



ARTICLE

Authenticated Encryption with Associated Data and ECDH-Based Key Exchange for Secure Smart Grid Power Monitoring and Simulation

Chung-Pao Lin¹, Yi-You Hou^{2,*} and Teh-Lu Liao¹

¹Department of Engineering Science, National Cheng Kung University, Tainan, Taiwan

²Department of Intelligent Commerce, National Kaohsiung University of Science and Technology, Kaohsiung, Taiwan

*Corresponding Author: Yi-You Hou. Email: yyhou@nkust.edu.tw

Received: 18 May 2026; Accepted: 16 July 2026; Published: 15 September 2026

ABSTRACT: Smart grids (SG) integrate multiple network entities to achieve automation, but their interconnected nature also exposes communication networks to various security threats, such as replay, tampering, and man-in-the-middle (MITM) attacks. Existing encryption frameworks for smart grid edge devices often suffer from high computational complexity or lack of dynamic key management, leading to key leakage risks and communication bottlenecks. To address these challenges, this research proposes a lightweight end-to-end secure communication architecture specifically designed for smart grid power monitoring. This framework employs the Message Queuing Telemetry Transport (MQTT) protocol as the asynchronous communication backbone, effectively alleviating network traffic pressure and minimizing message latency. To ensure robust data protection without compromising efficiency, we introduce an Authenticated Encryption with Associated Data (AEAD) model based on the ChaCha20-Poly1305 algorithm, tightly binding device context identity with associated data to prevent unauthorized node impersonation. Furthermore, this study combines an ephemeral Elliptic Curve Diffie-Hellman (ECDH) protocol with localized pre-shared authentication to establish a secure dynamic key exchange framework. In addition to verifying threat mitigation capabilities against the STRIDE model through targeted security analyses, the framework's performance was evaluated on an industrial testbed. Benchmark results across three experimental devices, totaling 30,000 continuous execution events (10,000 iterations per device), prove the framework achieves an exceptionally low average edge encryption latency of less than 3.5 μ s (≈ 0.0035 ms) and maintains a physical control loop latency well under 1 ms. Compared to state-of-the-art security solutions, our architecture significantly reduces cryptographic overhead and transmission latency, providing a highly secure, scalable, and easy-to-operate solution for power telemetry in modern smart grids.

KEYWORDS: Smart grid; authenticated encryption with associated data (AEAD); ChaCha20-Poly1305; elliptic curve diffie-hellman (ECDH); man-in-the-middle (MITM)

1 Introduction

1.1 Research Motivation

Conventional power grids primarily rely on centralized power generation plants and are vulnerable to unexpected power blackouts and system-wide failures [1–4]. In contrast, Smart Grids (SGs) have emerged as the next generation of power systems that improve efficiency, reliability, and sustainability by integrating power and communication infrastructures [5,6]. SGs are sophisticated cyber-physical systems that couple the physical power network with advanced computing, sensing, and communication technologies. By coordinating energy and information flows, they form critical infrastructure with far-reaching socio-economic

consequences. In addition to reliable data transmission, Economic Dispatch (ED) and unit commitment play a central role in SG energy management by allocating power demand among multiple generation units to minimize the overall operational cost [1,7]. To safeguard these critical SG operations and ensure the economic stability of ED processes, the communication backbone must strictly observe the three fundamental pillars of cybersecurity: confidentiality, integrity, and availability [2,4,8].

1.2 Overview of Modern Communication and Security Works

Communication and Security Works to achieve responsive data acquisition and flexible resource management, modern SG architectures widely deploy edge devices or data concentrators at the network periphery, alongside cloud computing infrastructures and Industrial Internet of Things (IIoT) [9,10]. Cloud platforms enable on-demand infrastructure scaling, real-time sharing of energy consumption, and pricing information, thereby reducing dedicated hardware investments [11]. Concurrently, to mitigate communication latency at the edge, application-layer protocols have been extensively utilized. Among these, the Message Queuing Telemetry Transport (MQTT) protocol has demonstrated minimal message overhead and the shortest delivery time, making it highly suitable for time-sensitive SG scenarios compared to conventional alternatives [12,13].

To protect the critical data flowing through these channels, recent studies have increasingly focused on integrating various cryptographic frameworks to address escalating cyber-threats [6]. Gunduz and Das [2,8] systematically analyzed cyber-attacks on SG applications, emphasizing that active attacks on communication lines can easily violate data integrity and availability. Aoufi et al. [14] and other researchers [15] further highlighted that False Data Injection Attacks (FDIA) represent one of the most dangerous threats, where attackers falsify data packets to manipulate meter measurements and cause severe physical and economic damage. To counter these intrusions, advanced Intrusion Detection Systems (IDS) based on federated learning have been proposed [16].

In the domain of data aggregation and privacy preservation, Ruj and Nayak [5] proposed a decentralized security framework using homomorphic encryption and attribute-based access control to preserve customer privacy at the neighborhood area network level. Ding et al. [17] and Zhang et al. [18] proposed identity-based metering data aggregation schemes supporting batch verification and fault tolerance to guarantee the privacy and integrity of metering data against internal attackers. Furthermore, for ED problems, Chen et al. [7] developed a privacy-preserving distributed ED scheme using dynamic quantization and homomorphic encryption, supported by similar floating-point homomorphic schemes [19]. Additional studies have also expanded on privacy-preserving frameworks, such as anonymous data sharing, secure multidimensional analysis, and cloud-based simulations [3,20]. In terms of authentication and key agreement, the research community has explored various lightweight mechanisms to accommodate resource-constrained smart meters. Garg et al. [21] and Chaudhry et al. [22] designed mutual authentication-based key agreement protocols leveraging Elliptic Curve Cryptography (ECC) to reduce computational overhead. Ponnuru et al. [23] integrated ECC, Physically Unclonable Functions (PUFs), and blockchain to establish a robust authentication protocol for microgrids, while Liu et al. [24] proposed a lightweight batch authentication protocol for hierarchical networks. Furthermore, key exchange models utilizing Diffie-Hellman (DH) protocols combined with composite hash functions have been evaluated to authenticate remote systems [25], and novel variants of Elliptic Curve Diffie-Hellman (ECDH) key exchange protocols using Schnorr zero-knowledge proofs have been proposed to prevent unauthorized access [26].

To reduce network overhead in resource-constrained environments, various works have demonstrated the efficacy of employing the ChaCha20-Poly1305 authenticated encryption scheme for high-speed embedded Internet of Things (IoT) applications. Bernstein introduced ChaCha as a variant of Salsa20 [27], and

Poly1305-AES Message Authentication Code (MAC) constructions were formulated for high security [28]. Recent adoptions of ChaCha20-Poly1305 for Internet Engineering Task Force (IETF) protocols and Transport Layer Security (TLS) 1.3 have proven its effectiveness [29,30]. Hardware and software implementations of ChaCha20-Poly1305 have shown significant improvements in power and area efficiency for edge nodes and RISC-V processors [31–34].

1.3 Remaining Research Gaps and Security Vulnerabilities

Despite these structural and cryptographic advancements, existing frameworks exhibit critical shortcomings that limit their viability in real-world deployment:

- **Resource Bottlenecks and Network Latency:** Traditional power information systems suffer from low server resource utilization. Furthermore, although robust in security, traditional algorithms like Rivest-Shamir-Adleman (RSA) and Advanced Encryption Standard (AES) impose prohibitive computational and energy overheads for resource-constrained IoT edge devices, failing to meet the responsiveness required by modern SG applications [13,31,33,35].
- **Privacy Risks in Untrusted Environments:** Cloud environments are frequently regarded as untrusted. Third-party access introduces risks where service providers can view sensitive electricity consumption data in plaintext, leading to severe privacy breaches and data misuse [9–11]. These concerns diminish the willingness of industrial endpoints to share data, impeding grid progress.
- **Vulnerability to Multi-Layer Cyber Attacks:** SGs rely heavily on vulnerable communication paths. External intrusions, operator errors, and equipment failures expose grids to spoofing, tampering, and replay attacks. Malicious actors can exploit these flaws to alter multidimensional consumption reports, manipulate energy markets, and destabilize the power grid [14–16,34].
- **Key Exchange and Authentication Flaws:** While Authenticated Encryption with Associated Data (AEAD) schemes like ChaCha20-Poly1305 enhance security efficiently in constrained environments [27–30], they are symmetric and rely on a secure key exchange. Although plain ECDH is employed for key exchange, it lacks inherent node authentication [21,22,24–26]. Without a mechanism to cryptographically bind device contextual identities within the transmission payload, remote access via Information Technology and Operational Technology (IT/OT) networks remains highly vulnerable to sophisticated Man-in-the-Middle (MITM) and node impersonation attacks.

1.4 The Main Work of the Entire Study

To resolve the trade-off between rigorous security and strict resource constraints, this study proposes the Secure Smart Grid Power Monitoring and Sharing System (SSGPMSS). The main focus of this integrated communication architecture is to deliver a lightweight, asynchronous, and end-to-end protected telemetry pipeline.

- **At the Communication Layer:** The system utilizes the asynchronous properties of the MQTT protocol as a backbone to alleviate traffic pressure on edge devices and minimize message latency.
- **At the Cryptographic Layer:** The framework adopts the ChaCha20-Poly1305 algorithm within an AEAD model, achieving lightweight yet high-performance confidentiality and data integrity verification without requiring dedicated hardware [31].
- **At the Key Exchange Layer:** An ephemeral ECDH protocol is tightly integrated with a localized identity verification mechanism, enabling the secure dynamic generation of unique session keys for each power monitoring device.

1.5 Core Technical Novelties and Scientific Contributions

The primary novelties and scientific contributions of the entire study are detailed as follows:

- **Unified Async-Security Framework for SG Telemetry:** We present a novel communication architecture (SSGPMSS) that seamlessly harmonizes the lightweight, asynchronous efficiency of MQTT with robust cryptographic primitives, eliminating the overhead of traditional heavy protocols.
- **Context-Bound Identity Binding and Replay Prevention:** Unlike existing frameworks that treat cryptographic operations and device identities separately, our approach utilizes the Additional Authenticated Data (AAD) field of the ChaCha20-Poly1305 (CP) scheme to bind device contextual metadata (such as device IDs, session tokens, and timestamps) directly into the AEAD authentication process. This integration prevents replay attacks, unauthorized session reuse, and message impersonation during IT/OT remote communication.
- **Holistic Protection for Grid Energy Management:** The framework simultaneously ensures confidentiality, data integrity, and authenticity for multidimensional power reports, preventing malicious data manipulation in economic dispatch (ED) processes while preserving user privacy in untrusted cloud environments.

2 Background

This section describes the approach taken in the system design and the rationale for selecting each component. The key technologies used include Smart Grid (SG), Message Authentication Code (MAC), Authenticated Encryption with Associated Data (AEAD), ChaCha20-Poly1305 (CP), Elliptic Curve Diffie-Hellman Cryptography (ECDH), and Message Queuing Telemetry Transport (MQTT).

2.1 Smart Grid

Smart grid (SG) systems rely heavily on a deeply interconnected cyber-physical infrastructure to automate power generation, transmission, distribution, and consumption [2]. Within this network, the communication layer is crucial, as it manages real-time data transfer between sensors, controllers, substations, and management tier.

In recent years, Renewable Energy Sources (RESs) and Electric Vehicle Charging Stations (EVCSs) have been widely deployed in distribution networks to reduce carbon emissions. To maintain grid stability under these dynamic conditions, the communication system must securely transmit high-frequency telemetry data with ultra-low latency. This enables the control center to perform real-time economic dispatch, instantly allocate backup capacity, or regulate harmonic filters to ensure power quality.

However, this systemic reliance on continuous communication networks introduces severe security vulnerabilities. Because the communication infrastructure is inherently exposed to external networks, it becomes highly susceptible to cyber-attacks. If left unprotected, malicious exploits can trigger catastrophic consequences, including line overloads, equipment malfunctions, and large-scale blackouts, posing substantial risks to the stability of the entire grid infrastructure [4].

2.2 Message Authentication Code (MAC)

The Message Authentication Code (MAC) is a symmetric cryptographic mechanism designed to ensure data integrity and authenticity using a shared secret key (K). As illustrated in Fig. 1, the sender computes an authentication tag (T_S) based on the message (M) and key (K), transmitting both to the receiver. The receiver then recomputes the tag (T_R) using the same key and message; a successful match ($T_S == T_R$) verifies that

the data remains authentic and unaltered during transmission, while any discrepancy indicates potential tampering or corruption.

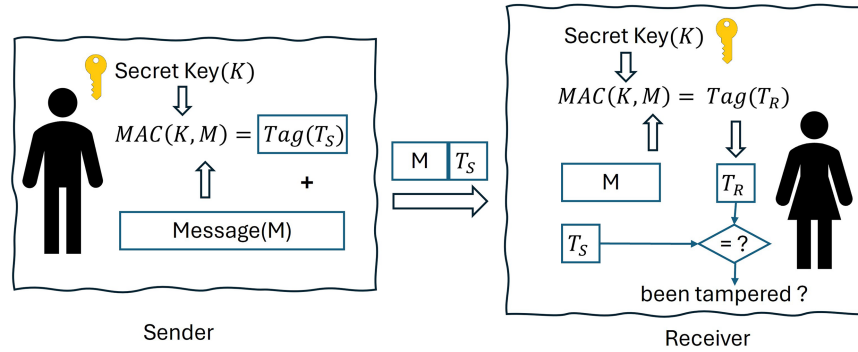


Figure 1: Message authentication code (MAC) schematic.

A standard MAC ensures message integrity but lacks confidentiality, whereas standalone symmetric encryption isolates secrecy without guaranteeing integrity. Bridging this duality through sequential multi-stage execution (e.g., Encrypt-then-MAC) inherently escalates system complexity and creates structural vulnerabilities. Improper primitive ordering, key management oversights, and logical flaws within these hybrid schemes routinely jeopardize cryptographic boundaries, introducing acute risks of ciphertext manipulation and payload forgery.

2.3 Authenticated Encryption with Associated Data

Authenticated Encryption with Associated Data (AEAD) overcomes these limitations by combining encryption and authentication into a unified process. AEAD schemes perform both confidentiality and integrity verification in one step, typically using a single symmetric key.

The Authenticated Encryption (AE) component ensures that the message content is encrypted and authenticated, while the Associated Data (AD) component allows additional information (such as headers or metadata) to be authenticated without being encrypted. Message authentication using AEAD is formalized using the model shown in Fig. 2.

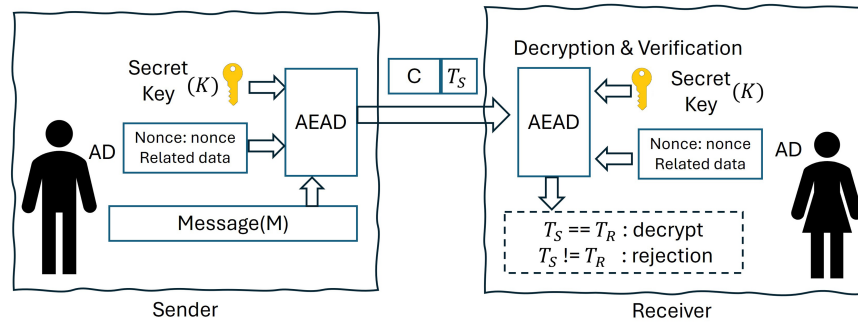


Figure 2: Authenticated encryption with associated data (AEAD) schematic.

Mathematically, an AEAD scheme consists of an encryption oracle $\mathcal{E}$ and a decryption oracle $\mathcal{D}$. Let K denote the symmetric secret key, *nonce* be a unique number used once, M represent the plaintext

message, and AD denote the associated data. The authenticated encryption process on the sender side yields a ciphertext C and an authentication tag T_S simultaneously:

$$\mathcal{E}(K, nonce, M, AD) \rightarrow (C, T_S) \quad (1)$$

Upon receiving the transmission, the receiver executes the authenticated decryption algorithm $\mathcal{D}$. This process evaluates the integrity of both the ciphertext and the unencrypted associated data in a single verification step. The receiver recalculates the authentication tag (denoted as T_R) and compares it against the received tag T_S :

$$\mathcal{D}(K, nonce, C, AD, T_S) = \begin{cases} M, & \text{if } T_S == T_R \\ \perp, & \text{if } T_S \neq T_R \end{cases} \quad (2)$$

If the tags match ($T_S == T_R$), the oracle successfully outputs the original message M . However, if any bit of the ciphertext C or the associated data AD has been tampered with by an adversary, the tag verification will fail, and the oracle will output a rejection symbol $\perp$ (bottom), discarding the payload.

This dual functionality makes AEAD particularly suitable for networked and embedded systems, where performance and security assurance are critical. AEAD also simplifies implementation by reducing the risk of misuse and enhancing resistance to certain types of attacks, such as those that exploit the separation of encryption and authentication.

2.4 ChaCha20-Poly1305

Designed by Daniel J. Bernstein, ChaCha20-Poly1305 is an AEAD algorithm combining the ChaCha20 stream cipher for confidentiality and the Poly1305 message authentication code for integrity [29,33]. Standardized by the IETF in RFC 8439 and widely adopted in modern secure protocols including TLS 1.3 and SSH, this efficient cryptographic construction delivers high performance and strong security. Notably, it is optimized for platforms lacking hardware acceleration for traditional ciphers such as AES, making it highly suitable for resource-constrained smart grid edge devices [13,29,32,35].

As illustrated in Fig. 3, the internal architecture tightly couples these two primitives. The overall authenticated encryption process E can be mathematically expressed as:

$$(CT, T) = E(M, C, K, N, Cons, AD) \quad (3)$$

where the 256-bit symmetric key K , 96-bit nonce N , block counter C , and 128-bit constants $Cons$ initialize the ChaCha20 state matrix. This state undergoes permutations to generate a pseudo-random Keystream, which is then XORed with the plaintext message M to yield the ciphertext CT .

Simultaneously, a portion of the keystream derives a one-time Poly1305 subkey, denoted as (Key_r, Key_s) . The unencrypted Associated Data AD and the generated CT are encoded and processed by the Poly1305 algorithm using this subkey to compute a 128-bit Authentication Tag T . This unified construction ensures that any unauthorized modification to CT or AD structurally triggers a tag mismatch during decryption, fulfilling the rigorous telemetry security requirements.

2.4.1 ChaCha20 Stream Cipher

ChaCha20 is a stream cipher based on a 512-bit internal state and outputs a 512-bit keystream block for each iteration. It consists of generating a keystream through a Generate Keystream process, and then performing cryptographic operations (XOR) with the message to produce the result.

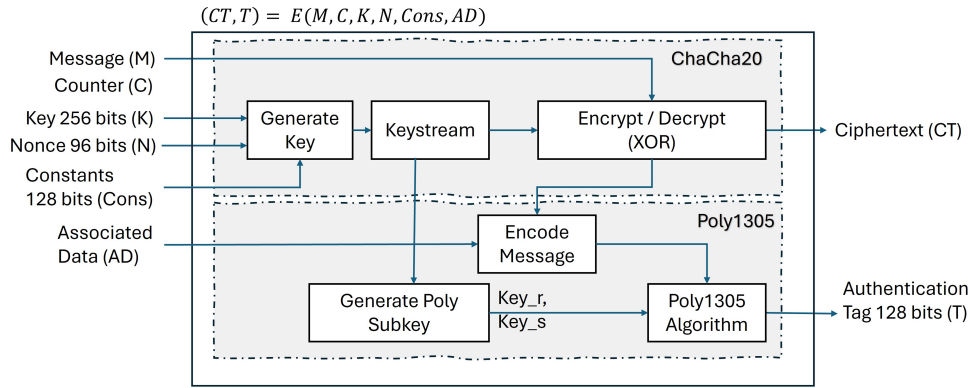


Figure 3: ChaCha20-Poly1305 AEAD algorithms.

The ChaCha20 algorithm uses a 256-bit key (K), a 32-bit initial counter (C), a 96-bit nonce (N), and 128-bit constants ($Cons$) to create an initial matrix (I_M) at the beginning of the encryption process. The I_M is represented as a 4×4 matrix containing 16 entries, each interpreted as a 32 bits unsigned integer, for a total of 512 bits. Furthermore, the matrix is arranged in little-endian format.

As shown in Fig. 4, the I_M is processed through a multiply quarter-round calculations consisting of ten double rounds, each of which consists of eight applications of the quarter-round (QR) function to produce the Operation Matrix (O_M). Subsequently, the I_M and O_M are added elementwise, and the results are serialized to produce the final keystream. This keystream is then used together with the original text (M) in the cryptographic function to perform encryption or decryption, producing the output text (CT).

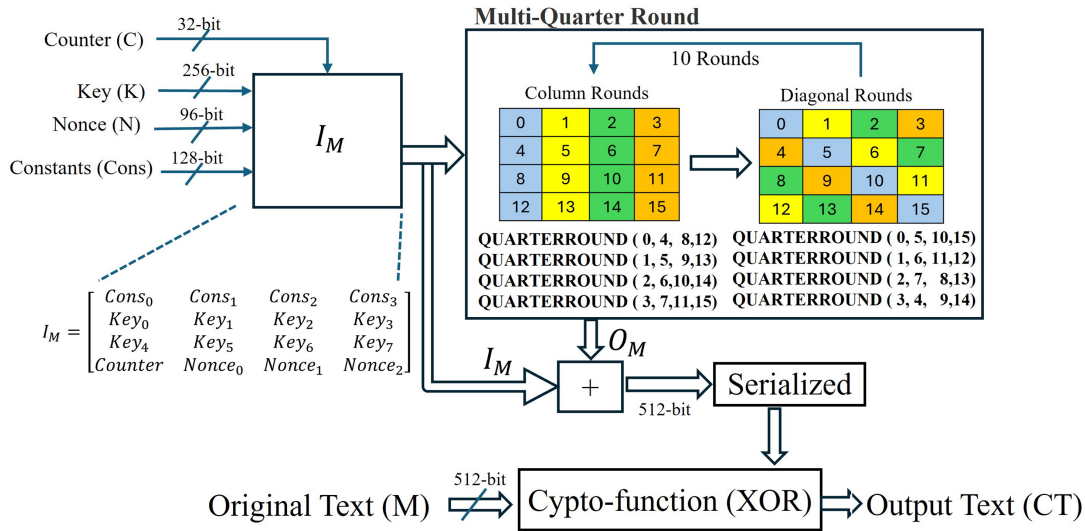


Figure 4: Architecture of ChaCha20 algorithms.

Algorithm 1 illustrates the ChaCha20 cipher process [29]. First, the algorithm initializes the state matrix using the key, nonce, and a constant value. Then, the state is subjected to 10 double-rounds (20 individual rounds in total), alternating between column and diagonal quarter-round (QR) transformations to achieve full diffusion. The resulting matrix is serialized to generate the keystream, which is then XORed with the plaintext (M) to produce the output text (CT). After each block, the counter is updated to ensure that a unique keystream is generated for every encryption cycle.

Algorithm 1: ChaCha20 cipher algorithm**Input:** $M \in (0,1)^*$, $C \in (0,1)^{32}$, $K \in (0,1)^{256}$, $N \in (0,1)^{96}$, $Cons \in (0,1)^{128}$ **Output:** $CT = ChaCha20(M, C, K, N, Cons)$

```

1:  $I_M \leftarrow Init(C, K, N, Cons);$  ▷ The Initial matrix.
2: for  $x \leftarrow 0$  to  $\lceil M/512 \rceil - 1$  do ▷  $x$ : 512-bit block index
3:    $O_M \leftarrow I_M;$ 
4:   for  $y \leftarrow 0$  to 9 do ▷ Multi-Quarter Round (10 round).
5:      $O_M[0, 4, 8, 12] \leftarrow QR(O_M[0, 4, 8, 12]);$ 
6:      $O_M[1, 5, 9, 13] \leftarrow QR(O_M[1, 5, 9, 13]);$ 
7:      $O_M[2, 6, 10, 14] \leftarrow QR(O_M[2, 6, 10, 14]);$ 
8:      $O_M[3, 7, 11, 15] \leftarrow QR(O_M[3, 7, 11, 15]);$  ▷ Column Rounds
9:      $O_M[0, 5, 10, 15] \leftarrow QR(O_M[0, 5, 10, 15]);$ 
10:     $O_M[1, 6, 11, 12] \leftarrow QR(O_M[1, 6, 11, 12]);$ 
11:     $O_M[2, 7, 8, 13] \leftarrow QR(O_M[2, 7, 8, 13]);$ 
12:     $O_M[3, 4, 9, 14] \leftarrow QR(O_M[3, 4, 9, 14]);$  ▷ Diagonal Rounds
13:  end for
14:   $S \leftarrow Serialize(O_M + I_M);$  ▷ S: Generate Keystream for cryptographic
15:  for  $z \leftarrow 0$  to 511 do
16:     $CT[512x + z] \leftarrow M[512x + z] \oplus S[z];$  ▷ Cryptographic via XOR
17:  end for
18:   $I_M[12] \leftarrow I_M[12] + 1;$  ▷ Update the counter( $C$ )
19: end for
20: return ▷  $CT$ : Output result through the cryptographic system

```

Fig. 5 shows the detailed structure of the ChaCha20 Quarter-Round (QR) algorithm. The QR function operates on four 32-bit words (a, b, c, d) and applies a sequence of modular additions, bitwise XORs, and left rotations. Each step uses a different rotation amount (i.e., 16, 12, 8, and 7 bits), with additions and XORs interleaved to thoroughly mix the input values. This process produces four updated 32-bit words (A, B, C, D) as output.

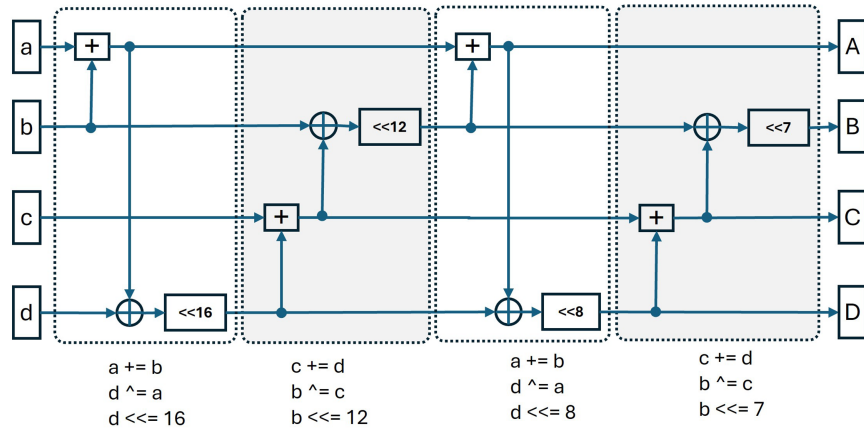


Figure 5: Quarter-Round (QR) algorithm (adapted from Serrano R, 2021). Here, a, b, c, d represent the input 32-bit words, and A, B, C, D denote the updated output words. The operator $+$ denotes 32-bit addition modulo 2^{32} , $\oplus$ represents the bitwise XOR operation, and $\ll n$ indicates a bitwise left rotation by n bits.

2.4.2 Poly1305 Message Authentication Code

Poly1305 is a one-time universal hash function designed by Daniel J. Bernstein for use as a MAC [28].

It is optimized to deliver high performance and strong cryptographic security when paired with a secure one-time key. In the ChaCha20-Poly1305 AEAD scheme, Poly1305 is employed to authenticate the ciphertext and associated data, thereby ensuring the integrity and authenticity of the data.

The core operation of Poly1305 is a polynomial evaluation modulo a large prime. Specifically, it processes the input message as a sequence of 16-byte blocks and treats each block as a little-endian integer. These integers are then used in a polynomial expression evaluated modulo $2^{130} - 5$. Formally, given a one-time 32-byte secret key (r, s) and a ℓ -byte message M , the 16-byte authentication tag is computed as follows [28]:

$$p = 2^{130} - 5. \quad (4)$$

$$n = \ell/16. \quad (5)$$

$$\text{Poly1305}(r, M) = \left(\left(\sum_{i=0}^n m_i \bar{r} \pmod{p} \right) + s \right) \pmod{2^{128}} \quad (6)$$

Here $M \mapsto (m_i)_{1 \leq i \leq n}$ chops the input message M into n , and the $n = \lceil \ell/16 \rceil$ 17-byte chunks $(m_i)_{1 \leq i \leq n}$, where each 16-byte block is padded with the byte 0x01 and the last block is padded with the byte 0x01 and $(16 - \ell) \bmod 16$ zero bytes.

Then, the function $\text{Clamp} : \{0, 1\}^{128} \rightarrow \{0, 1\}^{128}$, $r \mapsto \bar{r}$ clears some bits of r , such that

$$\bar{r} = r_0 + r_1 + r_2 + r_3,$$

where

$$r_0 \in \{0, 1, 2, 3, \dots, 2^{28} - 1\},$$

$$r_1/2^{32} \in \{0, 4, 8, 12, \dots, 2^{28} - 4\},$$

$$r_2/2^{64} \in \{0, 4, 8, 12, \dots, 2^{28} - 4\},$$

$$r_3/2^{96} \in \{0, 4, 8, 12, \dots, 2^{28} - 4\}.$$

The value $\bar{r}$ is a secret, clampable value used in the polynomial evaluation, while s is an offset added to the final result. The use of a prime modulus and a one-time key ensures that each tag is computationally indistinguishable from a random value under chosen-message attacks.

2.4.3 ChaCha20-Poly1305 Authenticator

In the ChaCha20-Poly1305 construction, the one-time key used by the Poly1305 authenticator is directly derived from the ChaCha20 cipher. Specifically,

ChaCha20 is initialized with the main encryption key, a 96-bit nonce, and a counter value initialized to zero. It then encrypts a 32-byte block consisting entirely of zeros. The output of this encryption (referred to as the first ChaCha20) is split into two 16-byte parts: the first 16 bytes are used as the Poly1305 r key (clamped to satisfy security requirements), and the remaining 16 bytes are used as the s key. The Poly1305 algorithm then uses this derived 32-byte one-time key (r, s) to generate a MAC (Message Authentication Code) tag over the ciphertext and its associated data.

This design eliminates the need for separate key management for encryption and authentication, thereby ensuring both confidentiality (via the ChaCha20 stream cipher) and authenticity/integrity (via the Poly1305

MAC) in a tightly coupled manner. Its efficiency and security make it highly suitable for deployment in resource-constrained environments as well as modern secure communication protocols.

2.5 Elliptic Curve Diffie-Hellman (ECDH)

Public-key cryptography enables secure communication between parties without the need for a pre-shared secret key. One of the earliest and most influential public-key protocols is the Diffie–Hellman (DH) key exchange. However, the traditional DH algorithm requires a key length of at least 2048 bits to achieve an acceptable level of security, which can be prohibitively expensive for devices with limited computational power.

Elliptic Curve Cryptography (ECC) was proposed by Miller and Koblitz in the 1980s to provide an alternative mathematical foundation. Using elliptic curves allows cryptographic protocols to achieve security comparable to classical methods with significantly reduced key lengths. For example, a 256-bit ECC key offers comparable security to a 3072-bit RSA key.

ECDH is an elliptic curve variant of the classical DH protocol. It runs on the elliptic curve E defined by the Weierstrass equation [26]:

$$E : y^2 \equiv x^3 + ax + b \pmod{p} \quad (7)$$

where $a, b \in \mathbb{F}_p$, the discriminant satisfies $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$, and p is an odd prime number ($p > 3$). A base point G on the curve, having a large prime order n , is publicly agreed upon.

To formalize the key exchange process, the protocol operates as follows:

- Alice randomly chooses a private scalar d_A where $d_A \in [1, n - 1]$, calculates her public key $P_A = d_A \cdot G$, and sends P_A to Bob.
- Bob randomly chooses a private scalar d_B where $d_B \in [1, n - 1]$, calculates his public key $P_B = d_B \cdot G$, and sends P_B to Alice.
- Alice receives P_B and calculates the shared secret $K = d_A \cdot P_B = (d_A d_B) \cdot G$.
- Bob receives P_A and calculates the shared secret $K = d_B \cdot P_A = (d_B d_A) \cdot G$.

By the associative and commutative properties governing scalar operations over the elliptic curve group, both parties compute the same shared secret $K = (d_A d_B) \cdot G$.

Security Properties: The theoretical security of ECDH is inherited from three fundamental cryptographic assumptions: the Elliptic Curve Discrete Logarithm Problem (ECDLP), the Computational Diffie-Hellman (CDH) assumption, and the Decisional Diffie-Hellman (DDH) assumption. Currently, no efficient algorithm is known to solve ECDLP in sub-exponential time; therefore, ECDH remains secure against passive eavesdropping when appropriate curves (e.g., NIST P-256, Curve25519) are used. However, plain ECDH primarily establishes shared secrecy and does not inherently provide message-level authentication. To safeguard smart grid operations, ECDH is integrated with an AEAD scheme (such as ChaCha20-Poly1305) to ensure data confidentiality, origin authenticity, and integrity for subsequent session communications.

2.6 Message Queuing Telemetry Transport (MQTT)

MQTT is a lightweight messaging protocol that is widely adopted in Internet of Things (IoT) applications due to its extremely low overhead and suitability for resource-constrained devices. Built on top of the TCP/IP stack, MQTT provides a publish–subscribe communication model optimized for low-bandwidth, high-latency networks.

As shown in Fig. 6, the MQTT architecture consists of four main components: Broker (B), Publisher (P), Subscriber (S), and Topic (T). In a typical deployment, the broker B acts as a central server that receives a message payload (M) from publishers, and then sends these messages to the corresponding subscribers based on predefined topics. To formalize this routing logic, the broker's dispatch mechanism can be mathematically expressed as:

$$\text{Dispatch}(M, T) \rightarrow \{S_i \mid S_i \text{ is subscribed to } T\} \quad (8)$$

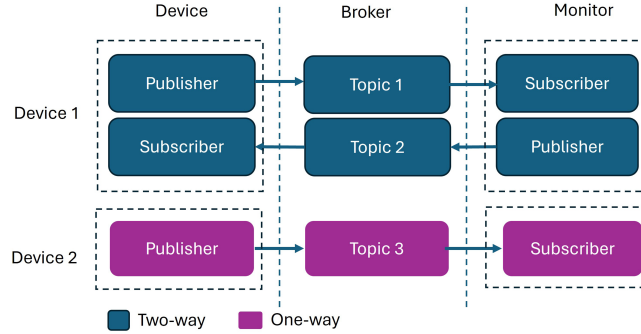


Figure 6: Schematic of MQTT architecture.

MQTT clients can act as publishers, subscribers (one-way), or both (two-way), depending on the needs of the application.

In order to ensure the stability and efficiency of data sharing between networked entities, this study combines cloud computing with the MQTT messaging protocol in the context of SG environments. The proposed system fully utilizes the scalability and centralized management capabilities of cloud computing, while leveraging MQTT to achieve lightweight, real-time communication between distributed devices.

Security Properties: Despite its high efficiency, standard MQTT operates over unencrypted TCP sockets by default, meaning it lacks inherent data confidentiality and cryptographic integrity. From a theoretical security perspective, if an adversary intercepts the network, the message M is exposed in plaintext. Furthermore, without origin authentication, the protocol is theoretically vulnerable to node spoofing and payload tampering. Therefore, to safely deploy this cloud-MQTT architecture in critical SG environments, the payload M must be encrypted and authenticated (e.g., via AEAD) *prior* to transmission, ensuring robust end-to-end security regardless of the broker's trustworthiness.

To address power telemetry vulnerabilities, this study proposes a unified secure communication architecture. We employ the ChaCha20-Poly1305 AEAD scheme to ensure data confidentiality and integrity with minimal computational overhead. To prevent static key leakage and mitigate Man-in-the-Middle (MITM) vulnerabilities inherent in plain exchanges, we tightly couple an ephemeral Elliptic Curve Diffie-Hellman (ECDH) protocol with the AEAD framework. Specifically, by leveraging AEAD's Associated Data (AD) field to bind the device's contextual identity, we render the dynamic ECDH key exchange implicitly authenticated.

Building upon this structural integration, we implement a hybrid smart grid distribution testbed framework incorporating physical UPS units. Our system demonstrates that orchestrating this context-bound AEAD, authenticated ECDH, and asynchronous MQTT communication successfully delivers end-to-end cryptographic protection, while preserving the real-time responsiveness essential for modern cloud-based smart grid operations.

3 System Design

To validate the proposed architecture, we constructed a smart grid testbed comprising three Uninterruptible Power Supply (UPS) units (emulating localized power generation) and an adversarial node (emulating cyber threats). As illustrated in Fig. 7, the communication backbone relies on a Google Cloud Platform (GCP)-hosted MQTT Broker to facilitate bidirectional, real-time telemetry acquisition and remote equipment actuation by a Monitoring Server.

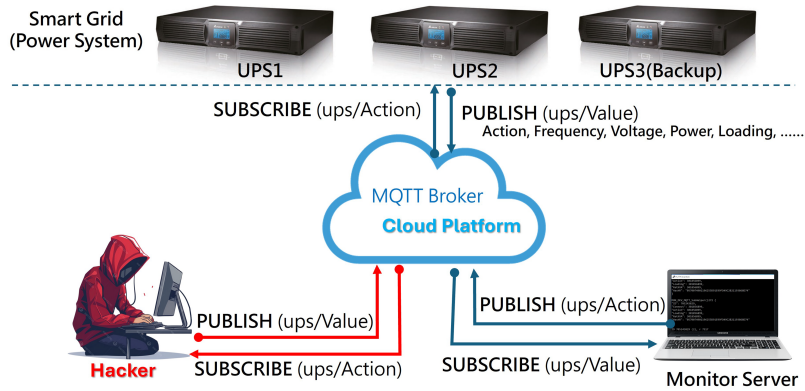


Figure 7: System design and architecture.

To secure this asynchronous data exchange, payloads are encapsulated using the ChaCha20-Poly1305 AEAD algorithm, ensuring strict confidentiality, integrity, and origin authenticity. Furthermore, an ephemeral ECDH key exchange dynamically establishes unique session keys for each UPS. By tightly coupling this dynamic ECDH exchange with context-bound AEAD, the architecture implicitly authenticates sessions, effectively neutralizing Man-in-the-Middle (MITM) attacks and symmetric key leakage.

To formalize the operational workflow, the overarching system architecture is decoupled into five interdependent functional modules, detailed in the subsequent subsections: (1) Cloud System; (2) ECDH Key Exchange System; (3) AEAD Cryptographic System; (4) Device System; (5) Monitoring System.

3.1 Cloud System

As depicted in Fig. 8, our design uses Google Cloud with the MQTT Broker and adopts the open-source “Mosquitto” software for data sharing.

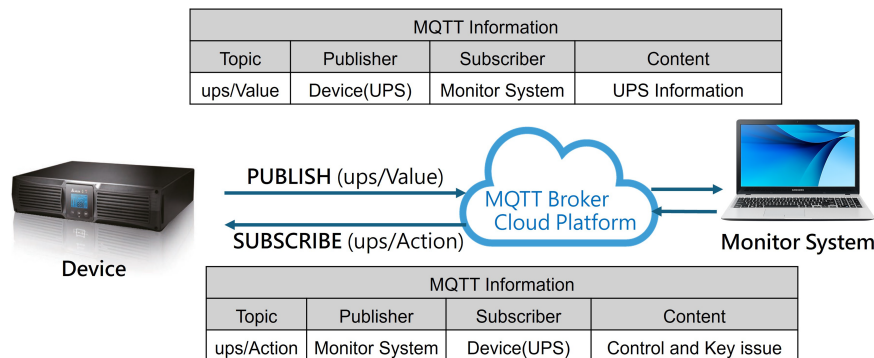


Figure 8: Cloud system and MQTT.

The MQTT setup uses a dual-subscription topic approach, allowing both the device and monitoring end to act as Publishers and Subscribers. This setup supports the exchange of power data and performs remote device control. The topics designated for power data and control are “ups/Value” and “ups/Action”, respectively.

3.2 ECDH Key Exchange System

Before establishing secure communication between the Power Supply System and the Monitoring System, an Elliptic Curve Diffie-Hellman (ECDH) key exchange is conducted to generate a shared symmetric key.

As shown in Fig. 9, each party independently generates a session key pair consisting of a public key and a private key using elliptic curve cryptography. These public keys are then exchanged securely over the network (Step 1).

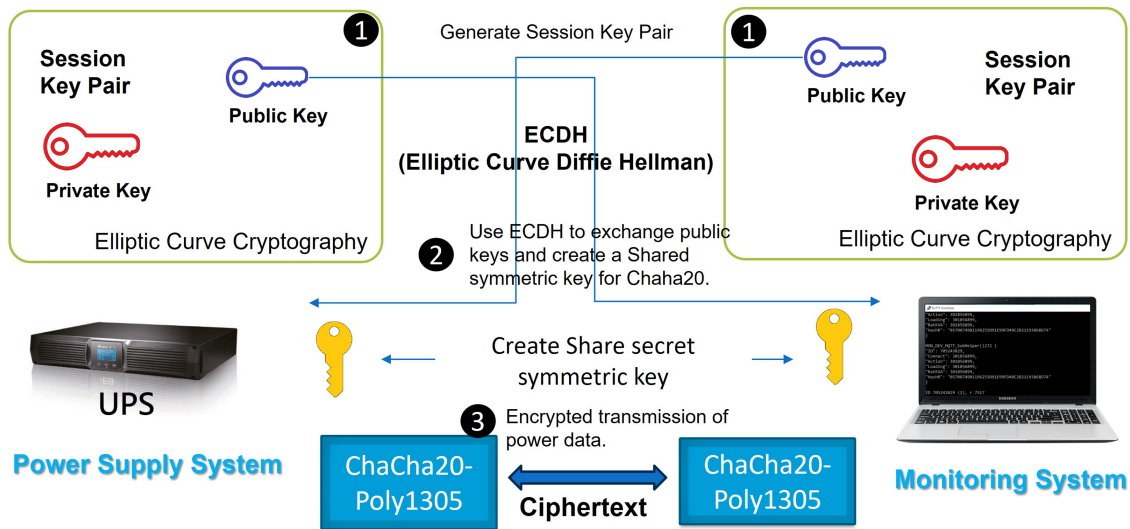


Figure 9: Cryptographic key exchange workflow between UPS and monitoring system.

After receiving the peer’s public key, both parties perform the ECDH algorithm to derive a common shared secret (Step 2), which is used to generate a symmetric session key. This session key is then used by the authenticated encryption algorithm ChaCha20-Poly1305 (Step 3) to ensure the confidentiality, integrity, and authenticity of the transmitted power data and control commands.

This decentralized key exchange mechanism eliminates the need for pre-shared symmetric keys and significantly reduces the risk of key leakage, thereby enhancing the system’s resistance to eavesdropping and man-in-the-middle attacks.

3.3 AEAD Cryptographic System

As depicted in Fig. 10, the figure illustrates the internal operation of the ChaCha20-Poly1305 algorithm, which is used for Authenticated Encryption with Associated Data (AEAD). This process accepts three primary inputs: a 256-bit symmetric key K from ECDH exchange module, a 96-bit nonce N , constants $Cons$, and associated data AD , and the plaintext message M .

ChaCha20 generates a keystream by processing the key K and nonce N in a series of counter-based blocks. The initial block (counter = 0) is used to derive a one-time 256-bit key for the Poly1305 message

authentication code. The subsequent blocks (counter = 1 to ℓ) produce the keystream that is XOR with the plaintext M to generate the ciphertext C .

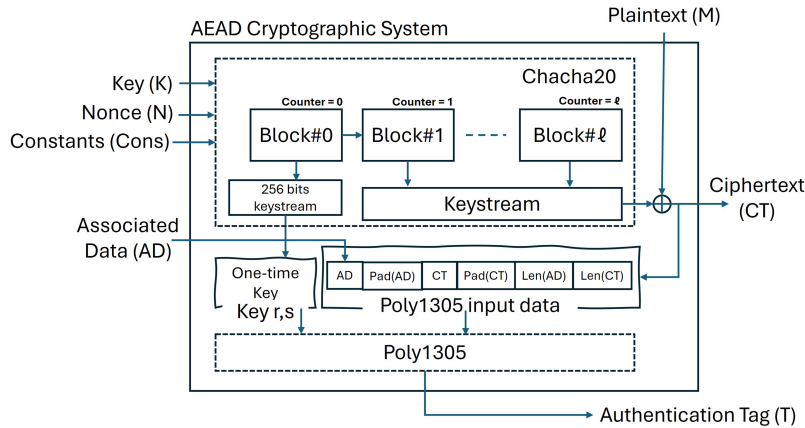


Figure 10: ChaCha20-Poly1305 AEAD cryptographic system.

Concurrently, the Poly1305 MAC processes the associated data AD , the ciphertext C , and their respective padded lengths to compute an authentication tag T . This tag ensures the integrity and authenticity of both AD and C , protecting them against tampering. The final AEAD output consists of the tuple (C, T) and is transmitted to the receiver for verification and decryption.

3.4 Device System

To achieve secure and real-time sharing of power information and output control in SG applications, we design a networked system that integrates power acquisition and distribution control, ECDH-based key exchange mechanism, ChaCha20-Poly1305 cryptographic algorithm, and MQTT-based messaging infrastructure.

As illustrated in Fig. 11, the system collects power data through a standardized UPS protocol and publishes the encrypted data using the ChaCha20-Poly1305 AEAD scheme. The encryption key is dynamically established through an ECDH exchange process between the device and the server, yielding a shared symmetric key without requiring explicit key transmission.

After receiving an encrypted control command from the MQTT broker, the system performs ChaCha20-Poly1305-based authentication to verify the validity and integrity of the command issued by the cloud server. If the verification is successful, the system will proceed to enable or disable the output power of the UPS device, thereby supporting real-time safe power dispatch in the SG environment.

3.5 Monitoring System

Our design integrates the MQTT messaging system, ChaCha20-Poly1305 cryptographic algorithm, power aggregation logic, and power distribution control units into a unified monitoring and control architecture.

Each UPS device publishes its power status in the form of encrypted messages via MQTT. Upon receiving these encrypted payloads from the MQTT broker, the monitoring system performs decryption and authentication using the ChaCha20-Poly1305 AEAD mechanism to verify the accuracy and authenticity of the power information from each device.

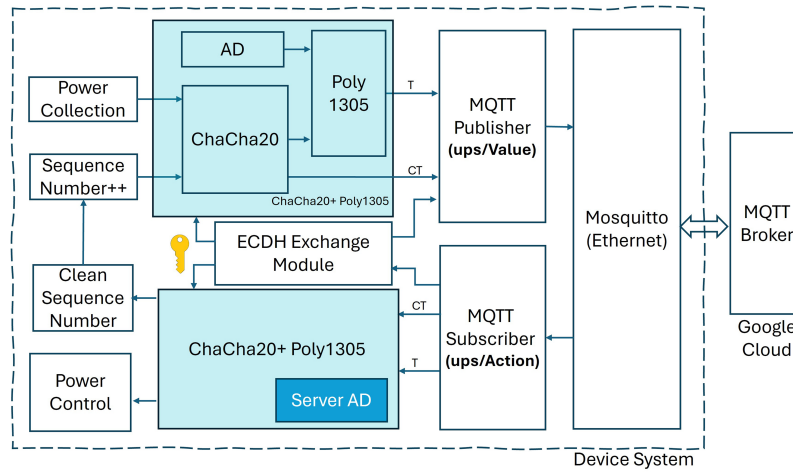


Figure 11: Device system architecture.

Similarly, the monitoring system generates power distribution commands, which are encrypted using the same ChaCha20-Poly1305 scheme before being published to the MQTT broker. The encryption keys used in both directions are dynamically established through an Elliptic-Curve Diffie-Hellman (ECDH) key exchange process between each device and the server, resulting in a shared symmetric key without the need to transmit the key directly.

This lightweight and secure communication framework ensures confidentiality and authenticity for distributed power coordination and cloud-based control in smart grid environments. The processing flow of the monitoring system is illustrated in Fig. 12.

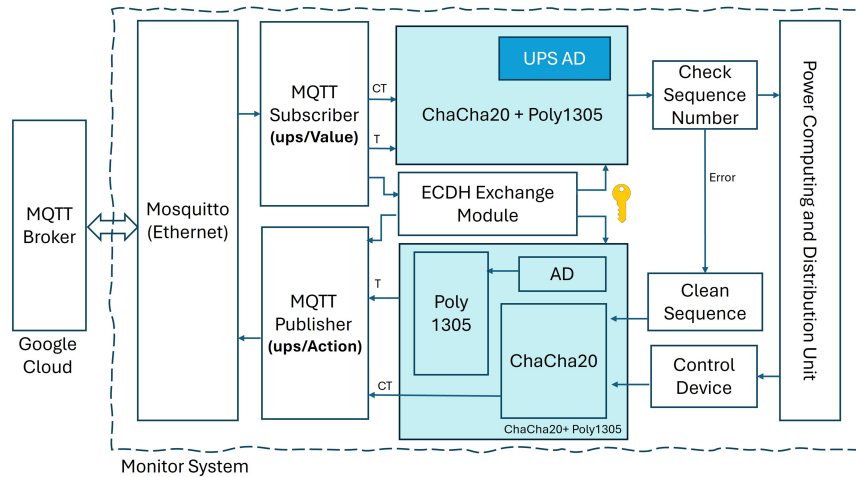


Figure 12: Monitoring system architecture.

4 System Implementation

To comprehensively evaluate the system overhead and guarantee experimental reproducibility, we deployed this architecture on a physical smart grid testbed. The precise hardware specifications, software stack, and cryptographic parameters are consolidated in Table 1.

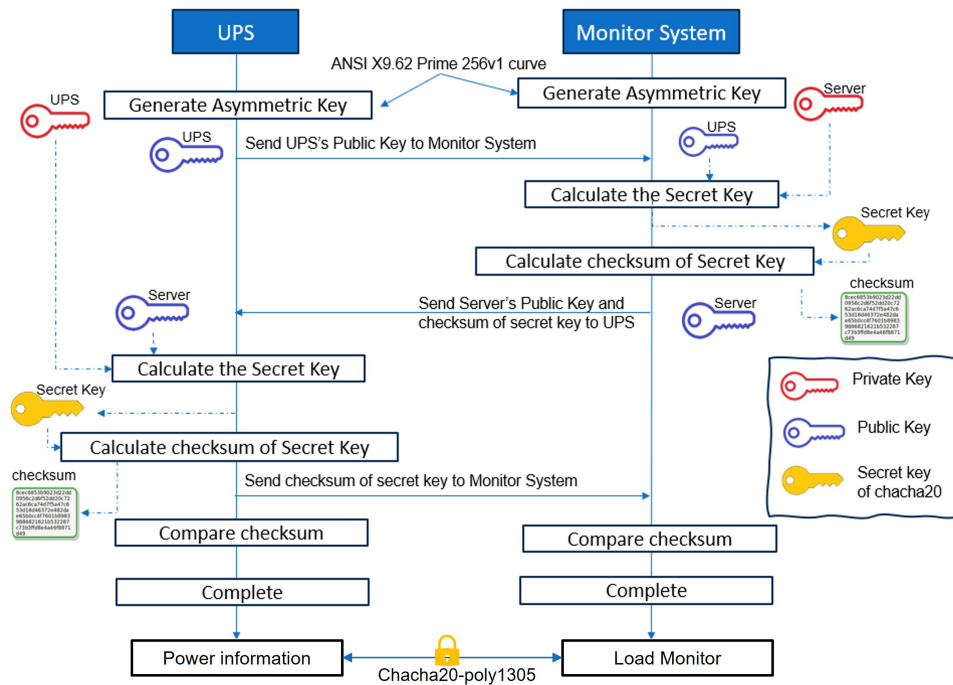
Table 1: System environment and cryptographic configurations.

Parameter	Specification
Device (UPS)	TI AM3352 (Arm Cortex-A8 @ 300 MHz, 4 GB RAM)
Monitoring Server	Intel Core i7-12700H @ 2.30 GHz, 16 GB RAM
Cloud MQTT Broker	Google Cloud Platform (GCP) e2-medium instance
Operating System	Ubuntu 22.04 LTS (Monitoring Server)/Embedded Linux (Device UPS)
Software Stack	Eclipse Paho MQTT C 1.3.2, OpenSSL 3.3.x
Key Exchange	ECDH over prime256v1 (NIST P-256)
AEAD Cipher	ChaCha20-Poly1305 (256-bit Key, 96-bit Nonce, 128-bit Tag)
Payload Configuration	1024-Byte Telemetry Data

To objectively quantify the cryptographic cost, the execution latency for key derivation and payload encapsulation was measured using high-resolution POSIX timers. All metrics were averaged over 10,000 independent iterations to systematically eliminate statistical anomalies.

In our implementation, secure communication is executed using the ChaCha20-Poly1305 AEAD scheme. To support dynamic symmetric key derivation, we utilize the ECDH protocol over the prime256v1 curve.

As illustrated in Fig. 13, the complete key-exchange workflow begins with each entity (the Monitor server and UPS nodes) independently generating an ephemeral ECC key pair. The public keys are subsequently exchanged and verified via the MQTT broker. Once the exchange is successfully completed, both parties use the ECDH algorithm to calculate a shared symmetric key. Both parties use the ECDH algorithm to calculate a shared symmetric key.

**Figure 13:** System integration of ChaCha20-Poly1305 with ECC-based key exchange.

After the key is established, all smart grid telemetry data and control commands are securely transmitted. ChaCha20 uses this symmetric key for encryption, while the Poly1305 component is responsible for ensuring the integrity and authenticity of the message.

The internal configurations and AEAD processing logic on the monitor server and UPS units are detailed in Fig. 14. In addition to encryption, the system incorporates device-specific metadata (such as the device name, MAC ID, and IP address) as Associated Data (AD) inputs to the Poly1305. These parameters are not encrypted but are cryptographically bound to each message to preserve context integrity.

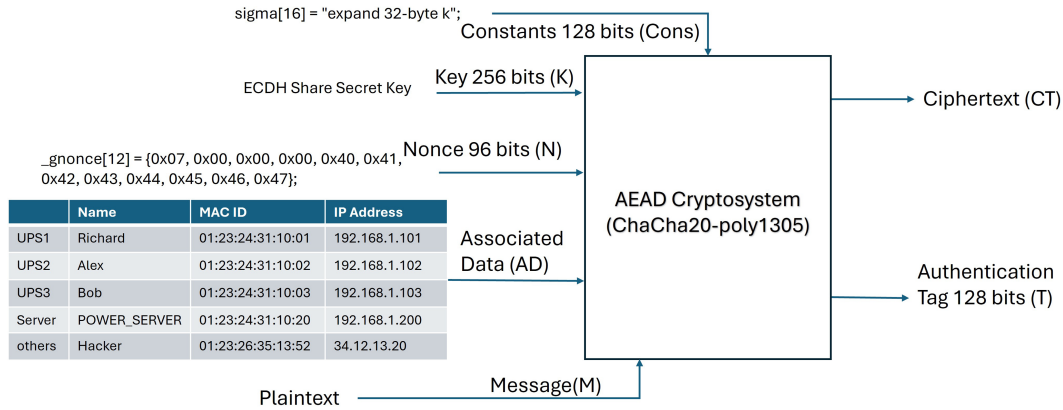


Figure 14: Practical AEAD encryption design with ChaCha20-Poly1305 and real-device metadata.

The generated ciphertext and authentication tag are transmitted via the MQTT protocol, ensuring that any tampering attempts during decryption can be reliably detected. This integration offers a lightweight, high-performance security mechanism suitable for resource-constrained SG environments. The verification encompasses the following aspects:

4.1 ECDH Key Exchange System

To establish the symmetric keys for ChaCha20 encryption, the system employs the Elliptic Curve Diffie-Hellman (ECDH) key exchange protocol. Each UPS device (UPS1 to UPS3) performs the key exchange process with the Server independently.

As a result, the Server and each UPS derive a unique but mutually shared ChaCha20 secret key. This means that although UPS1, UPS2 and UPS3 all maintain a distinct secret keys, the server holds a corresponding matching key for each UPS device.

This design ensures point-to-point confidentiality and isolates communication channels between the server and each UPS, thereby enhancing security and preventing key reuse across devices.

4.2 AEAD Cryptographic System

To verify the correctness and performance of the proposed AEAD implementation, we conducted a series of transmission tests using the shared key previously established via ECDH. Each UPS device (UPS1 to UPS3) encrypted its messages using the ChaCha20-Poly1305 algorithm and transmitted its message to the server via the MQTT protocol.

Each device performed 10,000 continuous transmissions, resulting in a total of 30,000 encrypted communication events. The objective of this verification was twofold: (1) to confirm that all transmitted messages could be successfully decrypted and authenticated by the Server; and (2) to rigorously quantify

the execution latency and deterministic stability of the ChaCha20-Poly1305 processes, thereby evaluating its practical scalability in a real-time smart grid environment.

4.3 Power Distribution

We employ two forms of UPS simulation software, designated as UPS1 and UPS2, and use a real machine UPS3 as a backup power system, as depicted in Fig. 15.

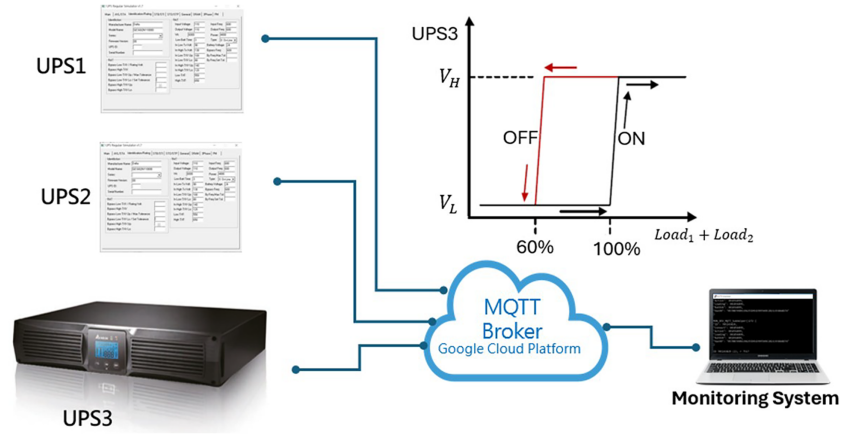


Figure 15: Architecture of power distribution.

We modify the simulated software output load separately to reflect real-time load changes.

In order to simulate SG's automatic power distribution system, we adopt the following testing criteria: when the total load percentage of the operating UPS units exceeds 100%, the monitoring system activates the output of UPS3. Conversely, when the total load drops to or below 60%, the monitoring system shuts down the output of UPS3.

4.4 Cybersecurity Verification

To systematically evaluate the risks inherent in power data transmission, we employ the STRIDE threat model to categorize potential adversarial risks following data interception. The analysis validates that an attacker, upon gaining access to the communication channel, can compromise the system through the following vectors:

- **Spoofing:** Impersonating legitimate Uninterruptible Power Supply (UPS) nodes to inject malformed telemetry.
- **Tampering:** Unauthorized modification of transmitted power metrics, compromising data integrity.
- **Repudiation:** Cannot be legally traced to the attacker.
- **Information Disclosure:** Executing packet sniffing to harvest sensitive operational patterns and power consumption profiles.
- **Denial of Service (DoS):** Executing replay attacks to induce erroneous control decisions based on outdated information, effectively manifesting as a logical DoS condition by rendering the monitoring service unreliable or unresponsive.
- **Elevation of Privilege:** Bypassing access control mechanisms to issue unauthorized turn-on and turn-off commands, granting the attacker direct control over the physical state of the power infrastructure.

As synthesized in Fig. 16, these multi-dimensional threats converge to disrupt the power load distribution logic of the monitoring system, potentially leading to cascading failures in the microgrid infrastructure. Therefore, we design these threat scenarios to verify whether the system can mitigate or resolve these threats.

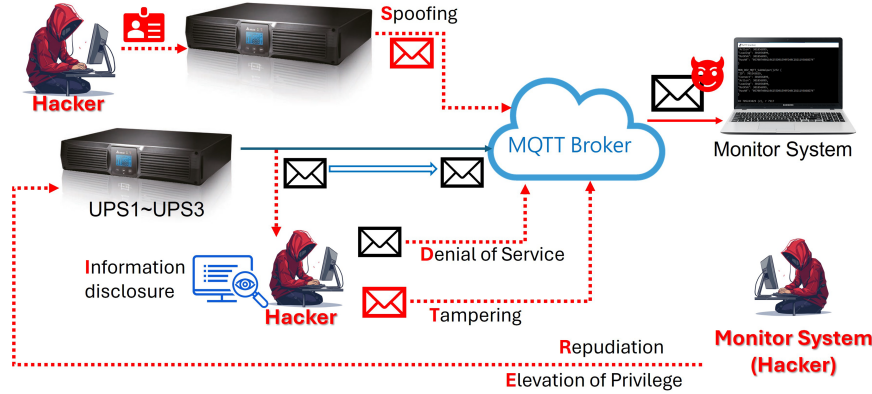


Figure 16: STRIDE threat modeling.

5 System Results

To systematically evaluate the architecture's performance and cybersecurity resilience, a rigorous four-phase workflow was executed on a physical testbed. As illustrated in Fig. 17, this validation process exhibits tight sequential dependency, where the results of each preceding phase serve as the security foundation for subsequent operations. The specific steps are delineated below:

- **Phase 1:** This phase verifies the system's capability to dynamically generate independent symmetric keys, establishing a cryptographic foundation for subsequent phases and mitigating the risks associated with static key compromise.
- **Phase 2:** Utilizing the session keys established in Phase 1, this phase validates the cryptographic binding and encryption of real-time power telemetry data with hardware-specific attributes, ensuring data confidentiality and integrity under ultra-low latency constraints.
- **Phase 3:** Based on the high-fidelity telemetry data secured in Phase 2, this phase evaluates the server's competence in issuing precise remote control commands under dynamic load variations, thereby achieving automated power distribution within the microgrid.
- **Phase 4:** To synthesize the efficacy of the aforementioned mechanisms, systematic penetration tests aligned with the STRIDE threat model were conducted to validate the resilience of the proposed architecture against real-world adversarial exploits.

5.1 ECDH Key Exchange System

To guarantee the security of all subsequent data transmissions, a secure cryptographic baseline must be established prior to initializing power telemetry. As validated by the log results in Fig. 18, the ECDH key exchange procedure ensures the successful establishment of shared symmetric keys between each UPS device (UPS1 to UPS3) and the monitoring server. Through an asynchronous handshake facilitated by the MQTT broker, both communicating entities independently compute and generate a unique, matching 32-byte ChaCha20 secret key, with the server maintaining corresponding key parameters for each individual device. In terms of smart grid cyber defense, this execution successfully achieves point-to-point confidentiality and secure domain isolation. Because each edge device maintains an independent dynamic session key, this mechanism eliminates the vulnerability of long-term static key storage and ensures secure key derivation

without over-the-air plaintext transmission. These securely generated, node-independent dynamic symmetric keys will directly serve as the primary input parameters for the subsequent Authenticated Encryption with Associated Data (AEAD) module in Section 5.2 to achieve secure data encapsulation.

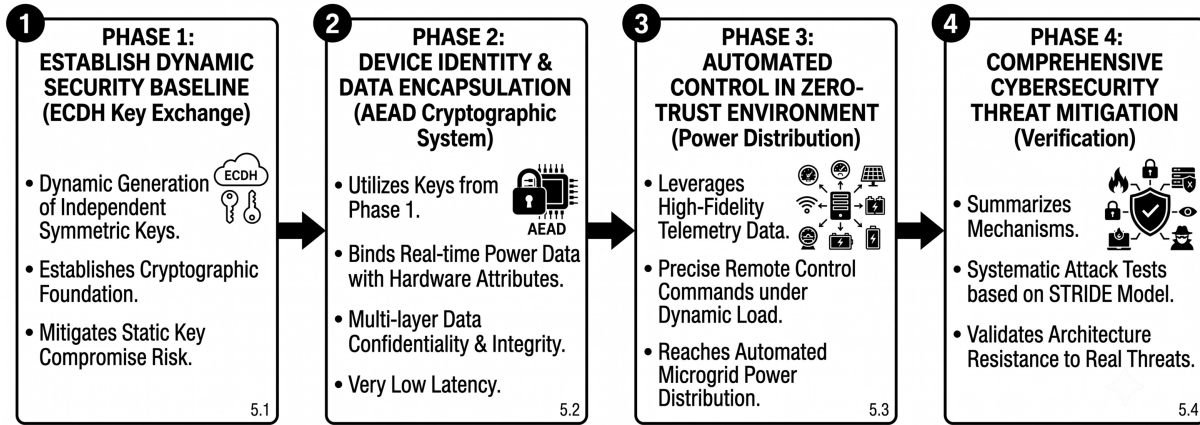


Figure 17: Sequential four-phase operational workflow for the systematic validation of system performance and cybersecurity resilience.

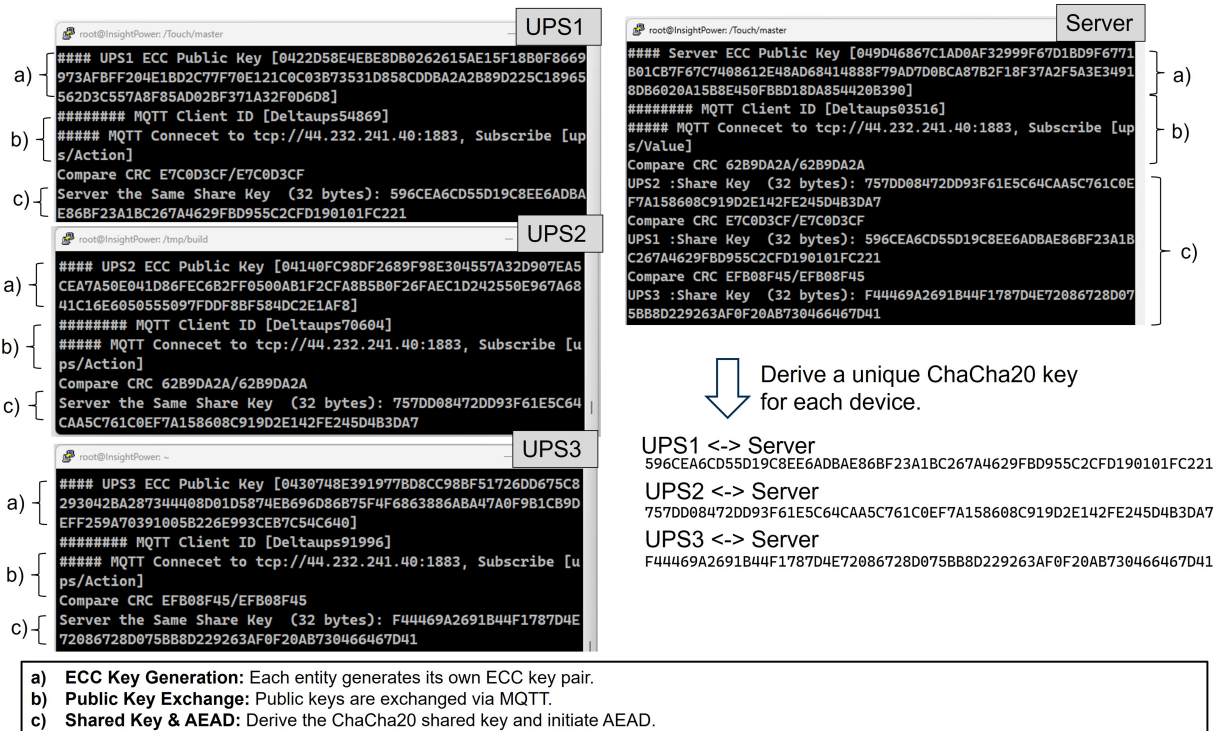


Figure 18: ECDH key exchange and verification process via MQTT: (a) Key generation step where each entity creates an ECC key pair. (b) Public key exchange stage via MQTT. (c) Verification of identical derived ChaCha20 shared keys between the server and multiple client devices to initiate AEAD-encrypted communication.

5.2 AEAD Cryptographic System

Building upon the shared keys from the previous phase (Section 5.1), this phase evaluates the cryptographic encapsulation of real-time power telemetry. As demonstrated in Figs. 19 and 20, the system executes ChaCha20-Poly1305 authenticated encryption, where UPS devices encapsulate power data via MQTT, and the server incorporates hardware metadata (MAC/IP addresses) as Associated Data (AD) for decryption and source authentication. To rigorously validate real-time responsiveness and scalability, the cryptographic overhead was quantified using a total of 30,000 execution samples collected across three distinct experimental devices (10,000 continuous iterations per device). The empirical results, summarized in Table 2, demonstrate that the AEAD encapsulation imposes negligible latency, with the average encryption time consistently remaining under 3.5 microseconds (approx. 0.0035 ms) on the resource-constrained UPS edge nodes, proving that lightweight cryptography satisfies stringent smart grid constraints.

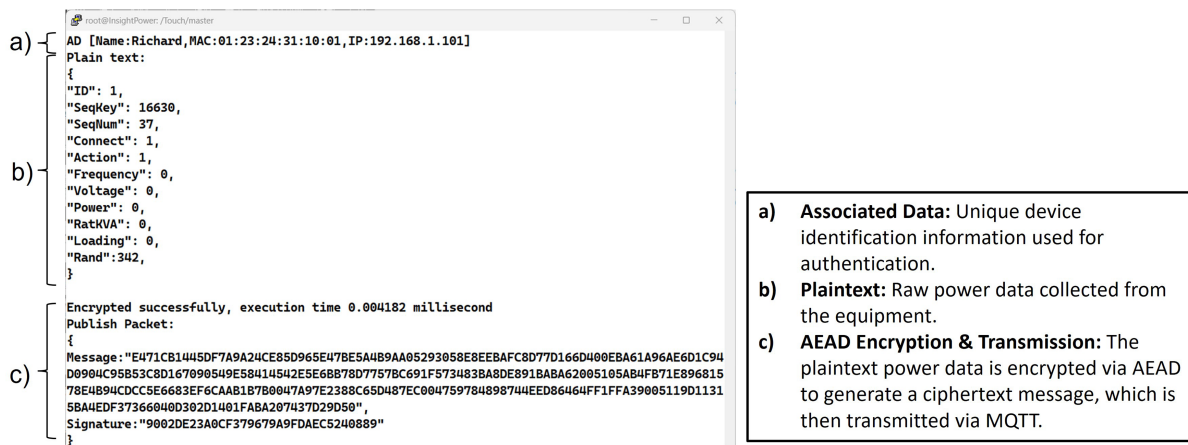


Figure 19: The AEAD encryption process: (a) Associated Data (AD) string containing unique device identification information (Name, MAC, and IP addresses). (b) Plaintext power data formatted as a JSON object, collected from the equipment. (c) Execution results indicating successful AEAD encryption, showing the generated ciphertext (Message), cryptographic tag/signature, and an execution time of 0.004182 milliseconds prior to MQTT transmission.

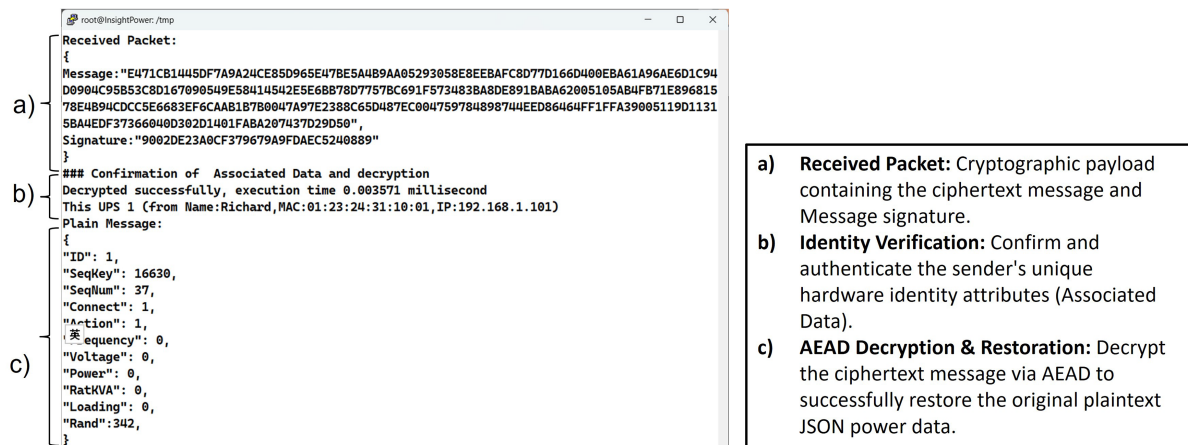


Figure 20: AEAD decryption process: (a) received encrypted packet containing the ciphertext (Message) and cryptographic tag (Signature), (b) successful confirmation of the Associated Data (AD) and decryption results, identifying the source device as UPS 1 (with matching Name, MAC, and IP) with an execution time of 0.003571 milliseconds. (c) Successfully restored plaintext power data formatted as a JSON object.

Table 2: Statistical summary of cryptographic execution time (ms) over 10,000 Iterations.

Processing Node	Mean $\pm$ Std. Dev. (ms)	Maximum (ms)	Minimum (ms)
UPS 1	0.00335 $\pm$ 0.00842	0.71672	0.00148
Server (for UPS 1)	0.00146 $\pm$ 0.00180	0.05177	0.00047
UPS 2	0.00334 $\pm$ 0.00359	0.13425	0.00151
Server (for UPS 2)	0.00164 $\pm$ 0.00342	0.23185	0.00049
UPS 3	0.00314 $\pm$ 0.00370	0.21452	0.00150
Server (for UPS 3)	0.00147 $\pm$ 0.00284	0.19773	0.00046

Furthermore, binding physical hardware attributes to cryptographic verification ensures the confidentiality and integrity of each telemetry payload, effectively preventing unauthorized node impersonation. These secure, identity-bound high-fidelity data payloads establish the foundation for physical power scheduling and directly trigger the automated control commands in the next phase (Section 5.3).

5.3 Power Distribution

Leveraging Section 5.2's secure inputs, the server executes the microgrid's automated dispatch logic within a zero-trust network. As demonstrated in Fig. 21(1), the backup mechanism is not triggered when the combined load of UPS1 and UPS2 remains below the 100% threshold. However, once the total load exceeds 100%, an encrypted "Turn ON" command is immediately issued to UPS3, as shown in Fig. 21(2). As depicted in Fig. 21(3), upon activation, UPS3 publishes its updated telemetry via MQTT, changing its status to "Turn ON"; furthermore, although the dynamic total load drops to 130%, it still does not satisfy the shutdown threshold ($\text{Load1} + \text{Load2} \leq 60\%$), thereby keeping UPS3 active. Conversely, when the total load falls to 50% as illustrated in Fig. 21(4), the server determines that the backup operation should cease and updates UPS3's status to "Turn OFF".

Furthermore, Fig. 22 confirms that the power backup system successfully receives, decrypts, and executes these remote control commands, transitioning from power output initiation as expected. For smart grid defense, these empirical outcomes validate that purely software-driven cryptographic protection seamlessly extends to physical microgrid actuation. This demonstrates the system's capacity to accurately, reliably, and efficiently execute remote operations under dynamic power conditions, supporting its deployment in critical real-world infrastructure. Having verified control loop efficacy, the final evaluation step (Section 5.4) systematically assesses the framework's resilience against active adversarial cyber-attacks.

<pre> ===== UPS[1] ===== Life Time : 110 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 112.4 V Output Power : 10000 W Output Load : 50 % ===== UPS[2] ===== Life Time : 110 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 60.2 Hz Output Volt. : 111.3 V Output Power : 8000 W Output Load : 40 % ===== UPS[3] ===== Life Time : 110 Second Connect Status: Link(1) Action : Turn OFF(0) Rating VA : 20000 VA Output Freq. : 0.0 Hz Output Volt. : 0.0 V Output Power : 0 W Output Load : 0 % </pre>	<pre> ===== UPS[1] ===== Life Time : 116 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 112.4 V Output Power : 12000 W Output Load : 60 % ===== UPS[2] ===== Life Time : 116 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 60.2 Hz Output Volt. : 111.3 V Output Power : 10000 W Output Load : 50 % ===== UPS[3] ===== Life Time : 116 Second Connect Status: Link(1) Action : Turn OFF(0) Rating VA : 20000 VA Output Freq. : 0.0 Hz Output Volt. : 0.0 V Output Power : 0 W Output Load : 0 % [Server] Need To Turn On Backup UPS </pre>	<pre> ===== UPS[1] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 112.4 V Output Power : 12000 W Output Load : 60 % ===== UPS[2] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 60.2 Hz Output Volt. : 111.3 V Output Power : 14000 W Output Load : 70 % ===== UPS[3] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 111.3 V Output Power : 16000 W Output Load : 80 % </pre>	<pre> ===== UPS[1] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 112.4 V Output Power : 4000 W Output Load : 20 % ===== UPS[2] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 60.2 Hz Output Volt. : 111.3 V Output Power : 6000 W Output Load : 30 % ===== UPS[3] ===== Life Time : 120 Second Connect Status: Link(1) Action : Turn ON(1) Rating VA : 20000 VA Output Freq. : 59.9 Hz Output Volt. : 111.3 V Output Power : 16000 W Output Load : 80 % [Server] Need To Turn Off Backup UPS </pre>
(1) Normal Operation	(2) Overload Trigger	(3) Backup Active	(4) Shutdown Threshold Met

Figure 21: The automated power distribution and backup mechanism: (1) Normal Operation: the combined load of UPS1 and UPS2 is below 100% ($50\% + 40\% = 90\%$), meaning the backup mechanism is not triggered, and UPS3 remains “Turn OFF” (2) overload trigger: the combined load exceeds the 100% threshold ($60\% + 50\% = 110\%$), and an encrypted command is immediately issued to trigger UPS3 (“Need To Turn On Backup UPS”). (3) Backup Active: UPS3 is activated and changes its status to “Turn ON”; although the total load drops to 130% ($60\% + 70\% = 130\%$), it remains active as it does not satisfy the shutdown threshold. (4) Shutdown Threshold Met: the total load falls below the 60% threshold ($20\% + 30\% = 50\%$), where the server determines the backup operation should cease and updates UPS3’s status to “Turn OFF”

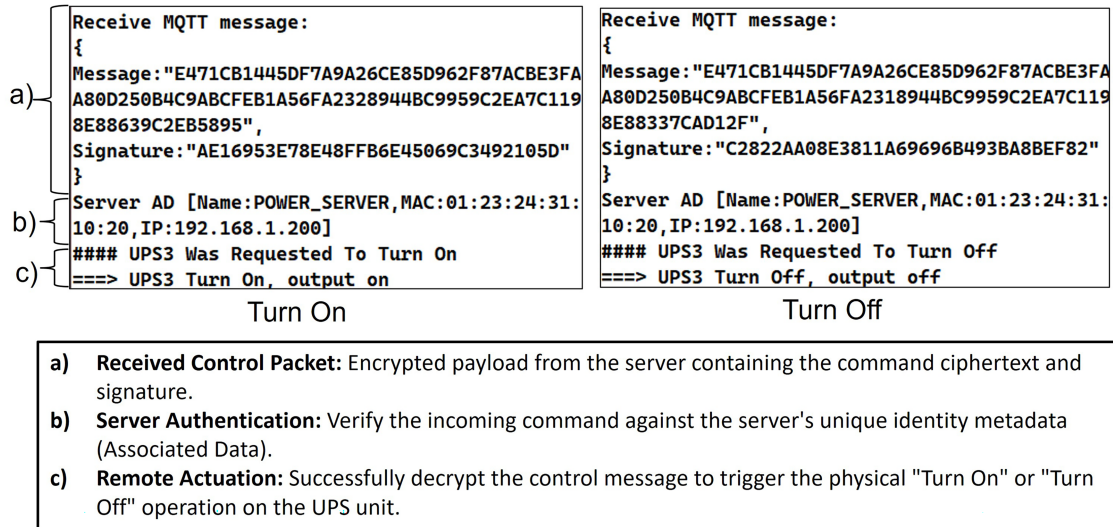


Figure 22: Power backup system operation results, the left panel illustrates the “Turn On” sequence, and the right panel illustrates the “Turn Off” sequence: (a) received encrypted MQTT packets containing the control ciphertext (Message) and authentication tag (Signature) from the server; (b) local verification utilizing the server’s unique identity attributes as Associated Data (AD); (c) successful de-encapsulation and physical actuation, transitioning the UPS3 output status accordingly.

5.4 Cybersecurity Verification

Based on the STRIDE threat modeling shown in Fig. 16, this phase evaluates the architecture's overall adversarial resilience by integrating the dynamic key generation from Section 5.1, the device identity binding from Section 5.2, and the secure control loop from Section 5.3. Comprehensive defensive testing was conducted, with terminal logs confirming the system's capability to intercept multi-dimensional, real-world cyber threats through a rigorous defense-in-depth strategy:

- **Spoofing:** Fig. 23 illustrates the system's ability to identify attempts to publish unauthorized data. In this scenario, a simulated attacker (Hacker) publishes related data to the ups/Value topic on the MQTT Broker using an information that is not in the predefined allowlist. Once the server receives the packet, it verifies the source identity and attempts to decrypt the data using the allowlist-based parameters. If both the identity check and decryption fail, the terminal log shows the system immediately classifies and rejects the packet as an illegal network transmission, demonstrating robustness against unauthorized messages.

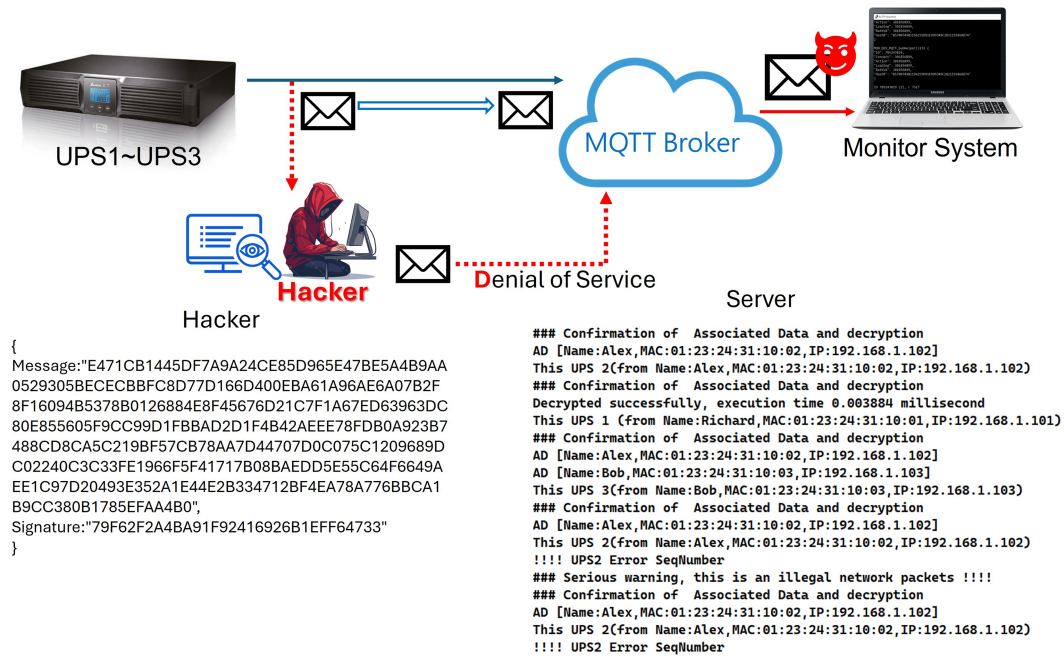


Figure 23: Spoofing: unauthorized.

- **Tampering:** Fig. 24 illustrates the system's capability to detect packet tampering after legitimate data transmission. After a valid power information packet is sent by the allowlist device UPS3, the attacker captures the transmission and modifies the message payload or the associated authentication tag before replaying the forged packet. Lacking the correct session keys and associated data, the server successfully detects and drops the invalid packet because the message integrity or authentication tag does not match.
- **Repudiation:** The integration of AEAD allows UPS3 to perform origin authentication for each received signal. This mechanism ensures that the system can definitively identify the server as the sender, satisfying the non-repudiation security criteria for power control commands and guaranteeing command authenticity. Fig. 25 demonstrates the experimental results, confirming the system's ability to verify command origin and prevent denial of action.

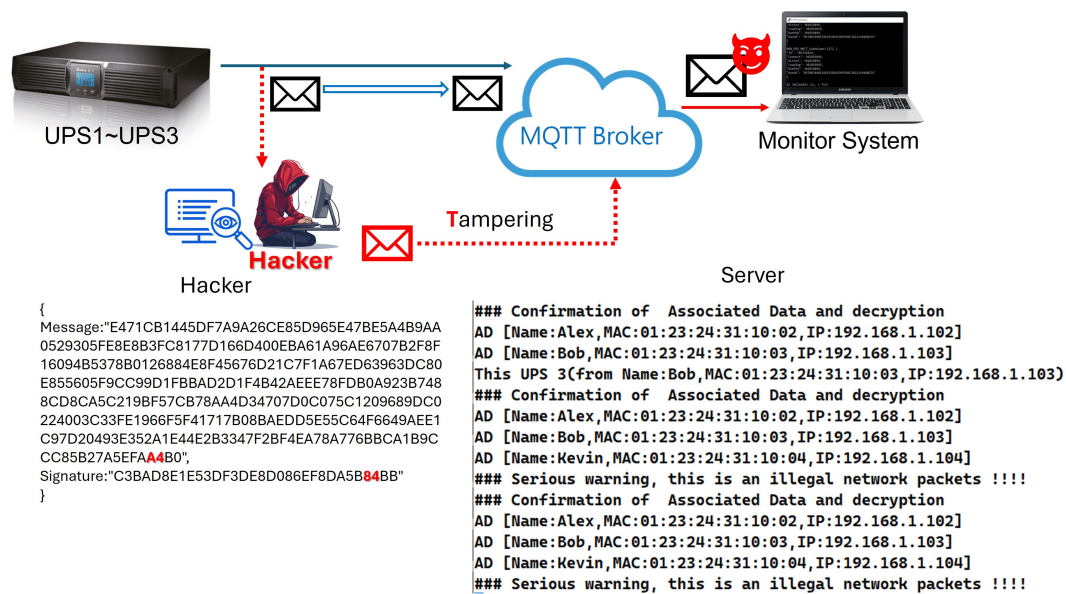


Figure 24: Tampering: modifies the message payload.

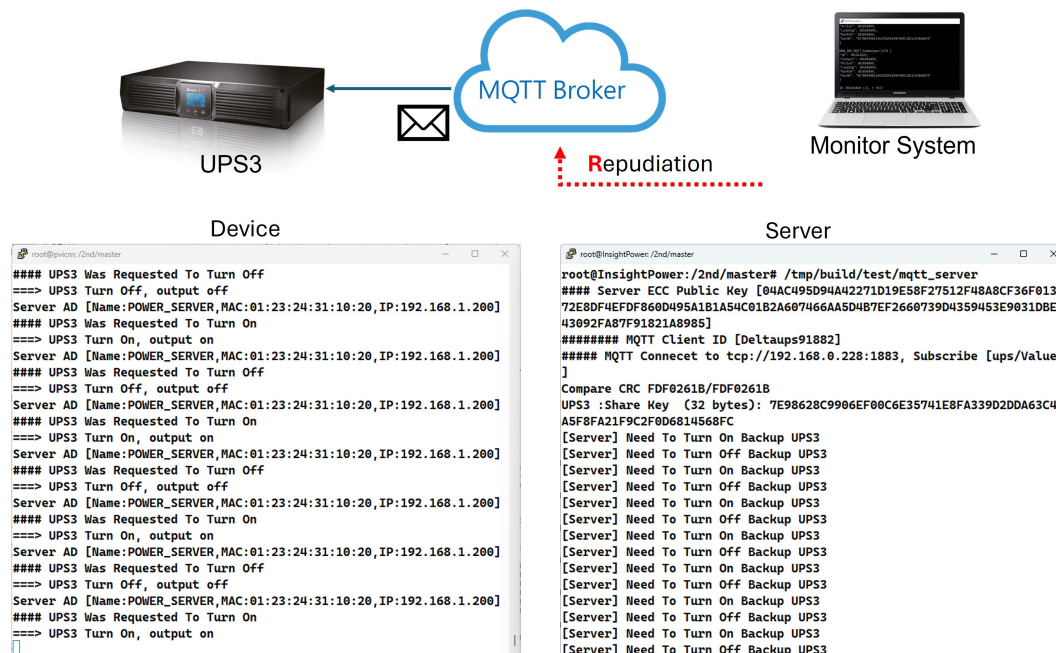


Figure 25: Repudiation: knowing who control.

- **Information Disclosure:**As shown in Fig. 26, the implementation of AEAD ChaCha20 encryption effectively secures smart grid communications. The mechanism ensures that even in the event of packet interception over the network, the sensitive telemetry and control information remains entirely encrypted as ciphertext and inaccessible to malicious actors, maintaining high levels of confidentiality.

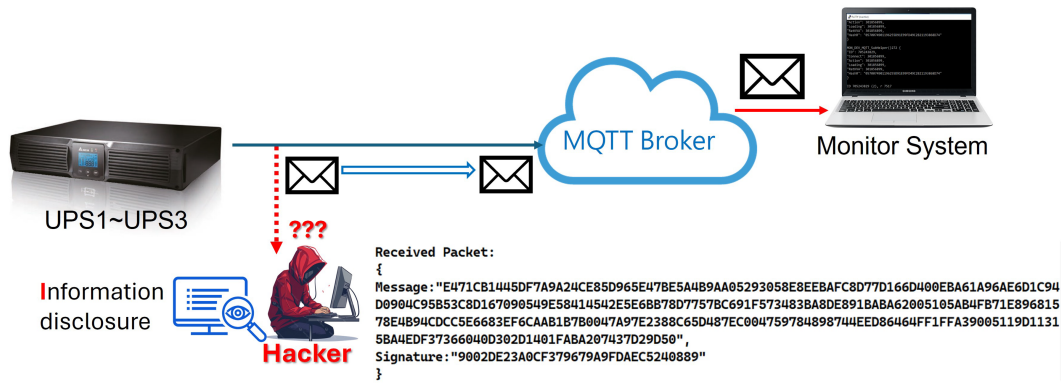


Figure 26: Information disclosure: SG information are transmitted via encryption.

- **Denial of Service (DoS):** Fig. 27 illustrates the system's response to a replay attack scenario. In this test, the legitimate UPS2 device first transmits valid power information. The attacker then captures the original packet and attempts to perform a replay attack by resending the previously recorded data. After the Monitor Server receives the replayed packet, it detects the sequence number anomaly and successfully blocks the duplicated transmission, validating the sequence-based mitigation mechanism.

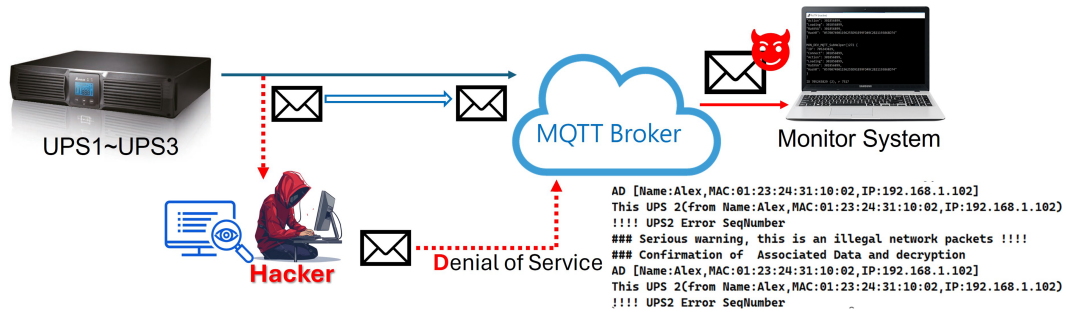


Figure 27: Denial of service: replay packet.

- **Elevation of Privilege:** Fig. 28 evaluates the authorization framework against privilege escalation attempts. While the attacker may successfully authenticate or forge a connection as a low-privilege entity (e.g., POWER_GUEST), the authorization module strictly enforces role-based permission boundaries. As demonstrated in the terminal log, when an unauthorized request is submitted, the system explicitly rejects execution with an “Insufficient permissions” error, ensuring that critical control commands (e.g., ‘Turn-OFF’) remain restricted to administrative roles (e.g., POWER_SERVER).

In summary, the four-phase empirical evaluations on a physical testbed thoroughly validate the performance and security of the proposed architecture.

The ECDH implementation ensures secure initialization by establishing independent dynamic 32-byte session keys, eliminating static key risks. Building on this, the ChaCha20-Poly1305 AEAD system provides robust data-plane protection by binding hardware metadata to telemetry payloads. Performance scaling over 10,000 iterations proves that the lightweight cryptographic overhead is negligible, consistently requiring less than 3.5 μ s on resource-constrained edge nodes to satisfy strict real-time constraints.

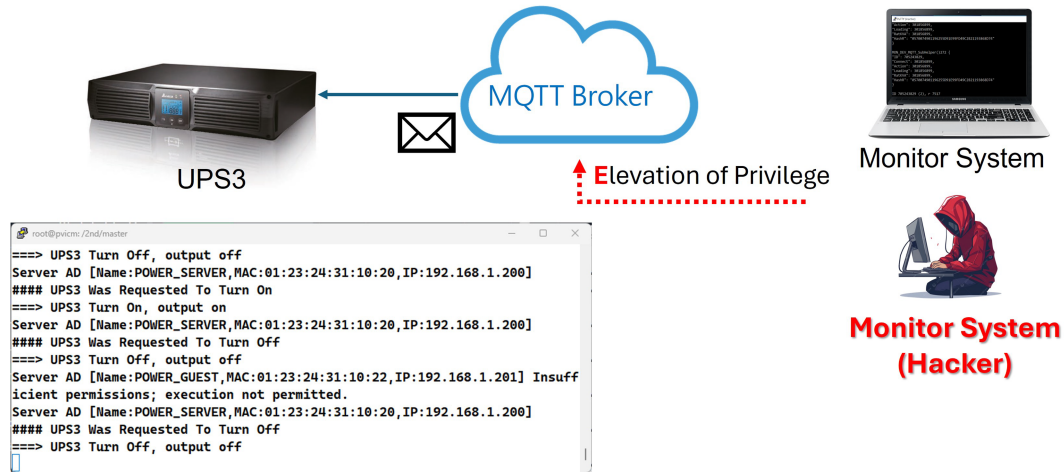


Figure 28: Elevation of privilege: insufficient permissions.

Furthermore, the cyber-physical control loop successfully extends software-driven cryptographic protection to physical microgrid actuation, achieving deterministic, automated power distribution under dynamic loads. Finally, systematic adversarial testing aligned with the STRIDE model confirms the architecture's defense-in-depth capability. Experimental results verify that the system effectively mitigates spoofing, tampering, unauthorized disclosure, replay attacks, and privilege elevation, ensuring comprehensive data confidentiality, operational integrity, and non-repudiation for critical smart grid infrastructure.

6 Conclusion

This research addresses the critical security and performance trade-offs in modern smart grid telemetry by introducing the Secure Smart Grid Power Monitoring and Sharing System (SSGPMSS). By leveraging the asynchronous nature of the MQTT protocol as a communication backbone and seamlessly embedding advanced cryptographic primitives at the application layer, this study provides a viable, secure, and highly efficient framework for contemporary microgrid environments.

The key achievements and findings of this study are summarized as follows:

- **Lightweight High-Assurance Protection:** By incorporating the ChaCha20-Poly1305 AEAD scheme, the system successfully guarantees simultaneous data confidentiality, payload integrity, and origin authenticity. Empirical evaluations over 10,000 iterations demonstrate an exceptionally low cryptographic execution overhead, consistently maintaining an average encryption latency of less than $3.5 \mu\text{s}$ ($\approx 0.0035 \text{ ms}$) on resource-constrained UPS edge nodes without requiring dedicated hardware acceleration.
- **Context-Bound Key Agreement:** To mitigate the inherent vulnerabilities of static key leakage and the lack of node authentication in plain Elliptic Curve Diffie-Hellman (ECDH) protocols, we introduced an implicit authentication mechanism. By tightly binding hardware-specific metadata (such as MAC and IP addresses) within the Additional Authenticated Data (AAD) field, the dynamic generation of unique, device-specific 32-byte session keys structurally eliminates Man-in-the-Middle (MITM) and node impersonation vectors.
- **Cyber-Physical Resilience:** The physical smart grid testbed implementation validates that software-driven cryptographic protection successfully extends to physical microgrid actuation. The system

maintains real-time responsiveness (with a control loop latency well below 1 ms), executing deterministic automated power distribution and backup triggering under dynamic load variations while comprehensively neutralizing multi-dimensional cyber threats defined by the STRIDE model (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege).

Despite its demonstrated efficacy, a remaining shortcoming of the current architecture lies in its reliance on a pre-established trust anchor or initialization allowlist for initial public-key verification during the ephemeral ECDH exchange, which poses efficiency challenges in massive decentralized deployments. Additionally, as an application-layer software solution, the framework remains potentially vulnerable to localized physical side-channel attacks or direct memory injection if an adversary gains local access to the edge devices.

To guarantee long-term operational resilience and scalability, our future works will focus on the following directions:

- **Post-Quantum Cryptographic Migration:** We plan to upgrade the public-key infrastructure by integrating Post-Quantum Cryptography (PQC) key encapsulation mechanisms (such as Kyber) and digital signature algorithms (such as Dilithium) to safeguard the smart grid infrastructure against emerging quantum computing threats.
- **Hardware-Hardened Edge Isolation:** Future implementations will explore the incorporation of Trusted Execution Environments (TEEs), such as ARM TrustZone, at the edge tier. This hardware-software co-design will isolate cryptographic operations, secure session keys at rest, and protect critical firmware routines from localized physical tampering or side-channel profiling.

Acknowledgement: The authors would like to express their gratitude to National Cheng Kung University for providing the academic resources that supported the completion of this study. AI tools were used only for language polishing and grammar correction.

Funding Statement: This work was partially supported by National Science and Technology Council (NSTC), Taiwan under the Grant NSTC 114-2221-E-006-171.

Author Contributions: Chung-Pao Lin: Investigation, Data Curation, Problem Identification, System Architecture Design and Verification, Writing—Original Draft, Review and Editing, Visualization. Yi-You Hou and Teh-Lu Liao: Writing—Review and Editing, Supervision, Funding. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets generated and analyzed during the current study, including cryptographic execution samples and latency measurements, are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li F, Qin J, Zheng WX. Distributed Q-learning-based online optimization algorithm for unit commitment and dispatch in smart grid. *IEEE Trans Cybern.* 2019;50(9):4146–56. doi:10.1109/TCYB.2019.2921475.
2. Gunduz MZ, Das R. Cyber-security on smart grid: threats and potential solutions. *Comput Netw.* 2020;169(11):107094. doi:10.1016/j.comnet.2019.107094.
3. Liao TL, Lin CP, Hou YY. Secure smart grid power monitoring and simulations based on homomorphic encryption. *IEEE Access.* 2024;12:122820–7. doi:10.1109/access.2024.3453998.

4. Mohammed A, George G. Vulnerabilities and strategies of cybersecurity in smart grid—evaluation and review. In: Proceedings of the 2022 3rd International Conference on Smart Grid and Renewable Energy (SGRE); 2022 Mar 20–22; Doha, Qatar. p. 1–6. doi:10.1109/sgre53517.2022.9774038.
5. Ruj S, Nayak A. A decentralized security framework for data aggregation and access control in smart grids. *IEEE Trans Smart Grid*. 2013;4(1):196–205. doi:10.1109/tsg.2012.2224389.
6. Meng F, Ye W. Summary of research on security and privacy of smart grid. In: Proceedings of the 2020 International Conference on Computer Communication and Network Security (CCNS); 2020 Aug 21–23; Xi'an, China. p. 39–42. doi:10.1109/ccns50731.2020.00017.
7. Chen W, Liu L, Liu GP. Privacy-preserving distributed economic dispatch of microgrids: a dynamic quantization-based consensus scheme with homomorphic encryption. *IEEE Trans Smart Grid*. 2023;14(1):701–13. doi:10.1109/tsg.2022.3189665.
8. Gunduz MZ, Das R. Analysis of cyber-attacks on smart grid applications. In: Proceedings of the 2018 International Conference on Artificial Intelligence and Data Processing (IDAP); 2018 Sep 28–30; Malatya, Turkey. p. 1–5. doi:10.1109/idap.2018.8620728.
9. Chen L, Liu J, Ha W. Cloud service risk in the smart grid. In: Proceedings of the 2018 14th International Conference on Computational Intelligence and Security (CIS); 2018 Nov 16–19; Hangzhou, China. p. 242–4. doi:10.1109/cis2018.2018.00060.
10. Rahman MS, Khalil I, Moustafa N, Kalapaaking AP, Bouras A. A blockchain-enabled privacy-preserving verifiable query framework for securing cloud-assisted industrial Internet of Things systems. *IEEE Trans Ind Inf*. 2022;18(7):5007–17. doi:10.1109/tii.2021.3105527.
11. Alabdulatif A, Kumaraage H, Khalil I, Atiquzzaman M, Yi X. Privacy-preserving cloud-based billing with lightweight homomorphic encryption for sensor-enabled smart grid infrastructure. *IET Wirel Sens Syst*. 2017;7(6):182–90. doi:10.1049/iet-wss.2017.0061.
12. Sikic L, Jankovic J, Afric P, Silic M, Ilic Z, Pandzic H, et al. A comparison of application layer communication protocols in IoT-enabled smart grid. In: Proceedings of the 2020 International Symposium ELMAR; 2020 Sep 14–15; Zadar, Croatia. p. 83–6. doi:10.1109/elmar49956.2020.9219030.
13. Islam MM, Paul S, Haque MM. Reducing network overhead of IoTDTLS protocol employing ChaCha20 and Poly1305. In: Proceedings of the 2017 20th International Conference of Computer and Information Technology (ICCIT); 2017 Dec 22–24; Dhaka, Bangladesh. p. 1–7. doi:10.1109/iccitechn.2017.8281857.
14. Aoufi S, Derhab A, Guerroumi M. Survey of false data injection in smart power grid: attacks, countermeasures and challenges. *J Inf Secur Appl*. 2020;54(5):102518. doi:10.1016/j.jisa.2020.102518.
15. Paula AdO, Meneguette RI, Giuntini FT, Peixoto MLM, Goncalves VP, Rocha Filho GP. STRAYER: a smart grid adapted automation architecture against cyberattacks. *J Inf Secur Appl*. 2022;67:103195. doi:10.1016/j.jisa.2022.103195.
16. Wen M, Zhang Y, Zhang P, Chen L. IDS-DWKAFL: an intrusion detection scheme based on dynamic weighted K-asynchronous federated learning for smart grid. *J Inf Secur Appl*. 2025;89(3):103993. doi:10.1016/j.jisa.2025.103993.
17. Ding Y, Wang B, Wang Y, Zhang K, Wang H. Secure metering data aggregation with batch verification in industrial smart grid. *IEEE Trans Ind Inf*. 2020;16(10):6607–16. doi:10.1109/tii.2020.2965578.
18. Zhang X, Deng X, Zhang H, Zhang Y, Xue J. Fine-grained encrypted data aggregation mechanism with fault tolerance in edge-assisted smart grids. *J Inf Secur Appl*. 2024;86(1):103888. doi:10.1016/j.jisa.2024.103888.
19. Bai S, Yang G, Shi J, Liu G, Min Z. Privacy-preserving oriented floating-point number fully homomorphic encryption scheme. *Secur Commun Netw*. 2018;2018(1):2363928. doi:10.1155/2018/2363928.
20. Chang Y, Li J, Li W. 2D2PS: a demand-driven privacy-preserving scheme for anonymous data sharing in smart grids. *J Inf Secur Appl*. 2023;74(3):103466. doi:10.1016/j.jisa.2023.103466.
21. Garg S, Kaur K, Kaddoum G, Rodrigues JJPC, Guizani M. Secure and lightweight authentication scheme for smart metering infrastructure in smart grid. *IEEE Trans Ind Inf*. 2020;16(5):3548–57. doi:10.1109/tii.2019.2944880.
22. Chaudhry SA, Nebhan J, Yahya K, Al-Turjman F. A privacy enhanced authentication scheme for securing smart grid infrastructure. *IEEE Trans Ind Inf*. 2022;18(7):5000–6. doi:10.1109/tii.2021.3119685.

23. Ponnuru RB, Kumar SAP, Azab M, Palaniswamy B, Alavalapati GR. Robust authentication and key agreement protocol for smart microgrid environment. *J Inf Secur Appl.* 2025;94(2):104202. doi:10.1016/j.jisa.2025.104202.
24. Liu F, Sun Y, Liu Z, Yan Y, Li H, Han Y, et al. Lightweight batch authentication protocol for bus-NB-IoT hierarchical network in smart grid using physically unclonable function. In: *Proceedings of the 2023 IEEE 9th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*; 2023 May 6–8; New York, NY, USA. p. 107–14. doi:10.1109/bigdatasecurity-hpsc-ids58521.2023.00029.
25. Cardona-López MA, Chimal-Eguía JC, Silva-García VM, Flores-Carapia R. Key exchange with diffie-Hellman protocol and composite hash-functions. In: *Proceedings of the 2024 12th International Symposium on Digital Forensics and Security (ISDFS)*; 2024 Apr 29–30; San Antonio, TX, USA. p. 1–6. doi:10.1109/isdfs60797.2024.10527265.
26. Van Nghi N, Trung DQ, Thanh DT. Two novel variants of ECDH key exchange protocol using schnorr zero-knowledge proof. In: *Proceedings of the 2023 15th International Conference on Knowledge and Systems Engineering (KSE)*; 2023 Oct 18–20; Hanoi, Vietnam. p. 1–4. doi:10.1109/kse59128.2023.10299420.
27. Bernstein DJ. ChaCha, a variant of Salsa20; 2008 [cited 2026 Jan 1]. Available from: <http://cr.yp.to/chacha/chacha-20080120.pdf>.
28. Bernstein DJ. The Poly1305-AES message-authentication code. In: *Fast software encryption*. Berlin/Heidelberg, Germany: Springer; 2005. p. 32–49. doi:10.1007/11502760_3.
29. Nir Y, Langley A. ChaCha20 and Poly1305 for IETF protocols. IETF; 2015 [cited 2026 Jan 1]. Available from: <https://www.rfc-editor.org/rfc/rfc8439>.
30. Serrano R, Duran C, Sarmiento M, Pham CK, Hoang TT. ChaCha20-Poly1305 authenticated encryption with additional data for transport layer security 1.3. *Cryptography.* 2022;6(2):30. doi:10.3390/cryptography6020030.
31. Nguyen ND, Bui DH, Tran XT. A Lightweight AEAD encryption core to secure IoT applications. In: *Proceedings of the 2020 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*; 2020 Dec 8–10; Ha Long, Vietnam. p. 35–8. doi:10.1109/apccas50809.2020.9301683.
32. Fernandes JCV, da Rocha H, Espírito-Santo A. Encryption and authentication in smart transducers implemented in RISC-V softcore processors. In: *Proceedings of the 2023 IEEE International Conference on Industrial Technology (ICIT)*; 2023 Apr 4–6; Orlando, FL, USA. p. 1–6. doi:10.1109/icit58465.2023.10143043.
33. Serrano R, Duran C, Hoang TT, Sarmiento M, Tsukamoto A, Suzuki K, et al. ChaCha20-Poly1305 crypto core compatible with transport layer security 1.3. In: *Proceedings of the 2021 18th International SoC Design Conference (ISOCC)*; 2021 Oct 6–9; Jeju Island, Republic of Korea. p. 17–8. doi:10.1109/isocc53507.2021.9614016.
34. Ahmad Y, Singh KK. Spartan-7 FPGA based implementation of ChaCha20 accelerator for RISC-V systems. In: *Proceedings of the 2026 2nd International Conference on Cognitive Computing in Engineering, Communications, Sciences and Biomedical Health Informatics (IC3ECSBHI)*; 2026 Feb 12–14; Greater Noida, India. p. 490–5. doi:10.1109/ic3ecsbhi67834.2026.11469110.
35. De Santis F, Schauer A, Sigl G. ChaCha20-Poly1305 authenticated encryption for high-speed embedded IoT applications. In: *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017; 2017 Mar 27–31; Lausanne, Switzerland. p. 692–7. doi:10.23919/date.2017.7927078.