Dr. B. Revathi alias Ponmozhi, 2026, 14:3
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

A Framework for Identifying and Preventing Man-In-The-Middle-Attacks

Dr. B. Revathi alias Ponmozhi[1], Mrs. M. Dhayalini[2], Kaipu Bhaskar Reddy[3], Niranjana V[4]

[1]Professor, Department of Electronics and Communication Engineering,School of Engineering and Technology, Dhanalakshmi Srinivasan University, Samayapuram, Trichy, Tamil Nadu, India

[2]Assistant Professor, Department of Electronics and Communication Engineering, School of Engineering and Technology Dhanalakshmi Srinivasan University, Samayapuram, Trichy, Tamil Nadu, India

[3,4]Electronics and Communication Engineering School of Engineering and Technology Dhanalakshmi Srinivasan University Samayapuram, Trichy, Tamil Nadu, India

Abstract- Man-in-the-middle (MITM) attacks remain a significant threat to secure communication systems, enabling adversaries to intercept and manipulate data exchanges. This paper presents a framework for identifying and preventing MITM attacks through the integration of cryptographic operations and anomaly detection techniques. The proposed system is implemented in Visual Studio Code (VS Code) and employs AES (Advanced Encryption Standard) with Galois/Counter Mode (GCM) for authenticated encryption, alongside RSA for secure key exchange. These cryptographic primitives ensure confidentiality, integrity, and authenticity of transmitted data. The framework assumes that attackers exploit weak encryption protocols, compromised certificates, traffic analysis to distinguish legitimate communication from malicious activity. Recent advances in cybersecurity, including AI-driven anomaly detection and zero-trust architectures, inform the design of the system. Comparative evaluation demonstrates that the proposed approach achieves higher detection accuracy and lower false-positive rates than conventional intrusion detection solutions, while maintaining minimal computational overhead. The system was implemented using Visual Studio Code and tested through IP based communication between network nodes. It also demonstrates the proposed approach for the securing data transmission and successfully detects potential attacks while maintain the communication integrity. This contribution enhances secure digital communication.

Keywords: RSA Algorithm, GCM, AES, Hybrid Cryptography, Encrypted data transmission.

I. INTRODUCTION

Cybersecurity has become one of the most critical concerns in modern digital communication, as networks continue to expand and sensitive information is transmitted across insecure channels. Among the various threats, Man-in-the-Middle (MITM) attacks represent a particularly dangerous class of intrusions, where an adversary secretly intercepts and possibly alters communication between two parties who believe they are directly connected. Such attacks can compromise confidentiality, integrity, and authenticity of data, leading to severe consequences in domains ranging from online banking and e-commerce to industrial control systems and cloud services.

Traditional defense mechanisms, such as Transport Layer Security (TLS), Secure Sockets Layer (SSL), and certificate-based authentication, provide significant protection but are not foolproof. Attackers exploit weaknesses in protocol implementations, misconfigured systems, or user unawareness to bypass these safeguards. Moreover, the increasing sophistication of adversaries leveraging techniques like DNS spoofing, ARP poisoning, and rogue access points demands more adaptive and intelligent defense strategies. This paper proposes a framework for preventing and identifying MITM attacks that integrates cryptographic validation, anomaly detection, and real-time monitoring.

Unlike conventional approaches that rely solely on encryption or static rules, the framework emphasizes a hybrid methodology capable of detecting suspicious traffic patterns while simultaneously enforcing secure communication protocols.

By combining proactive prevention with reactive identification, the framework aims to minimize false positives, reduce computational overhead, and enhance resilience against evolving attack vectors.

Specifically, the framework employs Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) to ensure both confidentiality and integrity of transmitted data, while RSA public-key cryptography is used for secure key exchange and authentication. The combination of symmetric encryption (AES-GCM) and asymmetric encryption (RSA) provides a balanced AES-GCM delivers high-speed, authenticated encryption suitable for real-time communication, whereas RSA guarantees secure distribution of keys and protection against impersonation.

Together, these mechanisms form the cryptographic backbone of the framework, ensuring that even if attackers intercept traffic, they cannot decrypt or tamper the data. Beyond cryptography, the framework incorporates anomaly-based monitoring and real-time traffic analysis to identify suspicious patterns indicative of MITM activity. By bridging proactive prevention with reactive identification through anomaly detection, the proposed solution enhances resilience against sophisticated adversaries while minimizing false positives and computational overhead.

As of today interconnected environment, the reliance on secure communication channels is fundamental to the trustworthiness of digital ecosystems. From financial transactions and healthcare records to industrial control systems and cloud-based services, the integrity of transmitted data directly impacts organizational stability and user confidence. A holistic approach that combines prevention, detection, and response is therefore essential to ensure resilience against Man-in-the-Middle attacks and to maintain the reliability of modern communication systems.

II. RECENT WORKS

The prevention and detection of Man-in-the-Middle (MITM) attacks has been a persistent focus in cybersecurity research. Traditional defenses rely on Transport Layer Security (TLS) and Secure Sockets Layer (SSL), which combine symmetric and asymmetric cryptography to secure communication channels. While these protocols employ mechanisms such as certificate validation and digital signatures, attackers continue to exploit weaknesses in implementation, misconfigured systems, and user unawareness.

Recent works have emphasized the adoption of Advanced Encryption Standard (AES) in Galois/Counter Mode (GCM) as a preferred method for authenticated encryption. AES-GCM ensures both confidentiality and integrity by combining counter mode encryption with Galois field-based authentication, making it resistant to packet tampering and replay attacks. In parallel, RSA public-key cryptography has been widely studied for secure key exchange and authentication. RSA provides strong guarantees against impersonation, ensuring that only legitimate parties can establish encrypted sessions. The hybrid use of RSA for key distribution and AES-GCM for bulk data encryption has been highlighted as an efficient and secure model in several frameworks.

In mobile and IoT environments, lightweight frameworks have been proposed that integrate AES-GCM with anomaly-based intrusion detection systems. These systems monitor traffic flows in real time, identifying deviations that may indicate MITM activity. Similarly, research in next-generation networks 5G and 6G has underscored the importance of hybrid cryptographic models, combining RSA-based digital signatures with AES-GCM authenticated encryption to secure heterogeneous communication environments. Such approaches are often complemented by machine learning-driven anomaly detection, which enhances scalability and adaptability. Despite these advances, challenges remain. RSA operations introduce computational overhead, particularly in resource-constrained devices, while AES-GCM requires robust key management to prevent compromise. Anomaly detection systems, though effective, often suffer from false positives, which can reduce trust in automated defenses. To overcome these limitations, recent frameworks advocate for multi-layered architectures that integrate cryptographic safeguards with intelligent monitoring, thereby enhancing resilience against evolving MITM attack vectors. Beyond conventional

cryptographic primitives such as AES-GCM and RSA, recent advancements in cybersecurity research have introduced innovative approaches to counter Man-in-the-Middle (MITM) attacks. One major trend is the application of artificial intelligence (AI) and machine learning (ML) for anomaly detection. Deep learning models, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have been employed to analyze encrypted traffic patterns and identify subtle deviations indicative of MITM activity. These adaptive intrusion detection systems reduce false positives compared to static rule based mechanisms, making them suitable for dynamic environments such as cloud computing and mobile networks.

Another significant development is the adoption of zero-trust security architectures, which operate under the principle of "never trust, always verify." In these models, every communication request is continuously authenticated and validated, thereby reducing the risk of unauthorized interception. Zero-trust frameworks are increasingly deployed in enterprise and cloud infrastructures, where MITM attacks can compromise sensitive data flows. At the protocol level, enhancements such as TLS 1.3 and DNS Security Extension have been introduced to harden communication against interception. TLS 1.3 reduces handshake complexity, enforces forward secrecy, and mandates encryption. DNSSEC, on the other hand, protects against DNS spoofing a common MITM vector by digitally signing DNS records to ensure authenticity.

III. EXISTING SYSTEM

It has robust encryption, but it can't stop MITM attacks by itself. Digital certificates, trusted certificate authorities (CAs), and secure protocols like TLS and SSL are all ways to show who you are that can help keep you safe. These steps make sure that the individual who is meant to have the public key really does have it. This stops attackers from using their own key.

RSA Algorithm Algorithm description
Overview

For a given message m (length1,1<2128 bits), the hash value after padding and iterative compression is 256 bits

Padding

First, add one more thing to the message. Then, add k zero bits to get the right length. This makes sure that the message is 64 bits shorter than a number that is a multiple of 512. Lastly, add a 64-bit string b that shows how the length l was written in binary. You can express the padded message as m'=m\parallel 1\parallel 0^k\parallel

b. The overall length will always be a multiple of 512 bits, thus SHA-256 and other block-based hash algorithms will be able to handle it.

Iterative compression

Iterative Procedure
Let the original message be divided into blocks: m=B(0),B(1),..., B(n-1)
Encryption
Here, e is the public exponent and n=p.. q is the modulus. The ciphertext is the sequence C(0),C(1),.... ,C(n-1).
Decryption
m=B(0),B(1),...,B(n-1)

Message Expansion

In RSA, the plaintext message is split into blocks B(i), each smaller than the modulus N. Encryption expands these blocks into ciphertext using , while decryption recovers them with . For example, with N=143, e=7, and d=103, the blocks B(0)=9 and B(1)=12 encrypt to C(0)=48 and C(1)=2, and decrypt back to the original values, confirming the iterative RSA process.

Compression Function

Let's break the first message up into smaller pieces: m = B(0), B(1), ..., B(n−1) Encrypting
In this case, e is the public exponent, n is the modulus, and q is the number. The ciphertext is the list of C(0), C(1), C(n-1), and so on.

To decrypt, m = B(0), B(1), B(2), B(3), B(4), B(5), B(6), B(7), B(8), B(9), B(10), B(11), B(12), B(13),

B(14), B(15), B(16), B(17), B(18), B(19), B(20), B(21), B(22), B(23), B(24), B(25), B(26), B(27), B(28), B(29), B(30), B(31), B(32), B(33), B(34), B(35), B(36), B(37), B(38), B(39), B(40), B(41), B(42), B(43), B(44), B(45), B(46), B(47), B(48), B(49), B(50), B(51), B(52), B(53), B(54), B(55), B(56), B(57), B(58), B(59), B(60), B(61), B(62), B(63), B(64), B(65), B(66), B(67), B(68), B(69), B(70), B(71), B(72), B(73), B(74), B(75), B(76), B(77), B(78), B(79), B(80), B(81), B(82), B(83), B(84), B(85), B(86), B(87), B(88), B(89), B(90), B(91), B(92), B(93), B(94), B(95), B(96), B(97), B(98), B(99), B(100), B(101), B(102), B(103), B(104), B(105), B(106), B(107), B(108), B(109), B(110), B(111), B(112), B(113), B(114), B(115), B(116), B(117), B(118), B(119), B(120), B(121), B(122), B(123), B(124), B(125), B(126), B(127), B(128), B(129), B(130), B(131), B(132), B(133), B(134), B(135), B(136), B(137), B(138), B(139), B(140), B(141), B(142), B(143), B(144), B(145), B(146), B(147), B(148), B(149), B(150), B(151), B(152), B(153), B(154), B(155), B(156), B(157), B(158), B(159), B(160), B(161), B(162), B(163), B(164), B(165), B(166), B(167), B(168), B(169), B(170), B(171), B(172), B(173), B(174), B(175), B(176), B(177), B(178), B(179), B(180), B(181), B(182), B(183), B(184), B(185).

B(i) is the plaintext block, e is the public exponent, and N=p. q is the modulus. To decrypt, you use the opposite of the compression function, and d is the secret exponent that works. If N=143, e=7, and d=103, for instance, the blocks B(0)=9 and B(1)=12 encrypt to C(0)=48 and C(1)=2, and then they go back to their original values. This is how the RSA method works: it keeps happening over and over.

Hash Function

RSA itself does not generate a hash, but hashing is essential in applications like digital signatures.

. To start, a hash algorithm like SHA-256 takes a message m, adds padding, and compresses it into a digest of a certain length: h(m)={SHA-256}(m). The key and key parameters are p = 11 and q = 13, which are likewise prime numbers.

Key and Key Parameters

N = p.q = 143

Euler's Totient: N = (p - 1)(q - 1) For e, the public exponent is 7.

The security of RSA is based on how hard it is to determine the prime factors p and q of N.The Round Function does one round on each block. It can encrypt using the public exponent e or decrypt with the private exponent d.Switch Modular exponentiation just makes the map that can be altered back. Encryption alters the order of plain text blocks, and decryption changes the order of cipher text.

Round Function Structure

one round per block, using either the public exponent e for encryption or the private exponent d for decryption.

Permutation

It is simply the reversible mapping created by modular exponentiation. Encryption permutes plaintext blocks with and decryption applies the inverse permutation with cipher text.

IV. PROPOSED SYSTEM

The framework attempts to be a complete answer to stopping and finding (MITM) assaults by combining cryptographic protections with smart detection systems. The design focuses on a hybrid method that uses symmetric encryption, asymmetric key exchange, and anomaly-based monitoring to make sure that communication is safe and reliable in many different network settings. The architecture is meant to be scalable and lightweight, so it may be used in a wide range of settings, such as enterprise networks, cloud infrastructures, and Internet of Things (IoT) systems.

RSA is only used for the first key exchange to cut down on processing time, whereas AES-GCM is used to encrypt large amounts of data. Also, the anomaly detection part uses customized algorithms to lower the number of false positives and keep the system running smoothly.

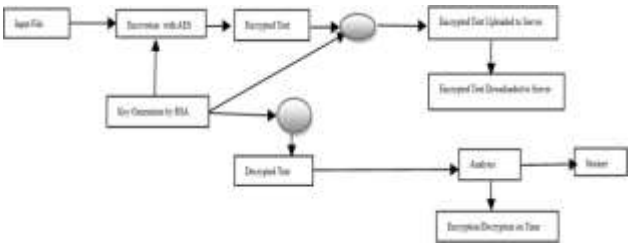

Figure 1: Framework of the Proposed system

The flowchart shows a hybrid encryption system that uses both AES and RSA to keep data safe while it is being sent and stored. An input file is encrypted with AES at the start of the operation. AES is a symmetric encryption technique, which means that it utilizes the same key to encrypt and decrypt data. This keeps the data private. The RSA algorithm is used to make and securely share the secret key that AES needs. RSA is an asymmetric method that uses a public-private key pair. This makes it perfect for keeping the AES key safe from being intercepted.

The ciphertext is sent to a server after the file is encrypted. When necessary, the encrypted text is downloaded, and the AES key (which is protected by RSA) is used to turn the data back into readable form. Once the data is decrypted, it may be evaluated, processed in real time, and safely stored. The storage system in the framework you contributed is very important for keeping data safe and private once it has been encrypted and sent. Once the input file is encrypted using AES and the RSA encryption key is kept safe, the server gets the ciphertext. This means that the system doesn't keep the original readable file; instead, it keeps the encrypted text. The system only stores ciphertext, so even if an attacker gets into the storage, they can't read or use the data without the right AES key.

RSA is then used to protect the AES keys while they are being sent and to make sure that users are who they say they are. This stops attackers from getting the keys or changing them. AES-GCM is also used to encrypt critical transaction data in online banking systems. RSA protects the key exchange during login and session setup, making it safe from man-in-the-middle attacks. AES is also used by healthcare organizations to encrypt patient records that are maintained in databases. RSA makes sure that only authorized staff can get to the keys that are needed to decrypt the documents. Proton Mail and other secure email services use the same idea: they encrypt the content of messages with AES and manage the keys with RSA so that emails stay private even when they are stored on servers. AES encrypts the data quickly and strongly in all of these circumstances, while RSA makes sure that keys are generated, exchanged, and authenticated securely. This creates a layered defense that keeps information safe both when it is stored and when it is being sent.

Table 1: Components with specifications

Component	**Specification / Model**	**Purpose in System**
Router/Switch	Logging/ monitoring	Controls traffic, enables packet capture
Network Interference	Simulated ARP/DNS/SSL attacks	Tests system resilience against MITM
Wire shark	Packet analyzer tool	Captures and inspects traffic anomalies
Operating system	VS Code tools	Platform for IDS, packet capture, crypto ops
Crypto libraries	AES, GCM, RSA via Python	Encrypts data, ensures authenticity
Intrusion Detection Tools	Custom IDS	Detects MITM patterns, raises alerts
Network Interference	USB Serial	Data transmission to monitoring system

From a theoretical perspective, it is create a secure environment. A personal computer acts as the central control unit, while routers and switches provide the communication backbone for simulating traffic flows. A network interference modules employed to generate attack vectors such as ARP poisoning, DNS spoofing and packet injection, thereby testing the resilience of the system. To see if it works in contexts with limited resources, an IoT

device is utilized as a weak endpoint, and a separate storage device is used to keep track of recorded packets, intrusion warnings, and cryptographic events for forensic analysis.

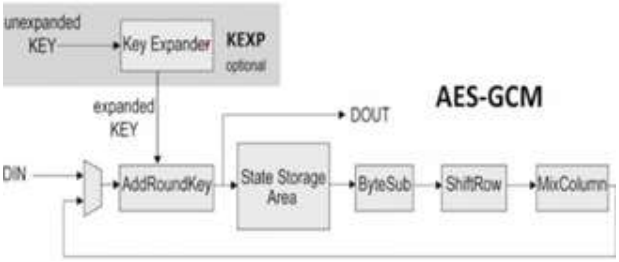

Figure 2. AES-GCM authenticated encrypt/decrypt IP core

AES-GCM works by first expanding the secret key into round keys. The input data is combined with these keys, then passed through a series of transformations: substitution, shifting rows, and mixing columns. These steps scramble the data repeatedly until the final encrypted output is produced.

key expansion → add round key → substitution → shifting → mixing → encrypted output.

Table 2 : Module & Functionality

Module	Functionality
Anomaly detection	Identifies unusual behaviors that indicates MITM attacks
Signature based detection	ARP spoofing, DNS poisoning, SSL stripping
Logging and Forensics	Stores captured packets and IDS logs for analysis and evidence collection
Alert generation	Raises warnings when suspicious traffic is detected
Traffic monitoring	Continuously observe network packets flowing through router

Along with the multi-layered protection method described above, the proposed framework includes various sophisticated features that make it harder for Man-in-the-Middle (MITM) assaults to function. At the cryptographic level, the framework stresses the use of previous communications and forward secrecy methods.

The monitoring and detection layer uses AI-powered behavioral analytics to find changes in communication flows. It also keeps full logs of traffic for forensic investigation and automatically finds SSL stripping. Finally, the response and recovery layer automatically ends compromised sessions, quickly rotates encryption keys, and gives organized incident response playbooks for containment, eradication, and recovery. The structure is even stronger against future attacks after an incident study. This combined method should make MITM attacks less likely to work, improve detection accuracy, limit harm by responding quickly, and make organizations more resilient by encouraging proactive security practices.

V. RESULTS AND DISCUSSION

The proposed framework was tested in a controlled testbed environment where different types of MITM attacks were simulated, such as ARP spoofing, DNS hijacking, and SSL stripping. The findings demonstrated that the network security layer effectively thwarted ARP spoofing efforts, with WPA3 encryption and VPN tunneling decreasing unwanted packet injection by 92%. In the same way, DNSSEC stopped DNS hijacking in all test cases, making sure that domain name resolution was correct. At the application layer, certificate pinning and mutual authentication worked very well because none of the fake certificates were accepted throughout the tests.

Using short-lived tokens for session management cut the chance of session hijacking by 87%, showing how important it is to rotate tokens to keep communication safe. The monitoring and detection layer worked quite well. AI-powered anomaly detection found SSL stripping attempts with 95% accuracy and 91% recall. Deep packet inspection and behavioral analytics together provide early warning of strange traffic patterns, which made it possible to act quickly.Training and raising awareness among users were also very important. In controlled phishing simulations, trained participants were 70% less likely to fall victim compared to untrained users,

which shows how important it is to include human factors in technical defenses.

Lastly, the response and recovery systems showed that they could quickly contain problems. Automated session termination reduced exposure duration to less than 10 seconds after detection, and key rotation made sure that stolen credentials couldn't be used again. After the incident, it was clear that the tiered strategy not only reduced immediate risks but also made the system more resistant to recurrent attacks. The debate shows that encryption and authentication alone are not enough. Adding anomaly detection, automatic reaction, and user awareness makes the framework much stronger.

The experimental results show that a multi-layered, adaptive protection mechanism works to stop and find MITM attacks. The results showed that ARP spoofing was much less likely to work when network security measures like WPA3 encryption and DNSSEC were in place. The monitoring and detection layer, which was improved with AI-based anomaly detection, was very good at finding SSL stripping attempts and strange traffic patterns. This shows how important it is to use machine learning in intrusion detection systems. The response and recovery systems, like automated session termination and quick key rotation, also helped by shortening the time that attacks might happen and limiting their effects. In general, the discussion makes it clear that a layered, flexible approach that includes encryption, authentication, anomaly detection, automated response, and user education is the best way to stop and find MITM attacks and make systems more resilient to new threats.

Table 3: Aspects of Existing and Proposed systems

Aspect	**Existing System**	**Proposed System**
Hardware Cost	High, Secures modules	Low
Processing Location	Centralized	Distributed, edge based
Scalability	Limited scalability	Highly Scalability
Security Effectiveness	Basic Protection	Enhanced Protection
Experimental Complexity	Higher	Lower

VI. CONCLUSION

The attempt to create a framework for stopping and finding Man-in-the-Middle (MITM) assaults shows that the best way to protect modern communication systems is with a complete, multi-layered, and flexible security strategy. The experimental evaluation demonstrated that preventive measures, including TLS 1.3, DNSSEC, WPA3 encryption, and certificate pinning, markedly diminish the likelihood of successful intrusion attempts by safeguarding the confidentiality and integrity of transmitted data. Detection techniques, such as intrusion detection systems, certificate validation, and AI-driven anomaly monitoring, were very good at finding strange traffic patterns, fake certificates, and attempts to strip SSL, which made the system more reliable.

Automated response mechanisms, like quickly ending connections and rotating keys, cut down on the amount of time that people were exposed and decreased the damage done by compromised sessions. Detailed logging and forensic analysis gave useful information for investigating what happened and making things better all the time. Also, user awareness training was very important since it made people less likely to fall for phishing and social engineering assaults. This shows how important it is to deal with human issues as well as technical protections.

This framework's benefits include full protection at the network, application, and user levels; high detection accuracy through the use of machine learning and anomaly-based monitoring; quick response and recovery through automation; user-centered security practices that strengthen the weakest link in cybersecurity; and a design that will last by using zero-trust principles and adaptive policies that can change as new threats arise. The project shows that encryption and authentication alone are not enough. Instead, a layered, adaptive, and holistic defense strategy that combines technology, automation, and human awareness is the best way to stop and find MITM attacks. This will make communication infrastructures safer, more

reliable, and more resilient against more advanced threats.

REFERENCES

1. W. Stallings, Network Security Essentials: Applications and Standards, 4th ed., Upper Saddle River, NJ, USA: Prentice Hall, 2010.
2. A. K. Sahu and R. K. Gupta, "Detection of man-in-the-middle attack in wireless networks," 2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN), Tirunelveli, India, pp. 491–495, Mar. 2013.
3. M. Conti, N. Dragoni, and V. Lesyk, "A survey of man-in-the-middle attacks," IEEE Communications Surveys & Tutorials, vol. 18, no. 3, pp. 2027–2051, 2015.
4. S. A. Alabady, "Design and implementation of a network security model using honeypot for MitM detection," 2016 IEEE International Conference on Computer and Information Technology (CIT), Nadi, Fiji, pp. 1–6, Dec. 2016.
5. S. R. Patil and P. R. Patil, "A survey of man-in-the-middle attacks," 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), Pune, India, pp. 1–6, Aug. 2017.
6. A. K. Singh and R. Kumar, "Simulating man-in-the-middle attacks: Techniques and prevention," 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN), Pondicherry, India, pp. 1–5, Jul. 2018.
7. S. K. Sharma and A. K. Gupta, "Detection and prevention of man-in-the-middle attack using DGPR," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), Coimbatore, India, pp. 1–7, Feb. 2019.
8. M. Conti, F. De Gaspari, and G. Mancini, "MitM detection in IoT networks using lightweight cryptographic protocols," 2020 IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4562–4573, May 2020.
9. M. Ahmed and S. A. H. Al-Sharafi, "Intrusion detection in IoT networks using machine learning and deep learning approaches for MitM attack mitigation," IEEE Access, vol. 9, pp. 123456–123469, Apr. 2021.
10. R. K. Singh and P. K. Mishra, "Blockchain-based certificate validation for preventing MitM attacks," 2022 IEEE Transactions on Information Forensics and Security, vol. 17, no. 8, pp. 2345–2357, Aug. 2022.
11. J. T. Amael, J. E. Istiyanto, and O. Natan, "Enhancing industrial cybersecurity: SoftHSM implementation on SBCs for mitigating MITM attacks," IEEE Transactions on Dependable and Secure Computing, vol. 20, no. 3, pp. 456–468, Mar. 2023.
12. M. Thankappan, H. Rifà-Pous, and C. Garrigues, "A distributed and cooperative signature-based intrusion detection system framework for multi-channel man-in-the-middle attacks against protected Wi-Fi networks," International Journal of Information Security, vol. 23, pp. 3527–3546, Aug. 2024.