Computer Modeling in Engineering & Sciences

Tech Science Press

Doi:10.32604/cmes.2025.064997

ARTICLE

Port-Based Pre-Authentication Message Transmission Scheme

Sunghyun Yu and Yoojae Won*

Department of Computer Science and Engineering, Chungnam National University, Daejeon, 34134, Republic of Korea

*Corresponding Author: Yoojae Won. Email: yjwon@cnu.ac.kr

Received: 28 February 2025; Accepted: 21 May 2025; Published: 30 June 2025

ABSTRACT: Pre-Authentication and Post-Connection (PAPC) plays a crucial role in realizing the Zero Trust security model by ensuring that access to network resources is granted only after successful authentication. While earlier approaches such as Port Knocking (PK) and Single Packet Authorization (SPA) introduced pre-authentication concepts, they suffer from limitations including plaintext communication, protocol dependency, reliance on dedicated clients, and inefficiency under modern network conditions. These constraints hinder their applicability in emerging distributed and resource-constrained environments such as AIoT and browser-based systems. To address these challenges, this study proposes a novel port-sequence-based PAPC scheme structured as a modular model comprising a client, server, and ephemeral Key Management System (KMS). The system employs the Advanced Encryption Standard (AES-128) to protect message confidentiality and uses a Hash-Based Message Authentication Code (HMAC-SHA256) to ensure integrity. Authentication messages are securely fragmented and mapped to destination port numbers using a signature-based avoidance algorithm, which prevents collisions with unsafe or reserved port ranges. The server observes incoming port sequences, retrieves the necessary keys from the KMS, reconstructs and verifies the encrypted data, and conditionally updates firewall policies. Unlike SPA, which requires decrypting all incoming payloads and imposes server-side overhead, the proposed system verifies only port-derived fragments, significantly reducing computational burden. Furthermore, it eliminates the need for raw socket access or custom clients, supporting browser-based operation and enabling protocol-independent deployment. Through a functional web-based prototype and emulated testing, the system achieved an F1-score exceeding 95% in detecting unauthorized access while maintaining low resource overhead. Although port sequence generation introduces some client-side cost, it remains lightweight and scalable. By tightly integrating lightweight cryptographic algorithms with a transport-layer communication model, this work presents a conceptually validated architecture that contributes a novel direction for interoperable and scalable Zero Trust enforcement in future network ecosystems.

KEYWORDS: Port knocking; single packet authorization; pre-authentication; zero trust; network security; HMAC; authentication; AIoT security

1 Introduction

As digital transformation accelerates, the proliferation of cloud infrastructures, remote work environments, and BYOD (Bring Your Own Device) policies has blurred the traditional boundaries of enterprise networks. This shift has exposed fundamental limitations in perimeter-based security models, which implicitly trust users and devices within the internal network. In response, the Zero Trust security model has emerged as a critical paradigm shift, enforcing the principle of "Never Trust, Always Verify" and requiring continuous authentication regardless of network location.

To implement Zero Trust, the Pre-Authentication and Post-Connection (PAPC) mechanism plays a vital role. Unlike traditional authentication methods where authentication occurs after a connection is established, PAPC ensures that only authenticated entities can initiate communication, thereby minimizing attack surfaces. Two primary implementations of PAPC are Port Knocking (PK) and Single Packet Authorization (SPA), both of which aim to conceal services and enforce access control at the network level.

Port Knocking (PK) allows clients to sequentially access a predefined series of ports as a form of authentication. This method is lightweight and does not require dedicated client software, making it suitable for resource-constrained environments. However, its lack of encryption exposes it to sniffing, spoofing, and replay attacks, while its reliance on port order introduces latency and usability issues [1].

SPA improves upon PK by embedding encrypted authentication data into a single UDP packet, which is verified by the server before allowing access. While SPA enhances security through encryption and Hash-based Message Authentication Code (HMAC) validation, it presents several limitations. It requires raw socket access to craft user datagram protocol (UDP) packets, making it incompatible with typical web clients such as browsers.

Moreover, SPA imposes a high computational burden on servers that must analyze all incoming UDP packets, making it unsuitable for bandwidth- and resource-constrained environments such as Artificial Intelligence of Things (AIoT) devices [2]. SPA's reliance on a dedicated client and limited protocol flexibility further hinder its deployment in diverse or browser-centric scenarios.

1.1 Research Objectives and Contributions

To overcome the limitations of existing PAPC methods, we propose a novel port-based pre-authentication and post-connection protocol that is protocol-independent and clientless. Our method transmits encrypted authentication information using a sequence of port numbers, eliminating the need for packet payloads or dedicated clients. This enables lightweight, secure authentication even in web and AIoT environments, where raw socket access is restricted or infeasible.

The proposed method fragments authentication data, encrypts it using AES, applies HMAC for integrity, and maps it to port numbers that are transmitted in a specific sequence. The server reconstructs the original authentication message by observing port access attempts, and performs verification using pre-registered keys retrieved from a Key Management System (KMS). Nonce and timestamp mechanisms are used to prevent replay attacks. This approach supports browser-based environments and avoids unsafe/well-known ports using a dynamic signature-based mapping algorithm.

The main contributions of this study are as follows. First, we present a protocol-independent and clientless pre-authentication scheme that is compatible with both TCP/UDP environments and web browsers.

Second, we design a port-sequence based fragmentation and reconstruction mechanism incorporating sequence numbers and signature verification to ensure both message integrity and correct delivery order.

Third, we integrate AES encryption and HMAC validation, along with replay protection mechanisms using nonce and timestamp techniques.

Fourth, we propose a Key Management System (KMS)-based ephemeral key handling strategy, which allows secure and temporary storage and delivery of cryptographic keys without relying on persistent storage.

Finally, we implement and validate a fully functional web-based prototype, demonstrating the practicality and feasibility of the proposed approach under real-world constraints.

This paper is organized as follows. Section 2 surveys related work on Port Knocking (PK), Single Packet Authorization (SPA), and Zero Trust-based authentication. Section 3 describes the proposed port-based

Pre-Authentication and Post-Connection (PAPC) method. Section 4 presents the system implementation and integration details. Section 5 reports experimental results and provides a security analysis. Section 6 discusses the applicability and limitations of the proposed method. Finally, Section 7 concludes the paper and outlines future research directions.

2 Related Works

2.1 Relationship Between the Zero Trust Security Model and PAPC

With the rapid evolution of digital environments, network security threats have significantly increased. The perimeter-based security model has traditionally operated by clearly distinguishing between internal and external networks. However, the expansion of cloud computing, the rise of remote work environments, and the increase in insider threats have revealed the limitations of this approach [3–5].

To address these challenges, John Kindervag of Forrester Research proposed the Zero Trust security model in 2010 [6]. This model is based on the principle of "Never Trust, Always Verify", which enhances security policies by requiring continuous authentication and verification of all network activities [7]. Unlike traditional security models that rely on network boundaries, Zero Trust eliminates implicit trust and evaluates all access requests under the same stringent security criteria, regardless of user location.

Traditional network security methods clearly distinguish between internal and external users. However, Zero Trust continuously verifies all users, including those who have already been authenticated, to maintain security [8]. Even if a user is authenticated once, Zero Trust mandates continuous validation and restricts access based on the principle of least privilege, ensuring that users can only access the resources strictly necessary for their role [9–11].

The Zero Trust model differs from traditional security approaches through several core principles. First, it enforces verification of all network traffic, meaning that no user–whether internal or external–is implicitly trusted; all access requests must be verified in advance. Second, it applies the least privilege principle, whereby users are granted access only to the resources necessary for their roles. Third, access control is policy-based, allowing dynamic adjustment based on contextual policies and enabling continuous monitoring for abnormal activities. Finally, Zero Trust promotes network segmentation, enforcing security policies at the asset level to limit access to sensitive information and enhance data protection.

To implement these security principles, Zero Trust network architectures require the Pre-Authentication and Post-Connection (PAPC) mechanism. Unlike traditional methods that allow users to connect before authentication, PAPC enforces authentication before network access is granted. This approach effectively eliminates unauthorized access attempts [12,13].

In a Zero Trust environment, PAPC prevents unauthorized users from connecting to the network, allowing access only after authentication. By ensuring that even authenticated users undergo continuous validation, Zero Trust mitigates security risks associated with perimeter-based models and enforces strict access controls.

Furthermore, Zero Trust is designed around security policies that focus on endpoint-level protection and data-centric security rather than network boundaries. Consequently, PAPC plays a crucial role in minimizing attack surfaces and reducing unnecessary network exposure, thereby maximizing security effectiveness.

2.2 PAPC Mechanism and Limitations of Existing Approaches

With the widespread adoption of the Zero Trust security model, the Connection Before Authentication (CBA) approach has become increasingly vulnerable to security threats.

In traditional network security models, the client first connects to the network before authentication is performed. However, this approach is susceptible to various attacks, including port scanning, sniffing attacks, and unauthorized access attempts leading to availability attacks such as DoS and DDoS.

To mitigate these vulnerabilities, the Pre-Authentication and Post-Connection (PAPC) mechanism has been developed. Two representative implementations of this approach are Port Knocking (PK) and Single Packet Authorization (SPA).

2.2.1 Port Knocking

Port Knocking (PK) was first introduced by Martin Krzywinski in 2003 as a pre-authentication and post-connection technique that utilized a sequence of port accesses to stealthily authenticate clients without exposing open ports [14].

His key contribution was the conceptualization of a firewall-triggered authentication method, allowing closed systems to remain invisible until legitimate users completed the correct port sequence. However, PK lacked payload encryption and was highly susceptible to sniffing, spoofing, and replay attacks, making it vulnerable under modern threat landscapes.

The client attempts authentication by accessing a predefined sequence of ports in a specific order, and the server (or firewall) determines whether to grant access based on this sequence.

For example, if a client accesses ports 80, 443, and 8080 in sequence, the server compares the incoming sequence with a predefined pattern and grants authentication upon a match. Port Knocking is notable for its simple structure, as it can be easily implemented in resource-constrained environments, operates independently of TCP and UDP protocols, and does not require additional client-side software.

However, Port Knocking is vulnerable to sniffing and spoofing attacks, and its buffer used for sequence validation can be exploited for DoS attacks, among other security concerns. As shown in Table 1, it has several security weaknesses, and various methods have been studied to improve its robustness.

Table 1: Weaknesses of port-knocking on security and network performance

Weakness	Effect
Plain text port sequence	Vulnerability to traffic sniffing and scanning
Network Address Translate (NAT) Knock	Service exposure to users sharing a public IP
Denial of Service (DoS) knocks attacks	Knock sequence buffer overloading
Select the inappropriate range of port	Narrow range susceptible to traffic capture, wide range vulnerable to availability attacks
Out of order packet delivery	Potential for system malfunction
Lack of association between authentication and connection	Risk of additional transmission blocking and connection hijacking by users
The host behind the firewall	Authorized users may experience malfunctions due to insufficient privileges

Note: The table summarizes the security and performance weaknesses of port-knocking.

Khan et al. classified port-knocking authentication methods into two categories: Static Knock Sequence and Dynamic Knock Sequence techniques [1]. Their key contribution was to systematically formalize port-knocking variants based on sequence generation mechanisms, distinguishing between fixed and session-randomized approaches.

Static sequences involve pre-defined, unchanging port orders, while dynamic sequences utilize random number generators (RNGs) to assign new sequences for each session, enhancing resistance to replay attacks.

However, both techniques retained vulnerabilities such as susceptibility to packet sniffing (in static sequences) and potential compatibility issues with NAT traversal (in dynamic sequences), limiting their practical deployment in diverse network environments.

The static knock sequence method [14] predefines a port sequence for authentication, requiring the client to access these ports in a fixed order.

In contrast, the dynamic knock sequence method proposed by Isabel [15] employs a random number generator (RNG) on both the client and server sides to dynamically assign port sequence for each authentication session. This dynamic assignment enhances security by mitigating the risk of replay attacks and sequence sniffing, which are prevalent in static methods.

The key contribution of this approach lies in introducing session-specific randomness to the port sequence, thereby significantly increasing the difficulty for attackers to predict valid authentication patterns.

However, the method still faces practical challenges, particularly regarding compatibility with Network Address Translation (NAT) devices, where random ports may be remapped unpredictably, potentially disrupting the authentication process.

To enhance the security of port-knocking mechanisms, Vasserman et al. [16] proposed incorporating encryption into both static and dynamic knock sequences, leading to the development of Static Encrypted Knock Sequence and Dynamic Encrypted Knock Sequence techniques.

Their key contribution was introducing payload confidentiality to knock sequences, thereby preventing packet sniffing and spoofing attacks that plagued earlier plaintext implementations. By encrypting the sequence, these methods also strengthened resistance against replay attacks by generating unique sequences for each authentication session.

However, despite these improvements, encrypted knock sequences still faced limitations in environments with high packet loss or reordering, where maintaining correct decryption and sequence assembly could become challenging, potentially impacting reliability in real-world deployments.

To further enhance the security of port-knocking mechanisms, researchers proposed integrating supplementary authentication methods such as Short Message Service (SMS), One-Time Password (OTP) [17], and Mutual Authentication [18].

These enhancements contributed by introducing multi-factor authentication capabilities, thereby strengthening identity verification beyond mere possession of network credentials. Specifically, the use of OTPs enabled dynamic, time-sensitive validation, while Mutual Authentication schemes ensured that both client and server could verify each other's legitimacy before granting access.

However, despite these advances, these approaches introduced new operational complexities, such as dependency on external authentication servers (e.g., OTP servers) and potential availability concerns, which could limit autonomous operation in resource-constrained or isolated network environments.

However, even these improved port-knocking approaches retain certain vulnerabilities. For instance, static port sequence remain susceptible to sniffing attacks, while dynamic knock sequences may face compatibility issues in Network Address Translation (NAT) environments.

Moreover, they lack robust protection against DoS attacks, and authentication may fail if packet sequences arrive out of order. Notably, some authentication mechanisms rely on external authentication systems (e.g., OTP servers), introducing third-party dependency and making standalone operation difficult.

2.2.2 Authentication Information Transmission Using Port Space

In previous research on port-knocking mechanisms, deGraaf et al. [19] proposed an innovative approach that utilizes 16-bit destination port numbers as a medium for transmitting authentication information.

Their key contribution was the conceptual shift from simple port sequence monitoring to active use of the port number space as a covert data transmission channel, enabling servers to verify client authentication without requiring payload analysis. This approach opened a new direction for pre-authentication by leveraging the inherent properties of the transport layer, effectively addressing certain vulnerabilities of traditional Connection Before Authentication (CBA) models, such as payload exposure risks.

However, the study did not provide a comprehensive strategy for avoiding well-known and unsafe ports, raising concerns about potential conflicts with reserved services and susceptibility to security vulnerabilities, which could limit its real-world applicability.

The study by deGraaf et al. did not propose a comprehensive strategy for avoiding the use of well-known or unsafe ports, which could significantly limit the practical applicability of the approach. Well-known ports are reserved by the Internet Assigned Numbers Authority (IANA) for essential services such as Hypertext Transfer Protocol (HTTP, 80), File Transfer Protocol (FTP, 21), and Secure Shell (SSH, 22), and often require elevated privileges for access, introducing potential security and accessibility concerns [20,21].

Furthermore, certain port ranges, such as those associated with Telnet (23) and Network Basic Input/Output System (NetBIOS, 137–139), are known to facilitate malware propagation and expose systems to vulnerabilities. Utilizing such ports for authentication transmission without restriction could undermine the overall security model, highlighting an important limitation in the original design despite its conceptual contribution to covert authentication techniques.

Previous studies did not address these port utilization issues and focused solely on restricting conflicting port ranges. However, this constrained approach may hinder real-world applicability. To overcome these limitations, this study introduces a port mapping technique that avoids well-known and unsafe ports, thereby enhancing security.

Additionally, this method expands port numbers from being merely an authentication mechanism to serving as a message transmission channel. Furthermore, by leveraging the protocol-independent nature of port space-based authentication, this study addresses the limitations of previous research while improving applicability across diverse network environments.

2.2.3 Single Packet Authorization

Single Packet Authorization (SPA) was introduced by Michael Rash in 2006 as an enhanced alternative to Port Knocking (PK), aiming to resolve its inherent inefficiencies and security limitations [22].

Rash's key contribution was the design of a mechanism that encapsulates encrypted authentication data within a single, stateless UDP packet, thereby simplifying the authentication process and mitigating risks such as sequence sniffing, replay attacks, and DoS buffer exploits associated with traditional PK. This innovation significantly improved both the security and operational efficiency of pre-authentication.

However, SPA introduced its own limitations: it relies on the UDP protocol, which restricts compatibility with TCP-based services, and requires access to raw sockets or privileged networking APIs, thus necessitating dedicated client software and limiting its deployability in browser-based or constrained environments.

SPA simplifies the authentication process and enhances security by transmitting encrypted authentication information within a single UDP packet. It employs symmetric key encryption such as AES-128 to

protect authentication data and utilizes HMAC (Hash-based Message Authentication Code) to verify data integrity. This approach effectively prevents threats such as sniffing and replay attacks.

SPA plays a critical role in Zero Trust-based systems, particularly in Software-Defined Perimeter (SDP) architectures. SDP conceals network resources and blocks unauthorized entities by using SPA as an authentication mechanism. During the initial communication between the SDP controller and the host, SPA verifies access requests, ensuring that unauthorized entities cannot connect to the network [23].

Additionally, the system does not respond to unauthorized requests, thereby minimizing the attack surface. For example, controllers and hosts can mitigate threats such as DDoS attacks by dropping unauthorized requests [24].

In SDP environments, SPA serves as a key mechanism for initial authentication between the Initiating Host (IH) and the Accepting Host (AH). Upon receiving an SPA packet, the AH verifies it before allowing connection requests, subsequently establishing mutual TLS (mTLS) for encrypted communication [25]. Throughout this process, unauthorized packets are entirely blocked, reinforcing network security and stability.

SPA features a structured message format that includes multiple fields for secure data transmission. According to the SDP specification, an SPA message consists of ClientID (identifying the user and device), Nonce (preventing replay attacks), Timestamp (limiting validity duration), and HMAC (ensuring data integrity). These elements ensure reliable and secure authentication across the network [26].

Despite its advantages, SPA has several limitations. First, SPA relies on the UDP protocol, making it unsuitable for connection-oriented protocols such as TCP. Second, SPA requires access to raw sockets or low-level network interfaces, necessitating a separate client application.

Additionally, it requires access to raw sockets or low-level networking APIs, which often demand administrative privileges or custom client software depending on the operating system, thereby limiting SPA's ease of deployment in general environments. This dependency restricts its applicability across various network environments and limits its flexibility.

Moreover, since SPA typically uses fixed port numbers, it becomes susceptible to observation-based man-in-the-middle (MitM) attacks. This vulnerability has prompted related research to explore potential mitigations and improvements to SPA's security model [27].

Although PK and SPA have significantly contributed to pre-authentication mechanisms, both approaches have inherent limitations. PK lacks encryption and is vulnerable to replay attacks, while SPA, despite its security advantages, depends on UDP and requires a dedicated client application.

To address these issues, this study introduces a novel authentication mechanism that leverages encrypted port sequence, eliminating client dependency and ensuring compatibility with diverse network environments. Table 2 provides a comparative analysis of PK, SPA, and our proposed method, highlighting key differences in security, client dependency, resistance to DoS attacks, and applicability across different environments.

Table 2: Comparison of port knocking, SPA, and the proposed authentication method

Category	Port knocking (2003)	Single packet authorization (2006)	This study
Security	Low (plaintext transmission)	High (encryption, HMAC)	High (encryption and randomized sequences)
Client Dependency	None	Required	None (works in web environments)
Resistance to DoS attacks	Weak	Strong (non-responsive mechanism)	Strong (random signatures and port avoidance)
Shared public IP issue	Present	None	None (signature-based authentication)
Protocol dependency	Low (works with TCP/UDP)	UDP or Modified TCP based	Low (utilizes port sequence)
Authentication data transmission	port sequence	Single UDP packet	Encrypted port sequence
Applicable environments	Servers and some network devices	Secure authentication between client and server	Compatible with client-server and web environments

Note: The years in parentheses indicate the initial proposal of each technique. The comparison is based on known features, security properties, and implementation requirements from prior literature and our analysis.

2.2.4 Recent Advances in Port-Space-Based Pre-Authentication

In response to the evolving landscape of pre-authentication mechanisms and Zero Trust network architectures, several recent studies have explored innovative techniques utilizing port space and authentication obfuscation.

Xu et al. [28] proposed AHAC, a framework that integrates obfuscated Single Packet Authorization (SPA) with Zero Trust principles to enhance stealth and resistance to traffic analysis. Although effective, the approach introduces computational overhead and relies on custom client implementations, limiting its deployment in lightweight or browser-based environments.

Junquera-Sánchez et al. [29] introduced C-Lock, a port-knocking system that leverages time-based one-time passwords (TOTP) to generate time-variant authentication sequences, thereby improving security against replay attacks. However, its reliance on accurate time synchronization between clients and servers poses challenges in distributed or asynchronous systems.

Wang et al. [30] presented a dual-mode SPA framework designed for software-defined networks (SDNs), introducing identity obfuscation and dynamic policy-based access control. While this significantly enhances adaptability and anonymity in programmable networks, it requires SDN-specific infrastructure and entails high configuration complexity, limiting its compatibility with traditional network environments.

These recent contributions collectively advance the field by addressing vulnerabilities inherent in traditional Port Knocking (PK) and SPA methods. Importantly, studies such as AHAC and Dual-SPA not only propose theoretical enhancements but also validate their approaches through empirical implementation and evaluation, demonstrating a balanced progression from conceptual innovation to practical feasibility.

This research aligns with and extends these efforts by proposing a novel protocol-independent, clientless pre-authentication mechanism that enhances scalability, compatibility, and resilience against denial-of-service (DoS) attacks in real-world distributed environments. Table 3 provides a comparative analysis of recent studies, highlighting their key features, contributions, and limitations.

Table 3: Comparison of recent pre-authentication techniques and the proposed method

Category	Dual-SPA (2022)	AHAC (2024)	C-Lock (2025)	This study
Authentication strategy	Dynamic Policy-based SPA	Obfuscated SPA	TOTP-based Port Knocking	Encrypted port sequence
Client dependency	Required (SDN-compatible client)	Required (custom client)	Required (TOTP support)	None (browser-compatible)
Protocol dependency	UDP-based SPA for SDN	UDP-based SPA	TCP/UDP (Port Knock)	Protocol-independent (Transport Layer)
Resistance to replay attacks	Strong via dynamic policies	Improved via obfuscation	Improved via time-variant sequences	Strong via message based port sequence, nonce and HMAC
Infrastructure requirements	SDN Controllers	Standard networks with obfuscation support	Local synchronized networks	None (standard web environments)
Deployment complexity	High (SDN programming)	Moderate (obfuscation overhead)	Moderate (time synchronization needed)	Low (port sequence transmission)
Applicability scope	SDN-based networks	Private/enterprise networks	Local networks	Browser-based, AIoT, cloud environments

To ensure that this review reflects the current state of research, we incorporated recent studies published between 2022 and 2025, capturing both theoretical advancements and empirical validations in pre-authentication.

In contrast, our proposed method offers a protocol-independent, client-less, and browser-compatible pre-authentication framework. By utilizing encrypted port sequence while avoiding reserved and unsafe ports, our approach enhances compatibility and scalability, particularly for AIoT and web-based systems where raw socket access is impractical.

3 Port-Based Pre-Authentication and Post-Connection: A Protocol-Independent Model

3.1 Design Objectives

Existing Port Knocking (PK) and Single Packet Authorization (SPA) methods often face limitations due to their dependencies on specific network protocols and implementation constraints.

PK is susceptible to sniffing and replay attacks, while SPA relies on dedicated client software and is primarily designed for UDP-based authentication. To overcome these limitations, this study proposes a protocol-independent port number-based Pre-Authentication and Post-Connection (PAPC) mechanism.

The proposed method is designed with four key objectives in mind. First, it achieves client independence by eliminating the need for dedicated client software. Instead of transmitting authentication packets with payloads, the system encodes authentication information into port sequence. This approach supports seamless integration with web browsers, mobile applications, and AIoT environments without requiring additional configurations or protocol adaptations.

Second, the method ensures protocol independence. While traditional SPA is limited to UDP, the proposed mechanism operates at the Transport Layer without relying on any specific protocol such as TCP or UDP. This broadens its applicability across diverse network environments, especially where certain protocols may be restricted or blocked.

Third, the method enhances security by applying encryption and Hash-based Message Authentication Code (HMAC) techniques. These safeguards protect the confidentiality and integrity of authentication messages, defending against man-in-the-middle (MITM) attacks and preventing unauthorized message tampering or replay attacks. Additionally, by aligning with the Zero Trust security model, the proposed scheme maintains robust authentication even in environments unsuitable for conventional SPA.

Lastly, the system is designed for scalability. Because it uses only the port space for authentication, it integrates easily into existing network infrastructures without requiring significant architectural changes. This design ensures compatibility with a wide variety of devices and supports flexible deployment in heterogeneous networks.

3.2 Methodology Overview

The proposed methodology utilizes the Transport Layer's port number field as a covert and protocol-independent channel to securely transmit encrypted authentication information. This approach enables seamless pre-authentication without requiring custom client software or application-layer modifications, thereby offering strong compatibility across AIoT environments.

Fig. 1 depicts the interaction between the core components: the **Client**, which initiates the authentication; the **Server**, which monitors, verifies, and authorizes access; and the **Key Management System (KMS)**, which securely stores and supplies cryptographic keys. This figure illustrates the architecture of the proposed port-based pre-authentication framework, which consists of three primary functional blocks.

The **Client Block** is responsible for generating the authentication information. This information is encrypted using AES-128 and authenticated using HMAC-SHA256 to ensure confidentiality and integrity before transmission. The resulting encrypted payload is then fragmented into a sequence of destination port numbers, which are transmitted to the server without requiring application-layer interaction. This design minimizes client-side complexity by eliminating the need for custom communication protocols and enables broad compatibility across diverse network environments, including browser-based systems.

The **Server Block** monitors the incoming port sequence data passively without initiating communication with clients. It reconstructs the received port sequence and verifies the message's authenticity by interacting with the KMS. Upon successful verification, it applies firewall rules to authorize access and enforces timing and ordering constraints to ensure protocol integrity. This passive monitoring approach is deliberately chosen to reduce the server's exposure to active network scanning, enhancing stealth and resilience against reconnaissance and DoS attacks.

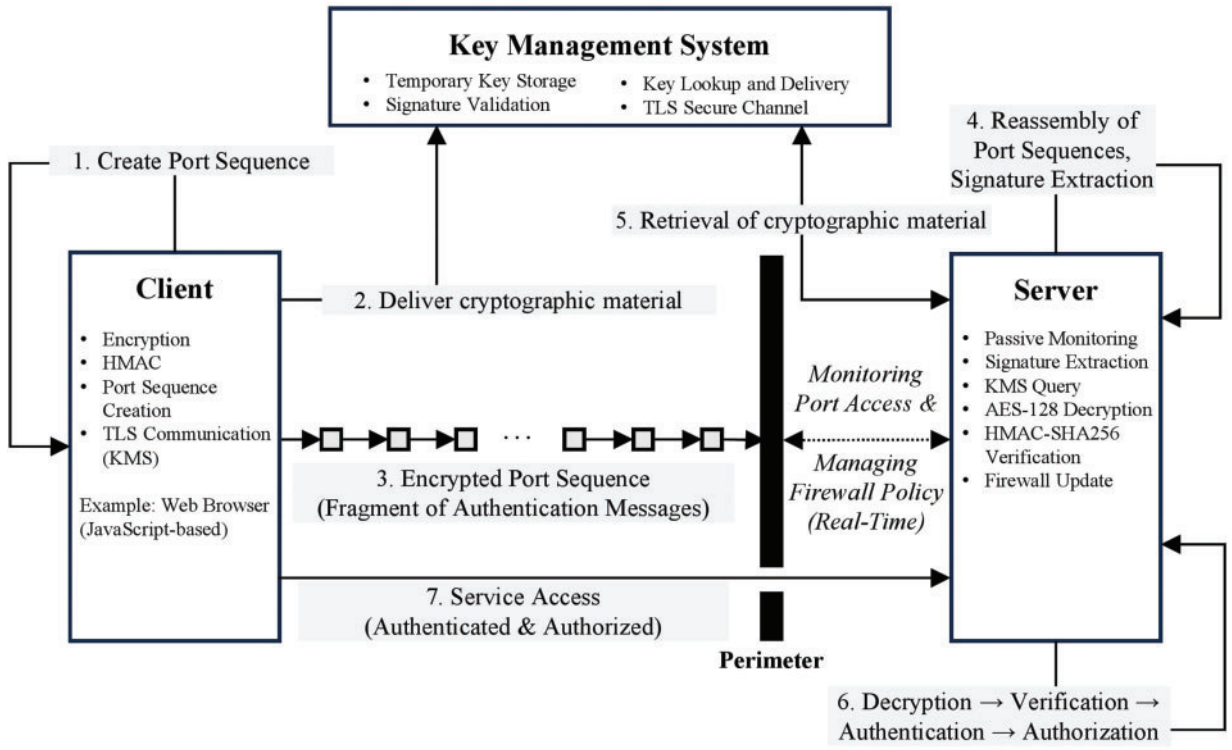

Figure 1: Overall architecture of the proposed PAPC system. The client generates encrypted authentication information and transmits it via port sequence. The server reconstructs and verifies the data using keys temporarily stored in the Key Management System (KMS), enforcing access control only after successful authentication

The **Key Management System (KMS) Block** handles the secure storage of cryptographic key pairs registered by the client over a TLS-secured channel. The KMS maps these keys to the client's IP address and signature identifier, enabling secure and dynamic key retrieval by the server during the authentication phase. This separation of key management from the authentication processing ensures scalability, facilitates temporary key handling without long-term storage on the server, and enhances overall system modularity and maintainability.

The system follows a seven-step operation procedure.

In the first step, the client encrypts the authentication data using AES-128, appends a nonce and timestamp, and generates a signature with HMAC-SHA256. This encrypted and signed payload is then fragmented and mapped to a sequence of destination port numbers.

In the second step, the encryption key and HMAC key are securely registered with the KMS via a TLS connection. The KMS associates the key pair with the client's IP address and the corresponding signature for future reference.

In the third step, the client transmits a sequence of packets where each destination port represents a fragment of the encrypted message. Sequence numbers are embedded to support reassembly even in the case of out-of-order delivery, and redundancy is introduced to improve reliability.

In the fourth step, the server monitors incoming port access attempts, extracts valid sequences, and identifies the embedded signature. It uses this signature to query the KMS for the associated cryptographic keys.

In the fifth step, upon receiving a valid signature query, the KMS verifies its authenticity and responds with the corresponding keys. If the signature is invalid or expired, the system enforces countermeasures such as buffering, blacklisting, or delaying the request.

In the sixth step, the server decrypts the message using AES-128 and verifies the integrity using HMAC-SHA256. Replay protection is enforced through nonce validation and timestamp freshness checks. The system supports ±5 s of time drift, with optional Network Time Protocol (NTP) based synchronization or round-trip delay-based estimation.

Finally, if all verification steps succeed, the server updates the firewall policy to grant the authenticated client access to the designated service or resource.

This authentication mechanism is designed to be lightweight and compatible with resource-constrained AIoT devices. The modular structure allows for integration with existing security infrastructure and Zero Trust Architecture.

The proposed method offers several key features. It provides robust resistance to replay attacks through nonce and timestamp verification, ensures message integrity via HMAC-SHA256, and maintains confidentiality through AES-128 encryption. Furthermore, it supports tolerance to packet reordering by embedding sequence numbers, and facilitates dynamic key handling through the use of a Key Management System. The clientless design additionally enhances interoperability across heterogeneous systems without requiring protocol-specific modifications. These features collectively contribute to a scalable and secure authentication mechanism suitable for a wide range of distributed and resource-constrained environments.

While the system demonstrates strong performance under controlled conditions, certain limitations remain. The current prototype has been implemented and tested only within a virtualized environment. Future work will involve evaluating the system in real-world network conditions characterized by high packet loss, congestion, and dynamic routing. Additionally, scalability testing with large numbers of simultaneous clients and long-term operational robustness under diverse threat scenarios will be explored to further validate the practicality of the proposed approach.

3.3 Authentication Message Design

3.3.1 Authentication Message Structure and Generation

The authentication message contains essential information required for authentication and authorization, which is transmitted via a port sequence. Unlike conventional PK, which lacks encryption and is susceptible to replay attacks, the proposed method generates a structured authentication message and applies security mechanisms such as encryption, HMAC, and replay attack prevention.

This approach is inspired by SPA's secure message transmission framework. Table 4 presents the attributes included in the authentication message.

Table 4: Authentication message attributes

Category	Name	Description
User-input	User_id	Unique identifier of the authentication requestor (Key owner)
	Nonce	Random value to prevent replay attacks
	Timestamp	Message creation time, used for timeout validation
Combined	Pub_ip	Public IP address of the authentication requestor, auto-generated for NAT handling

(Continued)

Table 4 (continued)

Category	Name	Description
Auto-gen	Message_type	Type of message (e.g., port opening, authentication request)
	Message_string	Raw authentication message containing user input or system-generated data
	HMAC	Hash-based Message Authentication Code for message integrity verification

The authentication message consists of seven attributes and can be transmitted to the server in various formats. To minimize the overhead caused by excessive port requests, the message follows the format:

```
User_id: Nonce: Timestamp: Message_type: Message_string: HMAC;
```

Each attribute is separated by a colon (`:`). The HMAC is computed over the raw authentication message before being appended. To ensure secure transmission, the authentication message is then encoded into a hexadecimal format before being sent.

3.3.2 Authentication Message Generation Process

To generate the authentication message, the process begins with the creation of a raw message. The user-input values–`User_id`, `Nonce`, `Timestamp`, `Message_type`, and `Message_string`–are concatenated into a single string.

Next, a Hash-based Message Authentication Code (HMAC) is generated using the concatenated string as input, thereby ensuring the integrity and authenticity of the message.

Once the HMAC is appended, the complete authentication message is encrypted using the AES algorithm to guarantee confidentiality and protect it from unauthorized interception.

The resulting encrypted message is then encoded into a hexadecimal format to facilitate compatibility with the port-based transmission mechanism.

Finally, the encoded message is fragmented into smaller segments, each mapped to specific destination port numbers. These port-mapped fragments are then transmitted sequentially to the server.

This structured message generation process enhances the security of the overall system, prevents replay attacks through unique nonce and timestamp usage, and ensures compatibility with protocol-independent authentication mechanisms.

3.3.3 Authentication Message Verification

The verification of authentication messages is performed on the receiving side to ensure that the received messages meet integrity and validity requirements.

This process is crucial for protecting the system from external threats and ensuring that only authenticated users can access resources. Algorithm 1 illustrates the step-by-step authentication message verification process.

Algorithm 1: Authentication message verification algorithm.

Input: $M = \{User\_id, Nonce, Timestamp, Pub\_ip, Message\_type, Message\_string, HMAC\}$,
$secret\_key$: A shared or dynamically retrieved key,
$nonce\_store$: Set of used Nonce values,
$current\_time$: Current server timestamp,
$time\_threshold$: Allowed time difference,
$actual\_ip$: IP address from which message was received.
Output: Verification Result (`Success` or `Failure`)

$generated\_HMAC \leftarrow HMAC(secret\_key, M.\text{contents})$;
if $generated\_HMAC \neq M.HMAC$ **then**
    return `Failure: HMAC Mismatch`;
else if $|current\_time - M.Timestamp| > time\_threshold$ **then**
    return `Failure: Timestamp Expired`;
else if $M.Nonce \in nonce\_store$ **then**
    return `Failure: Nonce Reuse Detected`;
else if $M.Pub\_ip \neq actual\_ip$ **then**
    return `Failure: IP Mismatch`;
else
    $nonce\_store \leftarrow nonce\_store \cup \{M.Nonce\}$;
    return `Success: Message Verified`;

Step 1: HMAC Verification

The first step is verifying the HMAC value of the received message. The HMAC is computed using the message contents and a pre-shared secret key and then compared with the received HMAC value.

This step ensures that the message has not been tampered with during transmission. If the received HMAC does not match the locally generated HMAC, the message is deemed invalid and rejected. In a Key Management System (KMS)-integrated system, the secret key can be dynamically retrieved for validation purposes.

Step 2: Timestamp Validation

Timestamp validation is performed next. The timestamp in the message represents the time at which the message was created and must fall within an acceptable time threshold set by the system.

This validation step prevents replay attacks by ensuring that messages are processed only within a valid time frame. If the received timestamp exceeds the allowed time difference, the message is immediately discarded.

Step 3: Nonce Verification

Nonce values play a crucial role in guaranteeing message uniqueness. The received Nonce is checked against a Nonce store that maintains a record of previously received Nonce values. If the Nonce has already been used, the message is identified as a replay attack attempt and is rejected. If the Nonce is unique, it is added to the Nonce store for future reference.

Step 4: (Optional) IP Verification

Depending on the security policy, an additional layer of validation can be applied by verifying the sender's public IP address (Pub_ip). The received Pub_ip is compared with the actual IP address from

which the request originated. If the IP addresses do not match, the message is considered unauthorized and discarded.

For environments using NAT (Network Address Translation), this step may need additional configuration to correctly map internal and external IPs. Failure to account for NAT behavior could lead to incorrect rejection of legitimate authentication requests.

Step 5: Verification Result Processing

If all the above conditions are met, the message is successfully authenticated and processed accordingly. If any validation step fails, the message is immediately rejected, and the failure reason is logged. The log data is used for security auditing and system monitoring. If necessary, an alert is sent to the security management system, and additional countermeasures, such as blocking the sender's connection, can be taken.

Security Considerations

The multi-step validation process effectively mitigates risks associated with message tampering, replay attacks, and unauthorized access attempts. It also aligns with the Zero Trust security model, ensuring continuous authentication and integrity verification before granting resource access.

3.4 Port Sequence-Based Authentication Mechanism

Fig. 2 illustrates the proposed port sequence-based authentication mechanism. In this example, both the sequence number and payload are assumed to be 8-bit in size. The client initiates the process by generating an authentication message, which is then transmitted as a sequence of port numbers to the server. Upon receiving the port sequence, the server reconstructs the original message and performs authentication.

The process begins with the construction of a raw authentication message, which combines user-provided input with automatically generated system data. Following this, the client generates cryptographic materials, including an AES encryption key, an initialization vector (IV), and an HMAC key. These cryptographic materials are securely transmitted to the Key Management System (KMS) via a TLS-secured connection.

Once the message and keys are prepared, the client appends the HMAC value to the authentication message and encrypts it using the AES algorithm. The resulting ciphertext is then fragmented into predefined sizes.

Each fragment is assigned to a unique port number based on a signature value. This signature not only forms an integral part of the transmitted port sequence but also ensures that unsafe or well-known ports are avoided. Moreover, it assists the server in identifying the corresponding cryptographic keys stored in the KMS.

The client combines each data fragment with a sequence number to form the final port sequence, which is transmitted to the server in parallel or sequential order. Upon reception, the server observes and reconstructs the original message from the received port sequence. The reconstructed message is then subjected to a comprehensive verification process, which includes validation of the HMAC, timestamp, nonce, and signature.

Once the message has been successfully authenticated, the firewall rules are dynamically updated to grant access to the verified client, thereby completing the pre-authentication and post-connection procedure.

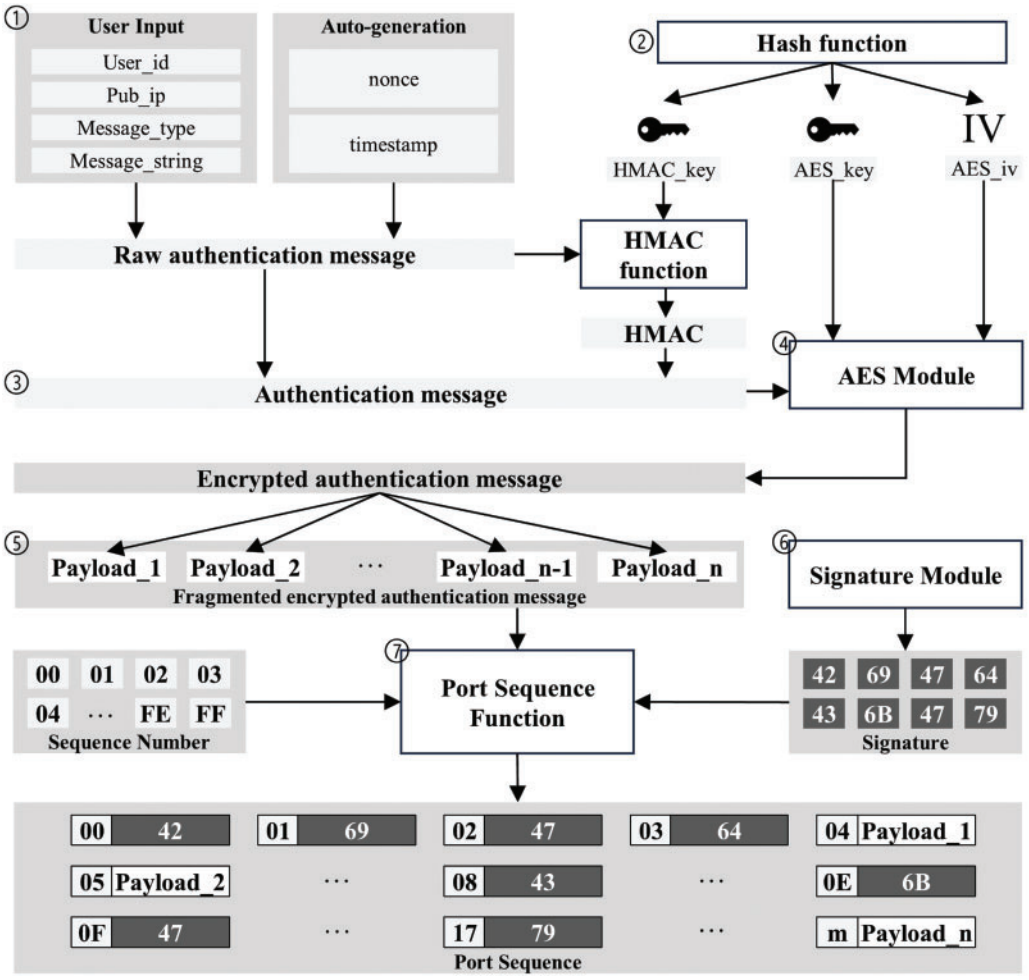

Figure 2: Process of port sequence creation. Authentication data is encrypted, fragmented, and mapped to destination ports while avoiding unsafe or reserved port ranges. Each fragment includes a sequence number and a signature prefix to enable correct reconstruction and verification on the server

3.4.1 Signature

Encrypted authentication messages are fragmented before transmission, increasing the risk of duplicate or ambiguous authentication attempts. To mitigate this issue, a unique signature is required to identify each authentication request distinctly.

The signature serves as a key component that allows the server to retrieve decryption and HMAC keys from the Key Management System (KMS), thereby ensuring the integrity and trustworthiness of authentication messages.

The signature is generated by utilizing port spaces while avoiding well-known and unsafe ports. Well-known ports are reserved for specific services, increasing the likelihood of conflicts, while unsafe ports are associated with security vulnerabilities and are often blocked by firewalls or browsers (e.g., Google Chrome).

These ports can cause packet loss or authentication failures during the message transmission process. The port avoidance mechanism is crucial in preventing such issues and ensuring a stable authentication procedure.

Avoided Port Grouping

The signature generation begins by collecting a predefined list of avoided ports. Each port in this list is grouped based on its upper n-bit values. Grouping is performed by extracting the upper n bits from each port number, clustering ports with identical values into the same group. This method ensures that reserved and restricted ports are systematically excluded from use.

Algorithm 2 details the grouping process while considering the sequence number size.

Algorithm 2: Avoid port grouping

Require: Port avoidance list P_{avoid}, Sequence number size n
Ensure: Grouped ports $G = \{G_1, G_2, ..., G_k\}$
Initialize $G \leftarrow \{\}$ {Initialize an empty set of groups}
for each port $p \in P_{\text{avoid}}$ **do**
  Extract sequence number $seq \leftarrow \text{upper\_bits}(p, n)$
  Extract payload $payload \leftarrow \text{lower\_bits}(p, n)$
  if $seq \notin G$ **then**
    Initialize a new group $G_{seq} \leftarrow \{\}$
    Add G_{seq} to G
  end if
  Add p to group G_{seq}
end for
return G

Signature Generation

After grouping, a pseudo-random value of $(16 - n)$ bits is generated for each group. The final 16-bit signature is constructed by concatenating the upper n-bit group value with the lower $(16 - n)$-bit pseudo-random value. This approach maintains uniqueness and minimizes the risk of port conflicts or authentication failures. Algorithm 3 outlines the signature generation procedure.

Algorithm 3: Signature generation algorithm

Require: Port avoidance list P_{avoid}, Sequence number size n (in bits), Random value generator $\text{rand}(k)$
Ensure: Set of 16-bit signature values S
Initialize $S \leftarrow \{\}$
Divide P_{avoid} into k groups $G = \{G_1, G_2, ..., G_k\}$ based on n
for each group $G_i \in G$ **do**
  $group\_value \leftarrow \text{upper\_bits}(G_i, n)$
  $r \leftarrow \text{rand}(16 - n)$
  $s_i \leftarrow \text{concat}(group\_value, r)$
  $S \leftarrow S \cup \{s_i\}$
end for
return S

Example Case: Avoiding Reserved Ports

Consider an avoided port list $P_{\text{avoid}} = \{FFFF, FFFE, FFFD, FFFC\}$, where the upper-bit size is set to $n = 8$. Extracting the upper 8 bits from each port results in a common group value of FF, forming a single

group G_{FF}. A pseudo-random 8-bit hexadecimal value (e.g., AB) is then generated. The final signature is computed by concatenating the group value {FF} with the pseudo-random value {AB}, yielding {FFAB}. The resulting set of signature values is $S = \{FFAB\}$.

The generated signature consistently maintains a 16-bit size, ensuring uniqueness and integrity by combining group values with pseudo-random data. Ports within the same upper-bit group are always assigned to the same category, preventing conflicts. This mechanism enables reliable authentication message differentiation while avoiding packet loss and enhancing system efficiency.

3.4.2 Port Range Selection and Scalability

To ensure compatibility and scalability across diverse network environments, the proposed authentication framework carefully selects port ranges used for transmission. The 16-bit port space (0–65535) includes reserved, unsafe, and user-defined ranges.

To avoid conflicts and transmission failures, well-known and unsafe ports are excluded from the candidate set. These ports are grouped and filtered during the signature generation phase (as described in Algorithm 2), ensuring that only safe and application-compatible ports are utilized.

The upper n-bit portion of each port is used to define logical groups, allowing the system to avoid restricted ranges while still preserving a wide space for signature and payload encoding. This grouping mechanism enhances scalability, enabling the system to adapt to network environments with varying firewall and NAT configurations.

Furthermore, the system dynamically assigns safe, pseudo-random lower bits to ensure port uniqueness while supporting multiple concurrent authentications. As a result, the proposed method is highly scalable and suitable for deployment in large-scale environments such as AIoT, cloud, and edge computing systems.

3.4.3 Port Sequence Generation

The port sequence generation method proposed in this study plays a crucial role in implementing Pre-authentication and Post-connection (PAPC) while ensuring protocol independence.

Unlike traditional Port Knocking (PK) mechanisms, which rely on sequential port hits without strong security guarantees, the proposed method introduces a structured, encrypted, and verifiable port sequence that ensures reliable authentication while maintaining high adaptability across different network environments.

To achieve robust and efficient authentication, the proposed method follows a multi-stage design. Initially, the raw authentication message is generated, incorporating both user-provided and system-generated data.

This ensures that each authentication attempt remains unique while maintaining integrity. Following this, cryptographic materials, including AES encryption keys, initialization vectors (IVs), and HMAC keys, are dynamically generated and securely stored within the Key Management System (KMS) to ensure secure key distribution and management.

Once the authentication message is securely structured, encryption and fragmentation processes are applied. The encryption phase utilizes AES-128 encryption, ensuring confidentiality while mitigating the risk of unauthorized message interception.

The fragmented message is then structured into discrete port sequence, each carrying an encrypted portion of the authentication message, thus preventing unauthorized access to the complete authentication payload.

A key innovation of the proposed approach is the incorporation of sequence numbers and cryptographic signatures. Unlike traditional PK-based methods, where port knocking sequences lack cryptographic integrity, the proposed port sequence generation technique integrates a signature value derived from a well-structured avoidance list. This ensures that the assigned port numbers remain within a safe range, avoiding well-known ports and unsafe ports that could lead to conflicts or security vulnerabilities.

The system is further optimized for parallel transmission and loss resilience. Traditional PK mechanisms suffer from strict sequential port knocking, which can be disrupted by packet loss or reordering.

The proposed method introduces sequence numbers that allow for out-of-order reception, enabling efficient parallel authentication attempts while ensuring the correct message reconstruction at the server side. Additionally, the protocol can be repeated or retransmitted as needed, greatly improving reliability in lossy networks.

Algorithm 4 presents the detailed procedure for generating the port sequence, ensuring cryptographic integrity and resilience against packet loss during authentication.

Algorithm 4: Port sequence generation algorithm

Require: Raw authentication message M, AES key K, Initialization vector IV, HMAC key H_k, Fragment size f, Signature S

Ensure: Port sequence P

Generate raw authentication message M
Generate cryptographic materials: K, IV, H_k
Append HMAC value: $M \leftarrow M \| \text{HMAC}(H_k, M)$
Encrypt message: $M_{\text{enc}} \leftarrow \text{AES}(K, IV, M)$
Fragment encrypted message: $F \leftarrow \{F_1, F_2, ..., F_n\}$
Extract group value from S: $G \leftarrow \text{group\_value}(S)$
Initialize sequence number: $seq \leftarrow 0$
for each fragment $F_i \in F$ **do**
  while $seq = G$ **do**
    $seq \leftarrow seq + 1$ Skip conflicting sequence numbers
  end while
  $P_i \leftarrow \text{concat}(seq, F_i)$
  Append P_i to P
  $seq \leftarrow seq + 1$
end for
Add S to the beginning of P: $P \leftarrow \{S\} \cup P$
return P

The algorithm systematically transforms the raw authentication message into a structured port sequence while avoiding potential port conflicts. It ensures that the server can reconstruct the message accurately, even if packets arrive out of order, thereby improving authentication robustness.

Protocol Filtering Robustness

Unlike SPA, which relies on UDP packets for authentication, the proposed port-based mechanism transmits authentication data through destination ports alone, without any payload. This enables compatibility in restricted network environments where both TCP and UDP packets are filtered by firewalls.

3.4.4 Port Sequence Verification

Port sequence verification is the process in which the server receives the port sequence generated by the client, analyzes it, and reconstructs the authentication message. This step ensures that the transmitted port sequence maintains integrity and validity, providing secure authentication message processing.

The primary objectives of port sequence verification are to validate the signature, prevent sequence number conflicts, reconstruct messages, and verify integrity, thereby mitigating replay attacks and unauthorized modifications. The verification process follows Algorithm 5, ensuring systematic authentication.

Algorithm 5: Port sequence security validation algorithm

Require: Port sequence P, KMS keys K, Current timestamp T_{current}, Sequence number bit size n
Ensure: Validation result (`Success` or `Failure`)
Extract Signature S and fragments $\{(seq_i, F_i)\}$ from P
Retrieve group value G from S
Verify Signature:
if $G \notin$ valid groups **then**
  return `Failure: Invalid Signature`
end if
Sort fragments by sequence number
Initialize expected sequence number: $expected\_seq \leftarrow seq_0$
Initialize empty set $seen\_seqs \leftarrow \{\}$ {Track seen sequence numbers}
Compute valid sequence number range: $seq_{\min} \leftarrow 0$, $seq_{\max} \leftarrow 2^n - 1$
for each fragment $(seq_i, F_i) \in P$ **do**
  Verify sequence number:
  if $seq_i = G$ **then**
    return `Failure: Sequence Conflict`
  end if
  if $seq_i \neq expected\_seq$ **then**
    return `Failure: Sequence Discontinuity`
  end if
  if $seq_i \in seen\_seqs$ **then**
  return `Failure: Duplicate Sequence Number`
  end if
  if $seq_i < seq_{\min}$ or $seq_i > seq_{\max}$ **then**
    return `Failure: Out-of-Range Sequence Number`
  end if
  Add seq_i to $seen\_seqs$
  $expected\_seq \leftarrow seq_i + 1$
  Reassemble message: $M_{\text{enc}} \leftarrow M_{\text{enc}} \| F_i$
end for
Decrypt message: $M \leftarrow \text{AES-decrypt}(K, M_{\text{enc}})$
if $M = \varnothing$ **then**
  return `Failure: Decryption Failed`
end if

(Continued)

Algorithm 5 (continued)

```
33: Perform Authentication Message Validation using Algorithm 1
34: if Algorithm 1 fails then
35:   return Failure: Message Validation Failed
36: end if
37: return Success: Message Validated
```

Timestamp Validation and Clock Drift Tolerance

To address potential issues with clock synchronization between the client and server, the system applies a tolerance window of ±5 s for timestamp verification. If Network Time Protocol (NTP) based synchronization is unavailable, the server can estimate round-trip delay to validate time freshness. This design ensures robust replay attack prevention without requiring perfect time alignment.

Initially, the server extracts the signature and fragmented data from the received port sequence. The signature serves as an identifier to maintain the uniqueness of the port sequence, and the server verifies whether the received sequence belongs to a valid group.

A valid group refers to a set of port sequence classified based on sequence number size within the port avoidance list. This classification prevents conflicts by ensuring that sequences do not overlap with reserved or unsafe ports.

Following signature validation, the server examines sequence numbers assigned to each data fragment. The verification step ensures that sequence numbers do not conflict with the signature group value. If a particular sequence number is found to be identical to the signature's designated group value, it is considered a conflict and is discarded.

Additionally, the sequence numbers are checked for order continuity, duplication, and compliance with the predefined range. Only after successful validation is the sequence reassembled and decrypted.

Once the data fragment verification is completed, the server reconstructs the encrypted authentication message in the correct sequence. The reconstructed message is decrypted using the cryptographic keys retrieved from the Key Management System (KMS).

Following decryption, the authentication message undergoes integrity verification using HMAC, timestamp validation, and replay protection as described in Algorithm 1. If the message is successfully validated, the authentication process is deemed successful, allowing secure access to the requested resources.

Port sequence verification is a fundamental security measure that ensures the safe processing of authentication messages. By leveraging Signature and HMAC, it guarantees data integrity, while Timestamp and Nonce enhance security by preventing message reuse and replay attacks. This verification mechanism effectively blocks tampered or reused authentication attempts, reinforcing the trustworthiness of the proposed authentication framework.

3.4.5 Security Enhancements

The proposed authentication mechanism incorporates multiple security enhancements to ensure secure and reliable message transmission. To prevent port collisions, the system strategically avoids well-known and unsafe ports, ensuring that port sequence do not interfere with reserved or restricted network services. This enhances the stability of message transmission and reduces the risk of conflicts within the network.

To mitigate the risks associated with Man-in-the-Middle (MITM) attacks, the authentication process employs encryption and HMAC verification. These security measures ensure that transmitted messages remain confidential and tamper-proof, preventing unauthorized modifications during transmission.

Furthermore, replay attack prevention mechanisms are integrated through the use of nonce values and timestamp validation. Each authentication message includes a unique nonce, ensuring that messages cannot be reused.

The timestamp validation mechanism ensures that messages remain valid only within a predefined time window, effectively preventing unauthorized retransmission and replay attacks.

Compared to existing Port Knocking (PK) and Single Packet Authorization (SPA) methods, the proposed approach demonstrates clear advantages in security, efficiency, and scalability.

Traditional PK methods lack inherent cryptographic protection, making them vulnerable to sniffing and replay attacks, while SPA requires dedicated client software and primarily operates over UDP, limiting its applicability in environments with strict firewall rules.

The proposed port sequence method overcomes these limitations by providing encryption, replay protection, protocol independence, and robust key management integration.

By leveraging KMS-assisted authentication, cryptographic port sequencing, and parallel transmission optimization, this approach not only enhances authentication security but also ensures high adaptability across diverse network environments, thereby establishing a scalable and secure authentication framework for modern networked systems.

3.5 Key Management System for Port-Based Authentication

3.5.1 Background and Motivation for KMS

Traditional authentication mechanisms typically rely on post-connection authentication, wherein cryptographic protocols establish security through a handshake process before exchanging encryption keys.

However, the pre-authentication and post-connection (PAPC) approach proposed in this study fundamentally differs in that authentication must occur before any connection is established. This distinction poses significant challenges in secure key exchange, necessitating an alternative key management strategy.

Historically, on-board key storage has been a common approach to handling pre-authentication environments. In this method, clients and servers pre-share encryption and authentication keys, storing them either physically or digitally for later use. While this approach ensures that authentication can occur without prior connection, it introduces multiple security and operational risks.

One major issue is key exposure–if an attacker gains access to stored keys, they can compromise the entire authentication process. Additionally, frequent key updates increase the operational complexity and maintenance burden, making it impractical for dynamic environments with multiple users or distributed systems.

To address these limitations, this study proposes the adoption of a Key Management System (KMS) to facilitate secure, ephemeral key management. The proposed KMS securely stores cryptographic materials generated by the client, including AES keys, Initialization Vectors (IVs), and HMAC keys.

Unlike traditional on-board storage, these keys are never permanently stored; instead, they are registered with the KMS upon generation and provided only to authenticated servers during the verification process. Once the authentication sequence is complete, the keys are immediately purged, ensuring that no persistent cryptographic material remains.

This dynamic key management approach enhances security by eliminating long-term key storage risks while reducing the operational overhead associated with key distribution and updates. Additionally, the proposed system ensures that authentication remains independent of pre-distributed credentials, thereby improving flexibility and scalability in dynamic network environments.

3.5.2 KMS-Based Encrypted Key Exchange Process

The **Key Management System (KMS)** ensures secure key management and ephemeral storage of cryptographic materials during the authentication phase.

Operating independently from both client and server, the KMS supports secure key registration, retrieval, and automatic deletion immediately after authentication is completed. Fig. 3 illustrates the step-by-step process involved in this secure interaction.

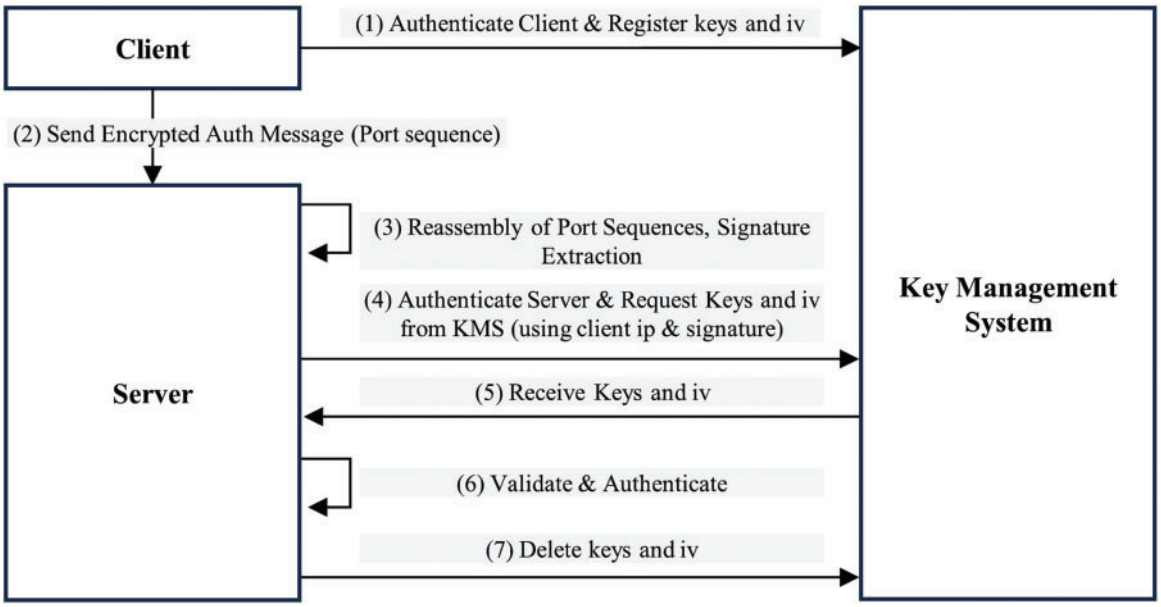

Figure 3: KMS-based encrypted key exchange process. The client registers encryption and HMAC keys along with a signature. The server uses this signature to securely retrieve the keys from KMS, validate the authentication message, and delete keys post-verification

Initially, the client generates the cryptographic keys–namely, the AES key, initialization vector (IV), and HMAC key–and securely registers them with the KMS. After key registration, the client transmits the encrypted authentication message, encoded as a port sequence, to the server.

Upon receiving the port sequence, the server reconstructs the original message fragments and extracts the embedded signature. This signature is then used, along with the client's IP address, to request the corresponding decryption keys from the KMS. If the validation is successful, the KMS returns the registered cryptographic keys to the server through a secure TLS channel.

The server uses these keys to decrypt the authentication message and verify its integrity using HMAC, as well as validate its freshness based on the timestamp. Once the authentication procedure has been completed successfully, the KMS immediately deletes the associated cryptographic materials, ensuring that no long-term storage of sensitive key data occurs.

This ephemeral key lifecycle minimizes the risk of key leakage while preserving the confidentiality and integrity of the authentication process.

By implementing **ephemeral key storage and Zero Trust-based authentication**, this process eliminates the risks associated with persistent key storage and unauthorized key access, enhancing overall security.

3.5.3 KMS Operations

The Key Management System (KMS) plays a central role in managing the encryption keys, IVs, and HMAC keys generated by clients while maintaining an independent operational environment separate from both clients and servers.

Unlike conventional pre-shared key (PSK) methods that require storing long-term encryption keys, the proposed KMS operates in a dynamic and temporary key management structure, ensuring that encryption materials are available only during authentication and immediately discarded upon completion.

When a client generates the necessary cryptographic keys for encrypting authentication messages, these keys are registered with the KMS. The KMS then associates these keys with the client's unique identifiers, such as IP addresses and signatures, enabling authenticated servers to request and retrieve the required keys during the verification process. Upon completion of authentication, the stored encryption materials are immediately deleted, thereby eliminating the risks associated with long-term key storage.

To prevent unauthorized access, the KMS employs pre-registered identity-based authentication mechanisms. The transmission of cryptographic data between clients, servers, and the KMS is protected using Transport Layer Security (TLS), ensuring secure communication channels.

The KMS stores only temporary key-related information, including AES encryption keys, initialization vectors (IVs), HMAC keys, and relevant client authentication attributes such as IP addresses, signatures, and user IDs. This information is managed according to a structured three-phase key lifecycle.

In the **transmission phase**, the client registers its keys, and they are temporarily stored in the KMS. During the **verification phase**, these keys are made accessible to authenticated servers that request decryption materials based on matching IP and signature values. Finally, in the **deletion phase**, once the authentication is successfully completed or expired, the corresponding cryptographic keys are permanently deleted from the KMS to eliminate the possibility of reuse or leakage.

Table 5 illustrates the fundamental database structure used for managing cryptographic keys within the KMS.

Table 5: KMS database structure and example data

IP Address	Signature	User ID	AES Key	IV	HMAC Key	Status	Expiration
192.168.1.10	ABC123DEF	client_01	XXXXX...	XXXXX	XXXXX...	Active	2024-02-10 12:30:00
192.168.1.11	XYZ789GHI	client_02	XXXXX...	XXXXX	XXXXX...	Expired	2024-02-10 12:05:00
192.168.1.12	LMN456JKL	client_03	XXXXX...	XXXXX	XXXXX...	Pending	2024-02-10 12:45:00

The KMS temporarily stores authentication keys registered by clients and allows servers to query them during authentication. The primary key attributes–**IP Address, Signature, and Client ID**–uniquely identify each client's key entry.

Once authentication is completed, the **AES Key, IV, and HMAC Key are immediately deleted**, ensuring that no long-term cryptographic material remains stored. The **Status** field categorizes keys as **Active**, **Pending**, or **Expired**, with the **Expiration** field tracking the validity period.

Unlike traditional methods that require pre-shared keys or external authentication devices such as OTPs or security cards, this system enables authentication without prior key distribution.

The proposed KMS-based approach provides **on-demand key management**, ensuring that authentication keys are valid only for a single session while significantly reducing operational overhead and security risks.

4 System Implementation

To validate the feasibility of the proposed protocol-independent port number-based pre-authentication and post-connection (PAPC) mechanism, we developed a fully functional system.

The system consists of three core components: Client-side, Server-side, and Key Management System (KMS). Each component operates as an independent module through an API-based architecture, allowing flexible integration into various network environments.

Unlike traditional authentication mechanisms that require dedicated client software, the proposed system is designed to function in a web-based environment, enabling authentication through a standard web browser. This approach eliminates the need for additional client software while maintaining security and usability.

4.1 System Architecture and Key Components

The proposed system is designed to implement the protocol-independent port number-based Pre-Authentication and Post-Connection (PAPC) mechanism in a real-world environment. The system consists of three major components: Client, Server, and the Key Management System (KMS).

Each component is implemented as an independent module using an API-based architecture, ensuring flexibility and scalability. To facilitate seamless authentication without requiring dedicated client software, the system is designed to operate in a web-based environment, enabling authentication directly through a web browser.

As illustrated in Fig. 4, the proposed system enables authentication and connection establishment through a structured data flow among three key components: the Client, the Server, and the Key Management System (KMS).

The **Client** initiates authentication requests through a web-based interface, eliminating the need for additional software installation. This interface allows users to generate a port sequence based on input values, ensuring both usability and security.

Upon initiating the authentication, the client generates encryption keys and initialization vectors (IVs), registers them with the KMS, encrypts the port sequence using AES-128, applies HMAC for integrity verification, and transmits the encrypted sequence through the network.

The **Server** continuously monitors incoming port sequence using a Port Sequence Receiver that captures traffic in real time.

When an authentication request is detected, the server reconstructs the received port sequence, queries the KMS to retrieve the corresponding encryption keys, decrypts the authentication message, and verifies its integrity using HMAC. If the authentication is verified successfully, the server updates the firewall policies dynamically via the Firewall Handler to allow the client access to the protected service.

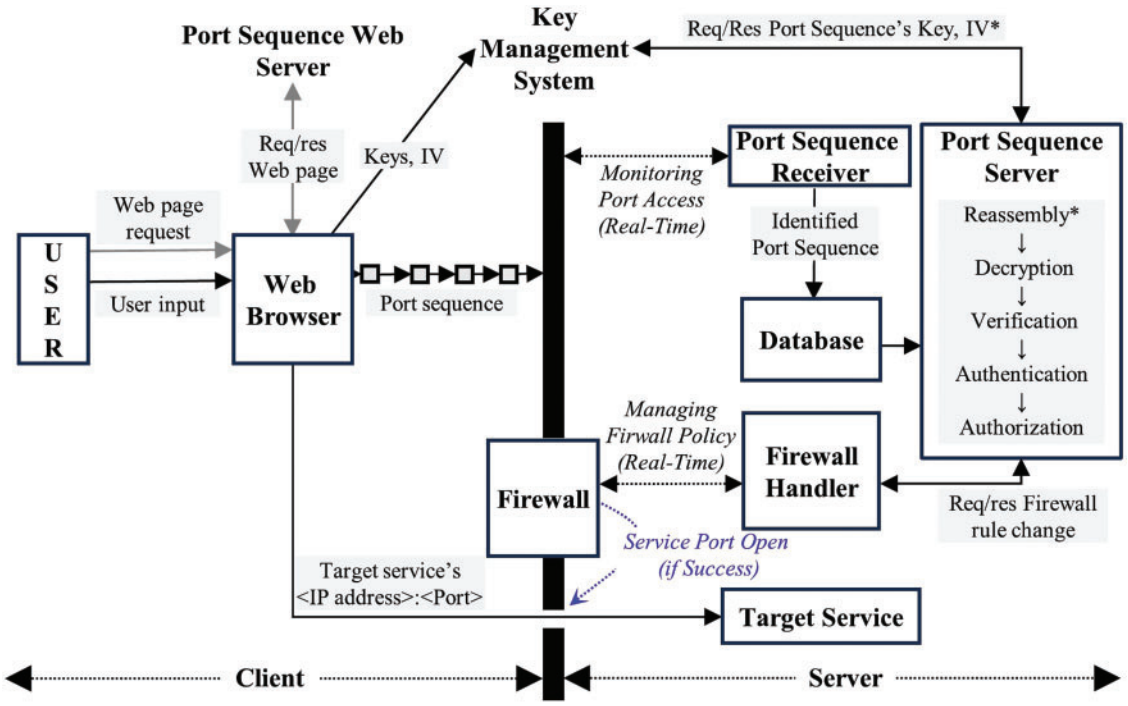

Figure 4: Overall system architecture and data flow among the client, server, and KMS. The client uses a web interface for authentication, while the server and KMS handle secure key management, port monitoring, message verification, and firewall updates

The **Key Management System (KMS)** manages encryption keys securely throughout the authentication process.

The client registers the AES key, IV, and HMAC key prior to authentication, and the server requests these keys when needed during message verification. The KMS enforces TLS-based secure communication for key exchange and only stores keys associated with active authentication sessions. After authentication is completed, the KMS immediately deletes all stored cryptographic materials to ensure ephemeral key management and prevent future misuse.

By integrating these components into a cohesive architecture, the proposed system achieves pre-authentication prior to connection establishment. Unlike conventional mechanisms that expose services before authentication, this approach ensures that no services are exposed until the authentication process is securely completed.

Additionally, the system incorporates essential security features including Nonce, Timestamp, and HMAC verification, which collectively ensure data integrity and robust protection against replay attacks.

4.2 Authentication and Connection Process

The authentication and connection process in the proposed system follows a structured approach to ensure both security and efficiency. Unlike traditional authentication mechanisms that expose services before verification, this system applies a Pre-Authentication & Post-Connection (PAPC) model, ensuring that no services are accessible before authentication is successfully completed.

4.2.1 Port Sequence Generation and Transmission

The user's request for a web page to generate a port sequence is initiated through a web browser, and the user input is used to generate an authentication message. Subsequently, the client generates cryptographic material, including an AES encryption key, an initialization vector (IV), and an HMAC key.

Upon initiating authentication, the web browser utilizes the cryptographic material to encrypt the authentication message and employs HMAC for integrity verification before transmitting it to the server. The client then dynamically generates a port sequence based on the encrypted authentication message and transmits it to the server.

4.2.2 Port Sequence Processing and Decryption

Upon receiving the encrypted port sequence, the server monitors the network interface in real time and reconstructs the original authentication message. Since port sequence may arrive out of order, the system employs a sequence alignment mechanism to correctly reorder fragments before processing.

The server then queries KMS to retrieve the associated cryptographic keys, decrypts the message, and verifies its integrity using the HMAC. For instance, if port sequence arrive in an unordered manner due to network conditions, the system reconstructs the correct order using sequence numbers before processing.

4.2.3 Authentication and Firewall Policy Update

Once the authentication message is validated, the server proceeds with user verification and authorization. If authentication is successful, firewall policies are dynamically updated to grant the authenticated client access to the designated target service. This step ensures that the service remains hidden from unauthorized users until authentication is completed.

4.2.4 Security and Key Expiry Mechanism

To maintain security, KMS enforces an automatic key expiration mechanism. Once authentication is completed, the cryptographic keys associated with the session are securely deleted after a predefined period. This approach mitigates risks associated with long-term key storage while ensuring that authentication keys cannot be reused for subsequent unauthorized attempts.

4.2.5 Security Enhancements

By transmitting authentication information through an encrypted port sequence, the system mitigates the vulnerabilities of traditional Port Knocking methods. Additionally, it integrates critical security features such as Nonce, Timestamp, and HMAC validation, effectively preventing replay attacks and ensuring message integrity. This implementation significantly enhances authentication security, making it resilient to known attack vectors.

The proposed authentication mechanism ensures that unauthorized users cannot gain access to services prior to verification, thereby minimizing the attack surface. Furthermore, the integration of KMS for dynamic key management reduces operational overhead while maintaining security, providing a scalable and adaptable authentication solution for modern networked environments.

4.3 Security and Scalability Design

The proposed system employs a port sequence-based authentication method that does not rely on specific network protocols or require custom packet formats. This approach ensures adaptability across

diverse environments, eliminating constraints imposed by traditional authentication mechanisms that depend on predefined communication protocols.

By implementing the encryption and authentication techniques described in Section 3, the system effectively protects against data tampering and eavesdropping. Additionally, the integration of a Key Management System (KMS) facilitates dynamic key handling, overcoming security vulnerabilities associated with pre-shared key mechanisms.

Unlike conventional authentication frameworks, the system is designed with an API-driven modular architecture, allowing for seamless integration into various network environments. The authentication mechanism is adaptable to cloud-based infrastructures, IoT devices, and other network services without requiring specialized client software. The web-based implementation further enhances usability by enabling clients to perform authentication directly through a browser, removing the need for additional software installation.

Furthermore, the system is engineered to maintain compatibility with both TCP and UDP-based communications, ensuring reliable authentication even in network environments utilizing NAT (Network Address Translation) or restrictive firewall policies. This flexibility broadens its applicability to real-world deployments where protocol constraints often hinder secure authentication.

5 Experimental Evaluation and Performance Analysis

This section presents the experimental validation of the proposed authentication method conducted within a controlled virtualized environment. The experiment was designed to evaluate the practical feasibility of the proposed scheme and verify its functional correctness. In addition, a theoretical analysis is included to provide mathematical support for the expected improvements in security and efficiency.

The experimental setup was implemented using VirtualBox 7.0, where two virtual machines (VMs) were deployed. The first machine hosted the Key Management System (KMS) and the web-based authentication interface, and the second machine operated as the Port Knocking Server along with the protected target service. Both VMs ran on Ubuntu 24.04, were allocated 2 vCPUs and 8 GB of RAM, and operated in network bridge mode to facilitate internal communication. The client, acting externally, accessed the system through a standard web browser.

The authentication system was built using a Node.js-based web API. On the client side, authentication was initiated via a browser (Google Chrome) without requiring additional client software. The client generated cryptographic materials including an AES encryption key, an initialization vector (IV), and an HMAC key, which were securely registered with the KMS. Subsequently, the client transmitted encrypted authentication data encoded as a port sequence to the server.

To verify the system's functionality, three key aspects were evaluated. First, the correctness of the overall authentication procedure was confirmed, including key generation, port sequence transmission, and server-side authentication handling.

Second, packet flow was monitored using Wireshark to ensure proper encoding and decoding of port sequence during transmission and reception.

Lastly, dynamic firewall policy changes were verified through iptables logs, confirming that access control rules were correctly applied based on authentication results. These findings validate the system's ability to perform secure and reliable authentication under realistic deployment conditions.

The experimental results demonstrated that the proposed authentication system functioned correctly under various network conditions. Through port sequence capture, server-side verification, and dynamic firewall updates, each step of the protocol was successfully validated. Table 6 summarizes the observed outcomes from the authentication process.

Table 6: Summary of experimental validation results

Validation step	Expected outcome	Result
Port sequence transmission (Client)	port sequence transmitted over network	Success
Port sequence detection (Server)	Correct reassembly of fragmented messages	Success
Signature verification	Identification of registered session via signature	Success
Key retrieval from KMS	Correct decryption keys returned for valid signature	Success
Message integrity check (HMAC)	HMAC validation for tamper detection	Success
Timestamp/Nonce validation	Replay prevention mechanisms enforced	Success
Firewall rule update	Access allowed upon successful authentication	Success
Unauthorized access handling	No rule changes for invalid attempts	Success

5.1 Performance and Resource Use of Client-Side Port Sequence Generation

To assess the feasibility of executing port sequence generation on lightweight client-side environments such as browsers or IoT edge devices, we conducted an integrated performance evaluation. The experiment measures both execution time and memory usage when generating all possible port sequence for increasing sequence number bit sizes, ranging from 5 to 15 bits.

Fig. 5 presents the results. As the sequence bit size increases, the number of generated port entries grows exponentially, leading to greater computational demands.

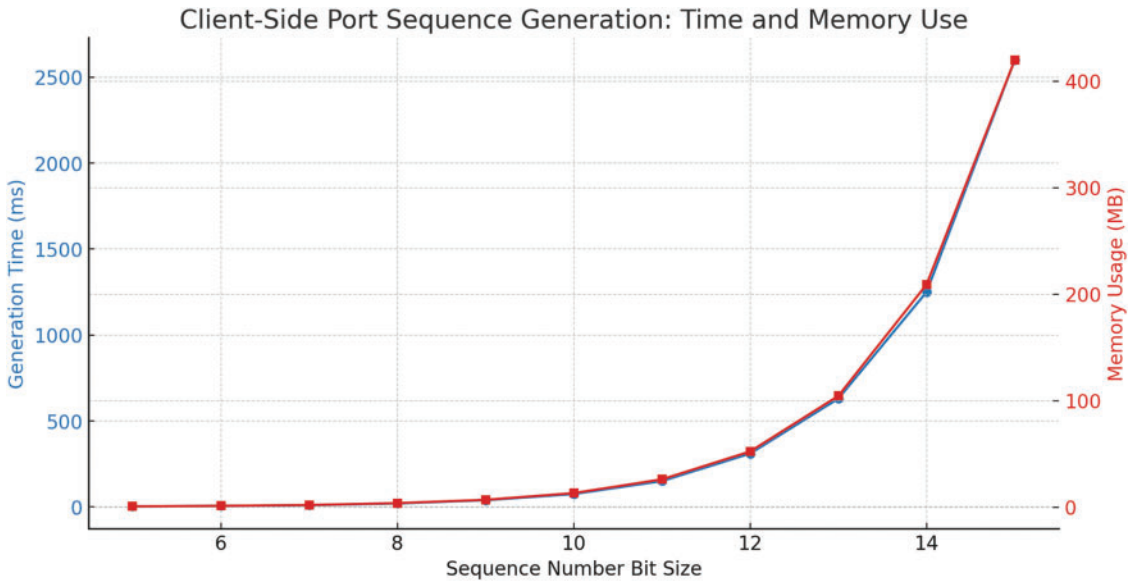

Figure 5: Performance analysis of client-side port sequence generation. Execution time and memory usage increase with sequence bit size, but remain within acceptable limits, confirming suitability for lightweight environments

However, the total generation time remains under 3 s even in the worst case (15-bit sequence), while memory consumption stays within 500 MB. This confirms the lightweight nature of the proposed method and its viability for real-time authentication on client platforms.

The observed trends demonstrate a nonlinear increase in time and memory usage as sequence size grows, particularly beyond 12 bits. Therefore, selecting an optimal sequence bit range (e.g., 6–11 bits)

offers a good balance between performance, capacity, and deployment flexibility, especially for mobile or embedded applications.

5.2 Unauthorized Request Detection Using Signature Filtering

To evaluate the system's capability in blocking unauthorized access attempts, we conducted an experiment focusing on signature-based filtering within the PAPC framework. The test was designed to assess whether the system could correctly differentiate between valid and invalid authentication sequences prior to decryption.

For the evaluation, the system was configured to operate with a 16-bit port encoding scheme, consisting of an 8-bit sequence number and an 8-bit data payload. This configuration aligns with the design presented in Section 3.4, enabling effective fragmentation and parallel transmission. However, for accuracy in detection evaluation, this experiment was conducted exclusively under the 16-bit configuration.

A set of 200 authentication attempts was performed, including both legitimate requests (generated with valid cryptographic keys and registered signatures) and unauthorized ones (crafted using invalid or reused signatures). The server performed passive signature verification without engaging the decryption module unless the signature matched a registered session in the Key Management System (KMS).

The results, summarized in the confusion matrix in Fig. 6, demonstrate that the signature-based mechanism effectively filtered unauthorized access at an early stage.

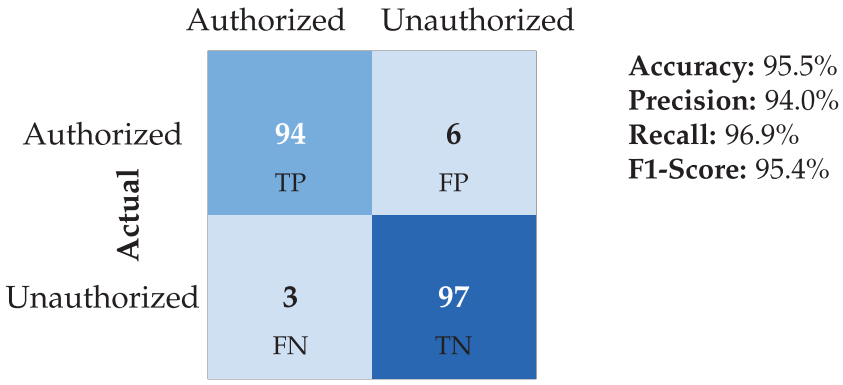

Figure 6: Confusion matrix illustrating detection accuracy of unauthorized authentication attempts using signature filtering. The proposed system achieved over 95% F1-score by rejecting invalid requests before full decryption, improving efficiency and reducing server load. Confusion matrix with intensity-based color highlighting and performance metrics (16-bit, 8-bit seq/data)

Of 100 unauthorized requests, 97 were correctly blocked based on invalid signatures, while only 3 bypassed the initial check. For the 100 legitimate requests, 94 were accepted, while 6 were incorrectly blocked due to signature collision or misclassifications. These results correspond to a precision of 94.0%, recall of 97.0%, and an F1 score of 95.5%, indicating the robustness of the early-stage filtering mechanism.

This result confirms that PAPC's design not only ensures secure message authentication but also enables pre-authentication threat mitigation at the network layer, significantly reducing unnecessary decryption attempts and resource usage.

5.3 Mathematical Security and Performance Analysis

While the experimental results confirm the system's feasibility, a theoretical analysis is essential to rigorously evaluate its security and efficiency.

Theorem 1: *The probability of two authentication requests generating identical port sequence is minimized when the Signature design is optimized.*

Proof: The probability of port sequence collision, given $P = 2^{16}$ available ports and$G$ Signature groups, with N simultaneous requests, is:

$$P_{\text{collision}} = 1 - \prod_{i=0}^{N-1} \left(1 - \frac{i}{P - G}\right) \tag{1}$$

By optimizing the Signature allocation and avoiding reserved or unsafe ports, we ensure that the probability remains negligibly low, as verified through combinatorial probability analysis.□

Lemma 1: *The probability of a replay attack occurring is negligible when using Nonce and Timestamp validation.*

Proof: For a randomly generated b-bit Nonce, the probability of reusing the same Nonce is:

$$P_{\text{replay}} = \frac{1}{2^b} \tag{2}$$

For a 128-bit Nonce, this results in $P_{\text{replay}} = 2^{-128}$, which is practically zero. Additionally, incorporating Timestamp validation:

$$P_{\text{replay,t}} = P_{\text{replay}} \times P_{\text{valid}}(T) \tag{3}$$

where $P_{\text{valid}}(T)$ is the probability of an authentication request being valid within a predefined time window. These combined mechanisms ensure near-zero replay attack probability.□

Proposition 1: *HMAC-based authentication prevents message tampering, ensuring resistance against Man-in-the-Middle (MITM) attacks.*

Proof: The probability of an attacker forging a valid HMAC is:

$$P_{\text{HMAC-collision}} = \frac{1}{2^n} \tag{4}$$

where n is the HMAC output length (e.g., for SHA-256, $n = 256$, so $P_{\text{HMAC-collision}} = 2^{-256}$). This ensures computational infeasibility for an adversary to forge a valid authentication message.□

Proposition 2: *The proposed method significantly reduces server processing overhead compared to Single Packet Authorization (SPA).*

Proof: In SPA, all UDP packets must be processed, leading to an overhead of:

$$O_{\text{SPA}} = R_{\text{total}} \times P_{\text{check}} \tag{5}$$

where R_{total} is the total received UDP packets per second, and $P_{\text{check}} = 1$ since all packets must be examined.

In contrast, the proposed Pre-Authentication and Post-Connection (PAPC) method selectively processes only authentication sequences:

$$O_{\text{PAPC}} = R_{\text{auth}} \times C_{\text{port-seq}} \tag{6}$$

where $R_{\text{auth}} \ll R_{\text{total}}$, reducing server overhead significantly.□

Proposition 3: *The proposed method reduces authentication latency compared to traditional Port Knocking (PK) due to parallel transmission.*

Proof: The latency of sequential Port Knocking (PK) is given by:

$$T_{\mathrm{PK}} = N \times T_{\mathrm{port}} \tag{7}$$

where$N$ is the number of port sequence, and T_{port} is the time per request.

The proposed PAPC method enables parallel transmission, reducing latency to:

$$T_{\mathrm{PAPC}} = \max(T_{\mathrm{seq1}}, T_{\mathrm{seq2}}, \ldots, T_{\mathrm{seqN}}) \tag{8}$$

Since the authentication sequence can be sent simultaneously rather than sequentially, authentication time is minimized, improving overall efficiency.□

6 Discussion

This section presents a structured discussion on the security posture and design evolution of port-based authentication methods. To contextualize the proposed protocol-independent port-based pre-authentication and post-connection (PAPC) scheme, we categorize the analysis into four parts: traditional Port Knocking (PK), Single Packet Authorization (SPA), the advanced PAPC mechanisms, and the limitations of existing approaches that PAPC seeks to overcome.

6.1 Port Knocking: Simplicity without Cryptographic Protection

Port Knocking (PK) is a legacy method that conceals services behind closed ports and opens them only after a specific sequence of connection attempts. Despite its simplicity and low overhead, PK lacks cryptographic protection, leaving it vulnerable to eavesdropping, spoofing, and replay attacks.

The static nature of knock sequences and the absence of integrity verification make it difficult to ensure the authenticity of the sender or the freshness of the request. Additionally, port sequence are predictable, and once discovered, can be reused by adversaries.

6.2 Single Packet Authorization: A Cryptographic Enhancement with Limitations

To address the lack of security in PK, Single Packet Authorization (SPA) introduces cryptographic authentication using a single encrypted UDP packet. While SPA improves security through HMAC and encryption, it relies on client-side software and UDP packet delivery, which presents compatibility issues in restrictive firewalled or NAT environments.

Moreover, the single-packet design requires the entire authentication payload to be reconstructed at once, increasing the risk of dropped or delayed packets and necessitating retransmission logic on the client side. Server-side processing overhead also increases, as all incoming SPA packets must be decrypted and verified in full.

6.3 Advanced PAPC Mechanisms: Protocol-Independent Cryptographic Port Sequence

The proposed PAPC scheme builds upon the lessons learned from PK and SPA by introducing a lightweight, protocol-independent mechanism that distributes encrypted authentication messages across a sequence of destination port numbers. This design removes the dependency on specific transport-layer protocols, enabling operation in both TCP and UDP environments.

Each port sequence fragment encodes part of an AES-128 encrypted message that includes a nonce, timestamp, and HMAC-SHA256 signature, ensuring confidentiality, integrity, and replay protection.

PAPC improves scalability by passively monitoring port attempts without opening service ports or requiring application-layer interactions. The randomness in port assignment, derived from a signature component, further obfuscates the authentication message.

Unsafe ports are dynamically avoided to enhance robustness. Experimental findings confirm that PAPC reduces overhead by requiring selective port sequence validation instead of full packet inspection, and dynamically adjusts firewall policies only after successful authentication.

6.4 Gaps in Existing Approaches and PAPC's Mitigations

Both PK and SPA fall short in meeting the demands of modern AIoT environments, which require low-latency, lightweight, and flexible authentication mechanisms. PK suffers from a complete lack of cryptographic integrity, making it inherently insecure in hostile network environments.

SPA, while more secure, depends heavily on dedicated client software and the transmission of UDP packets, which limits its compatibility in NAT-constrained or protocol-restricted networks.

The proposed PAPC scheme addresses these shortcomings through a combination of protocol and client independence, robust cryptographic integration, and operational efficiency. Specifically, PAPC's protocol independence enables its deployment across restrictive or hybrid network configurations without requiring transport-layer modifications. Its client-independent design allows users to initiate authentication through standard web or mobile interfaces, eliminating the need for special-purpose software.

Furthermore, the scheme leverages nonce-timestamp validation and HMAC to ensure strong resistance against replay and man-in-the-middle (MITM) attacks. Unlike SPA, PAPC minimizes server overhead by validating only targeted port sequence, rather than processing all incoming encrypted traffic.

These improvements collectively enable PAPC to offer a practical, secure, and scalable authentication method that aligns with the dynamic requirements of AIoT ecosystems while preserving system performance and network compatibility.

6.5 Comparative Security Analysis

Table 7 summarizes the comparative strengths of PAPC against PK and SPA in terms of cryptographic robustness, replay and MITM resistance, and adaptability in NAT environments.

Table 7: Comparison of security features in PK, SPA, and PAPC

Security feature	Port knocking (PK)	Single packet authorization (SPA)	Proposed PAPC
Cryptographic protection	None	HMAC + Encryption	HMAC + AES-128 Encryption
Replay attack prevention	Weak	Strong	Strong (Nonce + Timestamp)
Resistance to MITM attacks	Weak	Moderate	Strong (HMAC-SHA256)
Port sequence predictability	High	Not applicable (Single Packet)	Low (Randomized Sequence + Signature)

(Continued)

Table 7 (continued)

Security feature	Port knocking (PK)	Single packet authorization (SPA)	Proposed PAPC
Server overhead	Low	High	Moderate (Selective Port Processing)
Applicability in NAT environments	High	Low	High

6.6 Applicability in AIoT Environments

AIoT plays a critical role in realizing scalable services in next-generation living environments such as smart cities, by enabling secure and efficient system designs including actuator-level monitoring and human-centric architectures [31].

The PAPC scheme demonstrates notable adaptability for AIoT-based authentication, primarily due to its lightweight design and protocol independence. In such AIoT environments, where devices are often resource-constrained, authentication mechanisms must provide robust security while maintaining minimal computational overhead.

The proposed scheme addresses this requirement by avoiding persistent connections and reducing reliance on computationally intensive cryptographic operations, thereby ensuring low processing overhead. Furthermore, the protocol independence of PAPC allows for seamless integration into a wide range of IoT communication architectures without necessitating modifications to existing network protocols.

This flexibility enhances its applicability in diverse and constrained deployment scenarios typical of AIoT systems.

6.7 Limitations and Future Research Directions

While the proposed scheme offers several security and efficiency improvements, certain limitations remain.

6.7.1 Computational and Network Overhead

Despite the optimization of the proposed authentication process, certain levels of computational and network overhead remain inevitable due to the cryptographic operations and the management of port sequence.

These processes, although efficient, may still pose challenges in highly resource-constrained environments. Future research should focus on improving the efficiency of cryptographic key exchanges to reduce overall processing time.

Additionally, a thorough evaluation of PAPC's performance on low-power AIoT devices is necessary to ensure the scheme's suitability for deployment in such constrained systems.

6.7.2 Dynamic Threat Adaptation

As AI-driven cyberattacks become increasingly sophisticated, authentication mechanisms must evolve to counter emerging threats in real time [32]. Future research directions include the integration of AI-based anomaly detection techniques to identify suspicious or abnormal authentication behaviors [33,34].

Moreover, dynamic reconfiguration of port sequence, based on real-time threat intelligence and environmental factors [35], may further enhance the adaptability and resilience of the PAPC scheme against evolving attack vectors.

6.7.3 Extended Real-World Testing

The current evaluation was conducted in a virtualized environment, which may not fully capture real-world network variabilities such as packet loss, congestion, and dynamic routing changes. Further testing under diverse network conditions is necessary to refine the proposed approach and validate its robustness.

6.8 Summary of Key Contributions

The proposed PAPC scheme offers several significant contributions to the field of secure authentication. First, it enhances security by leveraging cryptographic port sequence, which effectively mitigate well-known vulnerabilities associated with traditional Port Knocking (PK) and Single Packet Authorization (SPA) techniques.

Second, it demonstrates improved adaptability by supporting deployment in a wide range of environments, including AIoT and blockchain-based systems, where lightweight and flexible authentication mechanisms are essential.

Lastly, the scheme exhibits strong scalability and protocol independence, enabling seamless integration into existing network infrastructures without the need for modifications to underlying communication protocols.

These characteristics collectively position PAPC as a viable and efficient alternative to conventional authentication methods in modern, heterogeneous network environments.

These findings reinforce the viability of PAPC as a robust alternative to existing authentication methods, providing strong security while maintaining operational efficiency across various network environments.

7 Conclusion

Pre-Authentication and Post-Connection (PAPC) schemes are essential for enforcing the Zero Trust security model by ensuring authentication before any exposure of network resources. This paper introduced a novel PAPC mechanism that transmits encrypted authentication data through destination port sequences, enabling protocol-independent, clientless authentication suitable for browser-based and AIoT environments. By applying Advanced Encryption Standard (AES) encryption, Hash-Based Message Authentication Code (HMAC) validation, and a nonce-based replay prevention mechanism, the proposed method ensures confidentiality, integrity, and freshness of authentication data while minimizing server-side processing burden. Compared to Port Knocking (PK) and Single Packet Authorization (SPA), this approach enhances compatibility, avoids payload-based transmission, and reduces dependency on specific protocols or privileged client configurations.

Through a functional web-based prototype, the proposed scheme demonstrated feasibility in lightweight environments, showing potential for integration with machine learning-based anomaly detection. Its modular design allows flexible deployment across AIoT, edge, and decentralized networks, where traditional authentication methods fall short. The PAPC mechanism's lightweight, scalable, and extensible design makes it a strong candidate for Zero Trust implementations in cloud-native and distributed systems.

Despite these strengths, some limitations remain. Future research should address performance optimization of cryptographic operations and evaluate robustness under real-world conditions including packet loss, congestion, and dynamic routing. Further work may also explore adaptive port sequence generation and AI-driven threat detection to strengthen resilience against evolving cyberattacks. The proposed PAPC

scheme lays a foundation for the next generation of authentication frameworks, offering a practical, secure, and scalable alternative to existing solutions.

Acknowledgement: The author would like to thank the journal editor for inviting him to write this review paper. He would also like to thank Chungnam National University along with IITP and MSIT for their support.

Funding Statement: This work was supported by Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2022-II221200, Convergence Security Core Talent Training Business (Chungnam National University).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing—original draft, Sunghyun Yu; Funding acquisition, Project administration, Supervision, Writing—review & editing, Yoojae Won. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Ethics Approval: Not applicable.

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

Abbreviations

AES	Advanced Encryption Standard
AH	Accepting Host
AIoT	Artificial Intelligence of Things
CBA	Connection Before Authentication
DDoS	Distributed Denial of Service
DoS	Denial of Service
HMAC	Hash-based Message Authentication Code
IANA	Internet Assigned Numbers Authority
IDS	Intrusion Detection Systems
IH	Initiating Host
IoT	Internet of Things
IV	Initialization Vector
KMS	Key Management System
MITM	Man-In-The-Middle
ML	Machine Learning
mTLS	Mutual TLS
NAT	Network Address Translation
OTP	One-Time Password
PAPC	Pre-Authentication and Post-Connection
PK	Port Knocking
PSK	Pre-Shared Key
RNG	Random Number Generator
SMS	Short Message Service
SPA	Single Packet Authorization
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol

References

1. Khan S, Shiraz M, Boroumand L, Gani A, Khan MK. Towards port-knocking authentication methods for mobile cloud computing. J Netw Comput Appl. 2017;97(3):66–78. doi:10.1016/j.jnca.2017.08.018.
2. Fang W, Zhu C, Zhang W. Toward secure and lightweight data transmission for cloud-edge–terminal collaboration in artificial intelligence of things. IEEE Internet Things J. 2023;11(1):105–13. doi:10.1109/jiot.2023.3295438.
3. Veeramachaneni V. Integrating zero trust principles into IAM for enhanced cloud security. Recent Trends Cloud Comput Web Eng. 2025;7(1):78–92.
4. Gambo ML, Almulhem A. Zero trust architecture: a systematic literature review. TechRxiv. 2025. doi:10.36227/techrxiv.173933211.18231232/v1.
5. Zhang H, Zhang Z, Chen L. Toward zero trust in 5G industrial internet collaboration systems. Digit Commun Netw. 2025;11(2):547–55. doi:10.1016/j.dcan.2024.03.011.
6. Kindervag J. Build security into your network's DNA: the zero trust network architecture. Cambridge, MA, USA: Forrester Research; 2010.
7. National Institute of Standards and Technology (NIST). Zero trust architecture. NIST Special Publication 800-207. 2020 [Internet]. [cited 2025 May 19]. Available from: https://doi.org/10.6028/NIST.SP.800-207.
8. Ward R, Beyer B. BeyondCorp: a new approach to enterprise security. Login. 2014;39(6):6–11.
9. Kindervag J, Balaouras S. No more chewy centers: introducing the zero trust model of information security. For Res. 2010;3:56682.
10. Selvarajan S, Manickam S, Manoharan H, Laghari SUA, Uddin M, Abdelhaq M, et al. Testing and substantiation of zero trust devices with blockchain procedures for secured data transfer approach. Hum Cent Comput Inf Sci. 2024;14:1–16.
11. Arif T, Jo B, Kang J, Park JH. Blockchain-enabled IDPS and federated learning for enhancing CPS security against advanced persistent threats in zero trust architectures. Hum Cent Comput Inf Sci. 2025;15:22. doi:10.22967/HCIS.2025.15.022.
12. Lucion ELR, Nunes RC. Software defined perimeter: improvements in the security of Single Packet Authorization and user authentication. In: 2018 XLIV Latin American Computer Conference (CLEI); 2018 Oct 1–5; São Paulo, Brazil. p. 708–17.
13. Krmelj GR, Pančur M, Grohar M, Ciglarič M. Open SPA—An open and extensible protocol for single packet authorization. In: Proceedings of the Central European Cybersecurity Conference 2018 (CECC 2018); 2018 Nov 15–16; Ljubljana, Slovenia. p. 1–6.
14. Krzywinski M. Port knocking-network authentication across closed ports. SysAdmin Magazine. 2003;12(6):12–7.
15. Isabel D. Port knocking: beyond the basics. GIAC Security Essentials Certification (GSEC). SANS Institute; 2005 [Internet]. [cited 2025 May 19]. Available from: https://www.giac.org/research-papers/1634/.
16. Vasserman EY, Hopper N, Tyra J. Silent knock: practical, provably undetectable authentication. Int J Inf Secur. 2009;8(2):121–35. doi:10.1007/s10207-008-0070-1.
17. Srivastava V, Kumar N, Mahapatra R. Advanced port knocking authentication scheme with QRC using AES. In: 2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC); 2011 Apr 22–24; Udaipur, India. p. 159–63.
18. Al-Bahadili H, Hadi AH. Network security using hybrid port knocking. Int J Comput Sci Netw Secur. 2010;10(8):8.
19. deGraaf R, Aycock J, Jacobson M. Improved port knocking with strong authentication. In: Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC 2005); 2005 Dec 5–9; Tucson, AZ, USA. p. 451–62. doi:10.1109/CSAC.2005.32.
20. Cisco Systems. Well-known ports and applications [Internet]. Cisco TechNotes. [cited 2025 Jan 23]. Available from: https://www.cisco.com/c/en/us/support/docs/security/firewalls/ports.html.
21. Internet Assigned Numbers Authority (IANA). Service name and transport protocol port number registry [Internet]. [cited 2025 Jan 23]. Available from: https://www.iana.org/assignments/service-names-port-numbers.
22. Rash M. Single packet authorization with fwknop. Login: USENIX Mag. 2006;31(1):63–9.
23. Cloud Security Alliance (CSA). Software-Defined Perimeter (SDP) specification v2.0. CSA research; 2022 [Internet]. [cited 2025 May 20]. Available from: https://cloudsecurityalliance.org.

24. Cloud Security Alliance (CSA). SDP as a DDoS Defense Mechanism. CSA Research; 2019 [Internet]. [cited 2025 May 20]. Available from: https://cloudsecurityalliance.org.
25. Campbell B, Bradley J, Lodderstedt T, Sakimura N, Jones M. RFC 8705: OAuth 2.0 mutual-TLS client authentication and certificate-bound access tokens. Bangkok, Thailand: Internet Engineering Task Force (IETF); 2020.
26. Køien GM. A brief survey of nonces and nonce usage. In: SECURWARE 2015: The Ninth International Conference on Emerging Security Information, Systems and Technologies; 2015 Aug 23–8; Venice, Italy. p. 85–91.
27. Yu S, Won Y. Enhanced Single Packet Authorization system via dynamic input port allocation using one-time passwords. In: Advances in Computer Science and Ubiquitous Computing (Proceedings of CUTE/CSA 2023). Singapore: Springer Nature Singapore; 2023. p. 318–24.
28. Xu Y, Cao Z, Ma X, Lu J. AHAC: an advanced network-hiding access control framework based on obfuscated single packet authorization for zero trust network environments. Appl Sci. 2024;14(13):5593.
29. Junquera-Sánchez J, Cilleruelo C, de-Marcos L, Martinez-Herráiz J-J, Wu D. C-Lock: local network resilient port knocking system based on TOTP. Wirel Commun Mob Comput. 2022;2022(157):1–9. doi:10.1155/2022/9153868.
30. Wang Y, He C, Xi Z, Zhang B, Zhang T. Dynamic anonymous access control based on dual-mode single-packet authentication. Preprints. 2025; 2025010527. doi:10.20944/preprints202501.0527.v1.
31. Alwakid G, Humayun M, Ahmad Z, Shaheen M. Development of a sustainable and optimized energy ecosystem with real-time monitoring through innovative IoT-based actuator design for human-centric security in smart cities. Hum Cent Comput Inf Sci. 2025;15:22. doi:10.22967/HCIS.2025.15.002.
32. Wang BX, Chen JL, Yu CL. An AI-powered network threat detection system. IEEE Access. 2022;10(12):54029–37. doi:10.1109/access.2022.3175886.
33. DeMedeiros K, Hendawi A, Alvarez M. A survey of AI-based anomaly detection in IoT and sensor networks. Sensors. 2023;23(3):1352. doi:10.3390/s23031352.
34. Goswami M. AI-based anomaly detection for real-time cybersecurity. Int J Res Rev Tech. 2024;3(1):45–53.
35. Tounsi W, Rais H. A survey on technical threat intelligence in the age of sophisticated cyber attacks. Comput Secur. 2018;72(3):212–33. doi:10.1016/j.cose.2017.09.001.