

Kietzmann, P., Schmidt, T. C., & Wählisch, M. (2023). PUF for the commons: Enhancing embedded security on the OS level. *IEEE Transactions on Dependable and Secure Computing*. https://doi.org/10.1109/TDSC.2023.3300368

Lee, I. (2020). Internet of Things (IoT) cybersecurity: Literature review and IoT cyber risk management. *Future Internet, 12*(9), Article 157. https://doi.org/10.3390/fi12090157

Manankova, O. A., Yakubova, M. Z., Rakhmatullaev, M. A., & Baikenov, A. S. (2023). Simulation of the rainbow attack on the SHA-256 hash function. *Journal of Theoretical and Applied Information Technology, 101*(4), 1594–1603.

National Institute of Standards and Technology. (2015). *Secure hash standard (SHS)* (FIPS Publication 180-4). U.S. Department of Commerce. https://doi.org/10.6028/NIST.FIPS.180-4

Reaz, K., & Wunder, G. (2025). A comprehensive framework for building highly secure, network-connected devices: Chip to app. *arXiv preprint arXiv:2501.13716*. https://doi.org/10.48550/arXiv.2501.13716

Salayma, M. (2024). Threat modelling in Internet of Things (IoT) environments using dynamic attack graphs. *Frontiers in The Internet of Things, 3*, Article 1306465. https://doi.org/10.3389/friot.2024.1306465

Sokol, I., Hubinský, P., & Chovanec, Ľ. (2021). Lightweight cryptography for the encryption of data communication of IoT devices. *Electronics, 10*(21), Article 2567. https://doi.org/10.3390/electronics10212567

Tariq, U., Ahmed, I., Bashir, A. K., & Shaukat, K. (2023). A critical cybersecurity analysis and future research directions for the Internet of Things: A comprehensive review. *Sensors, 23*, Article 4117. https://doi.org/10.3390/s23084117

Thapar, P., & Batra, U. (2022). Implementation of elliptical curve cryptography based Diffie-Hellman key exchange mechanism in Contiki operating system for Internet of Things. *International Journal of Electrical and Electronics Research, 10*(2), 335–340. https://doi.org/10.37391/IJEER.100245

Ul Haq, S., Singh, Y., Sharma, A., Gupta, R., & Gupta, D. (2023). A survey on IoT & embedded device firmware security: Architecture, extraction techniques, and vulnerability analysis frameworks. *Discover Internet of Things, 3*, Article 17. https://doi.org/10.1007/s43926-023-00045-2

Zhu, F., Li, P., Xu, H., & Wang, R. (2019). A lightweight RFID mutual authentication protocol with PUF. *Sensors, 19*(13), Article 2957.

Zikria, Y. B., Ali, R., Afzal, M. K., & Kim, S. W. (2021). Next-generation Internet of Things (IoT): Opportunities, challenges, and solutions. *Sensors, 21*(4), Article 1174. https://doi.org/10.3390/s21041174