SAP: A Low-latency Protocol for Mitigating Evil Twin Attacks and High Computation Overhead in WI-FI Networks

Vineeta Jain, Ulf Wetzker, Vijay Laxmi, Manoj Singh Gaur, Mohamed Mosbah, and Dominique Mery

Abstract—This paper discusses the detrimental effects of Evil Twin (ET) attacks on wireless networks and analyzes the security offered by existing 802.11 protocols against ET attacks. There has always been a direct correlation between attacks and the improvement of protocol standards. As the sophistication of attacks increases, protocol standards tend to move towards higher security, resulting in a significant increase in both latency and computational overhead and serious degradation in the performance of low-latency applications. A protocol is needed to provide optimum security against attacks while maintaining the latency and overhead. In this work, we propose a secure and low-latency protocol known as SAP, which uses Elliptic Curve Cryptography (ECC) to ensure mutual authentication and security from network attacks. SAP exchanges fewer messages, incurs low computation overhead, and yet manages to provide end-to-end security to messages exchanged, making it ideal for embedded and IIoT applications. We prove the security aspects of SAP by formal security analysis using the widely-accepted Scyther tool and application-oriented aspects by using the well-established OMNeT++ simulator. Finally, we perform a comparative analysis of SAP with existing IEEE 802.11 wireless network protocols.

Index Terms—wireless networks, security, low-latency, evil twins, elliptic curve cryptography, IIoT

◆

1 INTRODUCTION

Nowadays, wireless networks have become one of the ubiquitous and fastest means of accessing the Internet across the globe. According to the Cisco Visual Networking Index for $2017-2022$ [1], global IP (Internet Protocol) traffic is expected to reach $4.8ZB$ by the end of 2022. Out of $4.8ZB$, 71% is predicted to be received from wireless networks [1]. This is mainly due to the mobility and flexibility offered by wireless networks over the wired ones.

Due to the essential advantages of wireless networks, wireless communication systems are already ubiquitous in the automotive, medical, military and IIoT (Industrial Internet of Things) sectors. In medical technology, wireless networked devices, such as electronic medical records, physiological monitoring devices (wearables) or diagnostic equipment, are already being used in large numbers. In vehicles, the use of wireless communication is not limited to infotainment systems. Vehicular communication supports the driver via computer-assisted safety warnings and traffic information from external sources and inside the vehicle sensor information is transmitted wirelessly for condition monitoring. With the digitalization of the automation industry, the number of wirelessly networked devices has increased significantly. Rigid structures such as conveyor belts and overhead cranes are increasingly being replaced by intelligent, automated guided vehicle (AGVs). Augmented reality (AR) supports the worker in carrying out individual work steps by providing work and safety instructions via wirelessly connected data glasses. With the increase in data volume-dependent costs and international roaming charges on $4G$ networks, the popularity of public free Wi-Fi networks has increased for the use of data, voice and video services, resulting in deployment of APs (Access Points) in public places such as airports, railway stations, cafes/restaurants, etc. Most of the above applications have very high security and availability requirements [2] that must be met by the communication system.

Although there is a tremendous increase in wireless network usage, mechanism for the reauthentication between APs and clients[1] remain vulnerable. We define *authentication* as the first attempt of the client to get authenticated to an AP. Any subsequent authentication attempts between a client and an AP is defined as *reauthentication*. Whenever a client associates with an AP for the first time, it makes a unique cache entry for that AP based on its SSID (Service Set Identifier). For future connections with the AP, the client just compares the SSID of the AP with the cached entry, which is easy to spoof. The attackers exploit this feature of wireless networks to launch ET (Evil Twin) attacks.

An ET is a rogue AP (Access Point) deployed by attackers to mimic the genuine characteristics of legitimate AP in a network zone [3]. In ET attack, the attacker impersonates genuine AP by spoofing its SSID (and often BSSID or MAC address) to launch ET in the network. When the attacker arrives in the vicinity of the target SSID, it starts transmitting

---

Vineeta Jain is with Fraunhofer Institute of Integrated Circuits IIS-EAS, Germany & LNM Institute of Information Technology Jaipur, India (email: vineeta.jain@eas.iis.fraunhofer.de)
Ulf Wetzker is with Fraunhofer Institute of Integrated Circuits IIS-EAS, Germany (email: ulf.wetzker@eas.iis.fraunhofer.de)
Vijay Laxmi and Manoj Singh Gaur are with the Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur, India (e-mail: vlaxmi@mnit.ac.in; gaurms@mnit.ac.in).
Mohamed Mosbah is with LaBRI, CNRS, Bordeaux INP, University of Bordeaux, Talence, France (email: mohamed.mosbah@labri.fr).
Dominique Mery is with LORIA & University of Lorraine (email: dominique.mery@loria.fr).

[1] In this paper, the client refers to the user's device.

beacon frames[2] with the same SSID to attract the victim. Since a profile of the genuine AP already exists in the client's device, the client directly tries to establish a connection to the ET. The attacker can also break the existing connection between clients and genuine AP by launching deauthentication attack[3] to momentarily disconnect the client and AP. Further, the attacker can use power amplifiers and high gain antennas to broadcast SSID with higher signal strength, resulting in connection establishment between client and ET. Consequently, the attacker can intercept network traffic flowing through the device and push malicious payloads like malware or launch more sophisticated attacks on the victim's device. Fig. 1 shows the ET attack scenario.

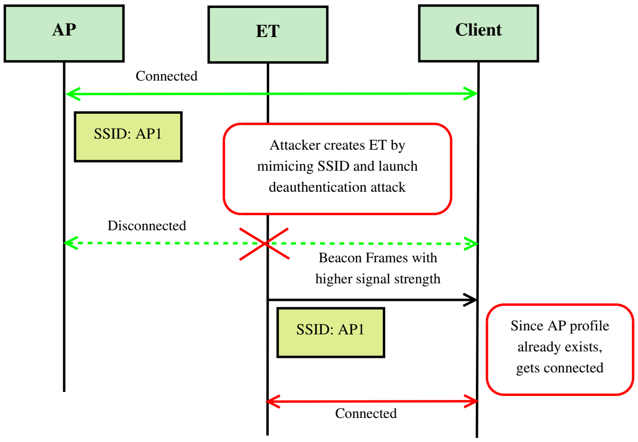

AP
ET
Client
Connected
SSID: AP1
Attacker creates ET by mimicing SSID and launch deauthentication attack
Disconnected
Beacon Frames with higher signal strength
SSID: AP1
Since AP profile already exists, gets connected
Connected

Fig. 1: ET Attack Scenario

ETs can be launched either by employing software APs on laptops or using embedded hardware or creating hotspot through mobile devices [3]. They can be easily deployed anywhere and remain undetected by the owners. As per an experiment conducted by Trend Micro^{TM} in London city in 2013, users do not even check the network's name before connecting and use such connections for online activities such as banking, social networking, surfing, etc. This additionally opens the possibility for Wi-Fi-based phishing attacks. Launching ET attacks in medical, military or industrial environments for sniffing or tampering the transmitted information could have serious consequences, including economic espionage, operational failure, physical damage, environmental harm, and injury or loss of life. In 2018, Russian Nation-State Spies used evil twins for eavesdropping sensitive information from anti-doping, nuclear reactions and chemical testing laboratories in various countries such as Brazil, Colorado, Switzerland and Netherlands [4]. Inadequate countermeasures and widely available attack tools, have made ET attacks one of the most basic attack surfaces in Wi-Fi networks.

1.1 Motivation

ET attacks are very prominent among security researchers and analysts due to the ease of deployment and serious consequences. Three types of countermeasures have been proposed: client-side, server-side, and protocol-modifying approaches. The client-side solutions rely on app-based approaches to detect ETs within a network. These solutions, are not guaranteed to be used by every user. The server-side solutions for detecting ETs require the installation of servers in all network segments. The nature of the countermeasures here is mostly left to the network administrators and turns out to be rather difficult in practice. This contrasts with solutions that modify the structure/security schemes of existing protocols because they are easy to distribute and deploy. This approach does not burden users or network administrators and is preferable to the other techniques in most cases. In this paper, we discuss the historical evolution of the standard protocols given by IEEE 802.11 to secure the network transmission from attacks.

For safety and security of user's information, IEEE 802.11 have launched several protocols:

1) **Wired Equivalent Privacy (WEP)** is the first protocol launched by IEEE 802.11 in 1997, which uses password-based authentication where password is a 40-bit static key already known to all the clients. Further, WEP applies Rivest Cipher 4 (RC4) stream cipher for encryption using 24-bit Initialization Vector (IV). Due to the unencrypted transmission of authentication messages, smaller key size and reuse of IVs [5], WEP is found vulnerable to ET, Man-in-the-middle (MITM) and replay attacks [6].
2) To overcome the shortcomings of WEP, IEEE 802.11 launched **WI-FI Protected Access (WPA)** protocol in 2003. The aim was to fix the limitations of WEP without upgrading the hardware. WPA uses password-based authentication, where the password is a passphrase also known as pre-shared key (PSK). WPA applies Temporal Key Integrity Protocol (TKIP) for encryption which uses the RC4 algorithm and introduces the concept of 4-way handshake after the authentication and association phases. In the 4-way handshake, all types keys used for encryption and transmission are generated using the PSK. Due to the use of RC4 for encryption and similar passphrase for all the clients, it is found vulnerable to offline dictionary attacks [7]. Once the attacker cracks the passphrase, setting up an ET is a cakewalk.
3) Further in 2004, IEEE 802.11 introduced **WPA2** protocol which is an improved version of WPA protocol. WPA2 uses Counter Mode with Cipher Block Chaining Message Authentication Code Protocol (CCMP) which utilizes Advanced Encryption Standard (AES) for encryption. In WPA2, both client and AP share a passphrase known as Pairwise Master Key (PMK), which is used to generate Pairwise Transient Key (PTK) for encrypting the user sessions. The usage of high-level encryption reduces the chance of cryptanalysis, but still the offline dictionary attack and ET attacks are possible [8].
4) In 2018, IEEE 802.11 has introduced **WPA3** protocol. For public networks, WPA3 uses opportunistic wireless encryption (OWE) mode also known as

---

[2] Beacon frame is a management frame broadcasted by APs to announce their capabilities in the network.

[3] Deauthentication attack is a targeted denial-of-service attack to disrupt communication between client and access point. Many open source softwares exist for launching deauth attack such as aireplay-ng module of aircrack-ng suite.

enhanced open WI-FI network [9]. In this mode, there is no pre-shared information between AP and client. Both the entities exchange their pubic keys to generate a shared secret key, i.e., PMK using Elliptic Curve Diffie-Hellman (ECDH) algorithm. The derived PMK is then utilized in the four-way handshake mechanism to generate session keys, i.e., PTK. Since OWE approach is based on Trust-on-first-use (TOFU) model where no validation of public keys is performed by any of the parties, launching ET attack is quite easy. For personal networks, WPA3 provides an extra layer of security (in addition to WPA2) in the form of simultaneous-authentication-of-equals (SAE) handshake (a variant of dragonfly handshake mechanism) and this standard is called as WPA3-personal. During SAE handshake, the passphrase shared between client and AP is converted into a high entropy key (PMK). Further, this key is used to produce PTK during four-way handshake mechanism. The computational overhead of WPA3-personal is very high given the complexity of the SAE handshake [10]. To provide legacy support, WPA3 supports transition to WPA2 which makes it vulnerable to downgrade and dictionary attack [10]. In this attack, an attacker tricks a client to get connected to an ET only supporting WPA2 mode and launches dictionary attack to catch the passphrase.

5) **802.1X** protocol is an IEEE Standard for Port-Based Network Access Control (PNAC) which uses unique certificates or credentials for every user to authenticate eliminating the reliability on single password for authentication. In addition to client and AP, 802.1X also requires a RADIUS[4] server and identity provider for authentication. The RADIUS server verifies the identity of a client by communicating with the identity provider (a directory containing user credentials/certificates information). Although 802.1X is the most secure protocol, it demands high-level expertise for the precise configuration and failing on that leads to security compromise. For instance, the RADIUS server identity verification by the clients is not a mandatory exercise in 802.1X [11]. As a result, many networks do not configure the same, leading to ET attacks as attackers can use a fake RADIUS server for authentication. Further, the credential-based authentication transmits the credentials unencrypted making them vulnerable to sniffing attack. Most of the organizations including small-scale industries and tertiary educational institutes (TEIs) still use credential-based 802.1X protocol without enforcing the optional RADIUS server identity verification making them vulnerable to ET and sniffing attacks. As per the study conducted in [12], out of 7045 TEIs across 56 contries, 86% are vulnerable to credential theft and ET attacks. The most secure 802.1X authentication is certificate based also known as EAP-TLS mode, where every user is provided a unique digital certificate for authentication, making it meticulous and expensive and thus, restricting its use mainly to large-scale industries. Although the EAP-TLS mode is secure against eavesdropping, the number of messages exchanged for authentication are way too high. It may lead to a delay in connection establishment process that can result in high latency in networks.

To comprehensively protect Wi-Fi networks against attacks, vulnerabilities in the IEEE 802.11 specification were addressed by new and improved security mechanisms. Unfortunately, each of the previously described protocols resulted in increased communication and computation overhead, making the authentication and reauthentication process more time-consuming. For a number of low latency applications, such as heath-care, intelligent transportation system, robotics, AR, etc., this is unacceptable as it would lead to significantly increased downtime during operation leading to unavailability. Due to increasing concerns about the latency of the IEEE 802.11 standard, many sectors have started to replace their wireless networks with private cellular networks [13]. This leads to an increase in their overall cost as private networks require greater upfront investment. Moreover, for contrained devices such as embedded and IIOT devices, the high computation overhead of security processes amounts to major source of latency. Therefore, a lightweight protocol is needed that provides optimal protection against current network attacks (such as ET, MITM, replay and sniffing attacks) while keeping latency and overhead as low as possible.

1.2 Contribution

To achieve the above requirements, we propose a low latency protocol named *Secure Authentication Protocol (SAP)* which provides security against contemporary network attacks through a secure authentication and reauthentication mechanism. SAP assures end-to-end security during the exchange of messages over the network by delivering one-to-one encrypted sessions between client and AP. There is no involvement of pre-shared knowledge or additional servers. SAP exchanges less number of messages, produces low computation overhead and thus, requires reduced bandwidth consumption. The formal security verification of the proposed protocol is performed using the widely-accepted Scyther [14] tool to evaluate SAP for security against network attacks. By conducting a comparative analysis of the proposed protocol with the previous IEEE 802.11 standard protocols, we have analyzed that SAP outperforms them in the aforementioned aspects. Finally, the practical demonstration of the proposed protocol is provided through the broadly-accepted OMNeT++ simulator [15].

1.3 Organization

The organization of the paper is as follows: Section 2 discusses the various types of keys and network assumptions followed by the proposed protocol, and provides a detailed description of the protocol. Section 3 theoretically analyzes the proposed protocol in various aspects such as security analysis and mutual authentication between client and AP. Section 4 evaluates the security aspects of the proposed protocol by formally verifying SAP using Scyther. Section

---

[4] It stands for Remote Authentication Dial-In User Service

5 explains the practical demonstration of SAP using OMNeT++ simulator. Section 6 compares SAP with the existing standard protocols. Section 7 concludes the paper.

2 THE PROPOSED PROTOCOL

To address the issues present in the existing standard protocols, we propose **S**ecure **A**uthentication **P**rotocol (SAP). SAP neither uses open nor password-based authentication. SAP employs Elliptic Curve Cryptography (ECC) to generate and exchange keys, and symmetric encryption scheme to encrypt transmitted messages. ECC is chosen because it has outperformed the existing key generation algorithms such as RSA, owing to its shorter key size and small computational overhead [16]. Short key size makes ECC faster and suitable for small and embedded devices. Further, SAP uses AES-CCMP (Advanced Encryption Standard Counter Mode with Cipher Block Chaining Message Authentication Protocol) [5] for symmetric encryption of the messages, as AES-CCMP provides a high level of security for encryption, used by all standard protocols (such as WPA2 and WPA3) and not been proved vulnerable to attacks [5]. By incorporating these cryptographic and encryption schemes, SAP assures mutual authentication, encrypted communication, secrecy against eavesdroppers and resistance to attacks.

2.1 Preliminaries

In this subsection, we present a concise description of Elliptic Curves and discuss the various types of keys and network assumptions followed by the proposed protocol.

2.1.1 Elliptic Curves

The elliptic curve over a finite field is defined by

$$y^2 = \{x^3 + ax + b\}\text{mod}\{p\} \tag{1}$$

It has domain parameters *(p, a, b, G, n, h)* where,

p = prime number specifying the size of finite field,
a, b = curve parameters,
G = Generator Point (generates a cyclic subgroup),
n = ord(G) (size of subgroup),
h=cofactor=$\frac{|E(Z/pZ)|}{n}$ (ideally 1), where $E(Z/pZ)$ represents elliptic curve defined over Z (integers) modulo p.

Suppose an elliptic curve is defined over integer modulo p as $E(Z/pZ)$ and $Q, P \in E(Z/pZ)$, where P and Q are points on the curve such that $P = kQ = Q + Q...k\ times$. According to **Elliptic Curve Discrete Logarithm Problem (ECDLP)**, the computation of P is simple when k and Q are known. However, given P and Q, the calculation of k is computationally challenging and expensive. This is the basis of Elliptic Curve Cryptography (ECC)[5] .

2.1.2 Keys used in SAP

The keys play a significant role in ensuring the security of the proposed protocol. The following keys are used in authentication and reauthentication phases of SAP:

- *Public-Private Key Pair*: AP produces a public-private key pair using ECC, and the public key of AP is known to everyone in the network.
- *Encryption-Decryption Key Pair*: Client produces encryption-decryption key pair using ECC. The functionality of encryption-decryption key pair is similar to public-private key pair in a way that the information encrypted by encryption key can only be decrypted by decryption key. But the difference is that unlike the public key, encryption key of client is not public in the network.
- *Master Key (MK)*: MK is uniquely generated by AP for each client and exchanged only once between the client and AP during their first connection attempt. MK is cached by both the parties as MK is used as a reauthentication parameter for further connection attempts between the client and AP.
- *Session Keys*: They are freshly produced for each session between client and AP, and used for encrypting the communication between them.

2.1.3 Assumptions

Following are the assumptions in proposed SAP protocol:

- The AP has a valid public key certificate[6] issued by a trusted and verified CA (Certification Authority).
- The key-pairs[7] for network entities are generated only once.
- The client and AP have sufficient storage and mechanism for MK Caching.
- The AP and client possess encoder/decoder to transpose an elliptic curve point into information.
- The protocol is public.
- The AP, attacker and client are in the same network.

The attacker possesses the following characteristics:

- The attacker can conduct active as well as passive attacks.
- The attacker has access to the public key of the AP and ECC domain parameters.
- Any authentic client in the network can be a target of the attacker.

2.2 Protocol Overview

SAP operates in two phases - *Registration* and *Authentication*. The registration phase is a one-time process borne by AP and client during their first association, whereas the authentication phase is a continuous process between client and AP whenever a new session begins. Table 1 represents the notations used in the proposed protocol.

---

[5] For more information on ECC, we recommend our readers to read [16].

[6] Public key certificate, also known as a identity certificate or digital certificate, is an electronic document issued by a CA (Certification Authority) to prove the ownership of a public key. It contains name of the certificate holder, public key of the holder and the digital signature of a CA for authentication.

[7] Public-private key pair for AP and encryption-decryption key pair for clients

TABLE 1
Notations with their descriptions used by SAP

S.No.	Notation	Description
1.	C	Client
2.	AP	Access Point
3.	n_c	Decryption key of client
4.	P_c	Encryption key of client
5.	n_{AP}	Private key of AP
6.	P_{AP}	Public key of AP
7.	ϕ_T	Point T on the elliptic curve
8.	m	Message m
9.	m'	Decrypted message m
10.	P_{MK}	Master Key (MK)
11.	K_{sk}	Seed key
12.	K_{se}	Session key
13.	T	Timestamp

2.2.1 Registration Phase

The registration phase is a one-time process which occurs when the client tries to connect to an AP for the first time. It consists of the following steps:

1) **Beacon Frame**: The AP broadcasts beacon frames in the network embedded with its public key certificate issued by a legitimate and verified CA containing the ECC domain parameters and public key of the AP.
2) **Probe Request:** The client, on receiving the beacon frame, verifies the certificate of the AP. It checks whether it implicitly trusts the certificate or it is trusted and verified by one of various CAs that it also implicitly trusts. If the client detects any problem in the certificate, i.e., either expired or hostname is different or not issued by any verified CA, it rejects the beacon and begin searching for new APs in the network. Else, it extracts the ECC domain parameters from the certificate and using them, the client chooses a decryption key n_c and produces an encryption key P_c as:

$$P_c = n_c G \tag{2}$$

Further, the client sends a probe request to the AP consisting of P_c and current timestamp value T_c by encoding it to a point T (ϕ_T) and encrypts ϕ_T using P_{AP} extracted from the certificate, such that $\phi_T = P_c||T_c$. The timestamp is included to prevent replay attacks. The message also includes the hash of the message to maintain integrity.

$$C \rightarrow AP : m_0 = \{kG, \phi_T + kP_{AP}\}, h(m_0) \tag{3}$$

3) **Probe Response:** The AP possess a public-private key pair as $\langle P_{AP}\text{-}n_{AP}\rangle$, where:

$$P_{AP} = n_{AP} G \tag{4}$$

On receiving the message, AP decrypts it using n_{AP}. Let the decrypted message be represented as m'_0. AP matches T_c with the current timestamp. If T_c is verified, then it computes the hash of m'_0, and matches against $h(m_0)$. If $h(m'_0) = h(m_0)$, then it selects a point on the curve ϕ_J. Using ϕ_J and P_c, it produces a master key (MK) as:

$$MK = \text{SHA-256}(\phi_J||P_c) \tag{5}$$

AP encrypts the MK using P_c and sends it to the client by encrypting it to the point S such that $\phi_S = P_{MK}||T_{MK}$, where T_{MK} represents the current timestamp.

$$AP \rightarrow C : m_1 = \{uG, \phi_S + uP_c\}, h(m_1) \tag{6}$$

This method of key exchange is known as Elliptic Curve Diffie-Hellman (ECDH) algorithm.
4) On receiving the message, the client decrypts it using n_c. Initially, it verifies T_{MK} and $h(m_1)$ to detect the legitimacy of the message. Further, it computes MKID (Master Key Identifier) as:

$$MKID = trucate_{64}\{h(P_c||P_{AP})\} \tag{7}$$

The client caches MK and MKID. Similarly, the AP also calculates MKID and caches MK and MKID. IEEE 802.11 implements“Pairwise Master Key (PMK) caching” for WPA where a client and AP can cache a PMK for a certain period and reuse it during the 4-way handshake occurring at the time of reassociation to bypass potentially expensive authentication. We have implemented the concept of caching to bypass the process of registration during reauthentication. For further connections, the client directly sends the authentication request encrypted with MK to the AP. Fig. 2 shows the steps involved in registration phase.

2.2.2 Authentication Phase

This phase authenticates a client to the network. Whenever the client gets disconnected from the network, the reassociation begins with this phase, known as reauthentication. The session keys are produced during this phase, which are utilized for encrypting further communication. All the message exchanges in this phase are encrypted. The phase contains the following steps:

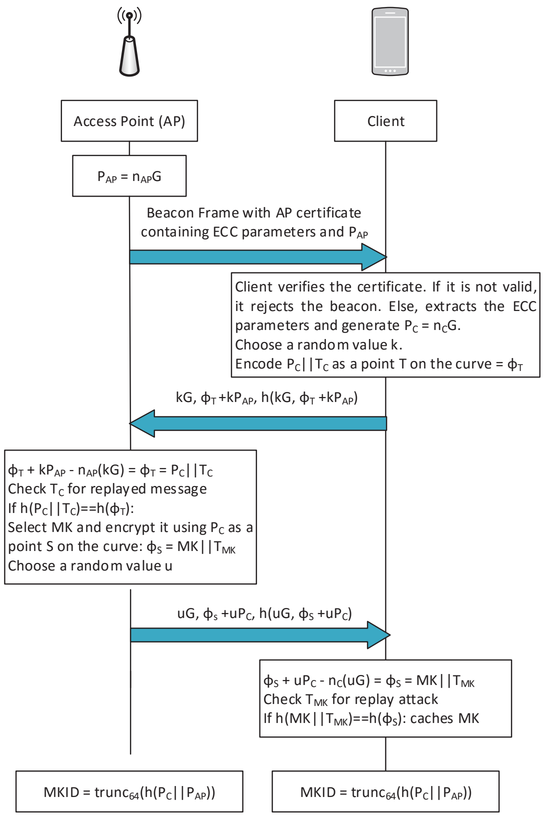
1) **Auth Request**: The client generates a nonce value N_A and sends it to AP with the current timestamp value T_A by encrypting them with MK.

$$C \rightarrow AP : m_2 = (N_A, T_A)_{MK} \tag{8}$$

2) **Auth Response**: On receiving Auth request, AP verifies T_A to check whether the message is genuine or any replayed message. Further, the AP generates a seed key K_{sk} and sends $\{N_A, K_{sk}, T_{sk}\}$ to the client encrypted with MK, where T_{sk} represents the timestamp value. The AP sends N_A again in the response to prove that the AP has successfully received the Auth request message and not any replayed message.

$$AP \rightarrow C : m_3 = \{N_A, K_{sk}, T_{sk}\}_{MK} \tag{9}$$

3) Next, the client and AP produce session key using PBKDF2 (Password-Based Key Derivation Function


Access Point (AP)
Client
$P_{AP} = n_{AP}G$
Beacon Frame with AP certificate containing ECC parameters and P_{AP}
kG, $\phi_T + kP_{AP}$, $h(kG, \phi_T + kP_{AP})$
uG, $\phi_S + uP_C$, $h(uG, \phi_S + uP_C)$
$MKID = trunc_{64}(h(P_C||P_{AP}))$
$MKID = trunc_{64}(h(P_C||P_{AP}))$

Client verifies the certificate. If it is not valid, it rejects the beacon. Else, extracts the ECC parameters and generate $P_C = n_C G$.
Choose a random value k.
Encode $P_C||T_C$ as a point T on the curve $= \phi_T$

$\phi_T + kP_{AP} - n_{AP}(kG) = \phi_T = P_C||T_C$
Check T_C for replayed message
If $h(P_C||T_C)==h(\phi_T)$:
Select MK and encrypt it using P_C as a point S on the curve: $\phi_S = MK||T_{MK}$
Choose a random value u

$\phi_S + uP_C - n_C(uG) = \phi_S = MK||T_{MK}$
Check T_{MK} for replay attack
If $h(MK||T_{MK})==h(\phi_S)$: caches MK

Fig. 2: Registration phase.

2) [17], which uses K_{sk} as the key and $N_A||MKID$ as the salt. It undergoes 4096 rounds of encryption to produce a key of 128 bytes in length. We use PBKDF2 because the cryptanalysis attack is highly expensive for this function [17].

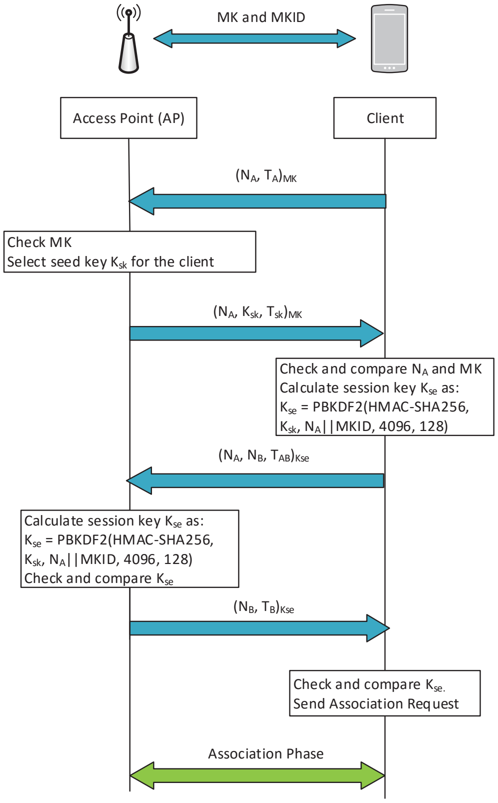
$$K_{se} = PBKDF2(HMAC \\ -SHA256, K_{sk}, N_A||MKID, 4096, 128) \tag{10}$$

4) **Auth Completion Request**: Client further generates a nonce value N_B, timestamp value T_{AB}, and sends $\{N_A, N_B, T_{AB}\}$ to the AP by encrypting it with new session key K_{se}.

$$C \rightarrow AP : m_4 = \{N_A, N_B, T_{AB}\}_{K_{se}} \tag{11}$$

5) **Auth Completion Response**: The AP verifies T_{AB} and N_A, and acknowledges the correctness of the received message by sending N_B and T_B encrypted with K_{se}. This message proves that both the parties have correctly generated the session key. The purpose of nonces and timestamps in the entire communication is to determine the continuity of messages and prevent replayed messages.

Fig. 3 shows the steps involved in authentication phase. Further, the client and AP proceed towards the association phase. Notably, all the subsequent transmitted messages are encrypted with the session key, which gets changed with every session as during reauthentication new session keys are produced by the client and AP.



Check MK
Select seed key K_{sk} for the client

Check and compare N_A and MK
Calculate session key K_{se} as:
$K_{se} = PBKDF2(HMAC\text{-}SHA256, K_{sk}, N_A||MKID, 4096, 128)$

Calculate session key K_{se} as:
$K_{se} = PBKDF2(HMAC\text{-}SHA256, K_{sk}, N_A||MKID, 4096, 128)$
Check and compare K_{se}

Check and compare K_{se}.
Send Association Request

Fig. 3: Authentication phase.

The registration phase of the proposed protocol is similar to HTTPS in a way that both AP and client verify the identity of other parties using certificates. But the difference is that for every session in HTTPS, the server send its certificate to the client for verification followed by the generation of session keys. Whereas in our protocol, the client verifies the certificate of AP only once, which reduces the computation time. Moreover, with the use of the proposed protocol, the encrypted communication will become a normal scenario which will enhance the security of the network.

Similar to WPA3-OWE, the proposed protocol also uses ECC during the initial handshake for generating master key. But the approach used is different. In WPA3-OWE, both the parties exchange public keys and further ECDH algorithm is utilized for deriving PMK. The unencrypted exchange of public keys and not validating them before proceeding towards PMK generation, make WPA3-OWE vulnerable to MITM and ET attacks. This proves that mere adoption of ECC does not guarantee the resistance of a protocol against network attacks.

The proposed protocol only exchanges the public key of the AP unencrypted, which is also validated by the client before proceeding further. After successful validation, the client sends it's encryption key to the AP by encrypting it with public key of AP, so that only AP can decode the message. This exchange catalyzes the generation and secure transmission of master key using ECDH algorithm. In SAP, every client possesses a unique MK used as a reauthentication parameter and for every new session, fresh session keys are generated. The uniqueness of master keys for each client and encryption of nonces make the proposed protocol resistant to dictionary attacks. The novelty of SAP lies in the secure exchange of encryption key of the client and reauthentication mechanism of the client.

3 ANALYSIS OF SAP

This section theoretically analyzes SAP in various aspects such as mutual authentication and security analysis.

3.1 Mutual Authentication between Client and AP

For authenticating client in the network, the AP verifies the MK received by the client in Auth request message. If the MK matches with the one generated and transmitted by the AP to the client in the registration phase (encrypted with the encryption key of the client), the AP authenticates the client in the network.

Suppose an attacker A sends an Auth-request message to AP encrypted with MK′:

$$A \rightarrow AP : (N'_A, T'_A)_{MK'} \tag{12}$$

On receiving the message, the AP tries to decrypt the message with the MK cached for the respective client. Since the message is encrypted with MK′ and not MK, the AP drops the message and does not send any Auth-response message further.

Similarly, the client also uses MK as a parameter for verifying the authenticity of the AP. The client sends Auth request message encrypted with the MK received from AP in the registration phase. If the AP can decrypt the message and send correct Auth response message, the client believes the legitimacy of the AP.

Suppose the attacker A captures the Auth request message sent by client to AP:

$$C \rightarrow AP : (N_A, T_A)_{MK} \tag{13}$$

Since the attacker does not possess MK, he is unable to decrypt the message. But he sends an Auth response message encrypted with MK′:

$$A \rightarrow C : \{N'_A, K'_{sk}, T'_{sk}\}_{MK'} \tag{14}$$

On receiving the message, the client tries to decrypt the message with the cached MK. When the client fails to decrypt, it drops the message and does not send any messages further.

Suppose, the MK shared between client and AP gets compromised. Consequently, the attacker successfully exchanges the Auth request and response messages with the client. But, when the attacker receives Auth completion request message from the client encrypted with the session key, he fails to decrypt the message because he is unable to produce correct session key. The reason being, the attacker does not have access to MKID generated in the registration phase. Thus, both client and AP can mutually authenticate each other in SAP and no third party can do this.

3.2 Security Analysis

In this subsection, we prove that the proposed protocol is able to prevent the exchanged messages from various types of network attacks, which we have formally proved in Section 4. We assume that capturing private key of the AP and decryption key of the client is not possible by the attacker as nowhere in the communication they are being exchanged. Fig. 4 shows the network model of SAP. In this model, three entities exist - client, AP and attacker. The figure shows the system setup of the network in the presence of the proposed protocol, the attacker's objectives and information possessed by the entities. In the figure, green-colored text represents publicly available information, black-colored text denotes the knowledge of the entities, and blue-colored text implies information being targeted by the attacker.

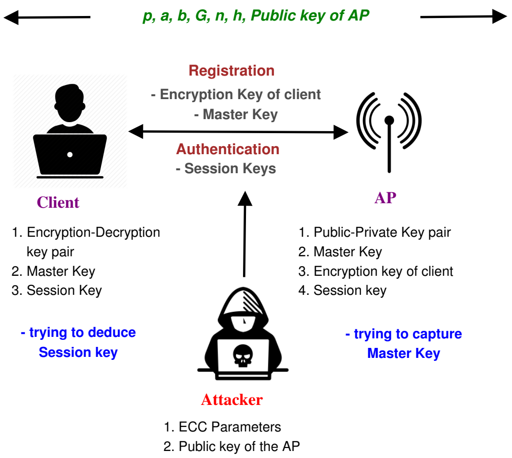

p, a, b, G, n, h, Public key of AP
Registration
- Encryption Key of client
- Master Key
Authentication
- Session Keys
Client
1. Encryption-Decryption key pair
2. Master Key
3. Session Key
- trying to deduce Session key
AP
1. Public-Private Key pair
2. Master Key
3. Encryption key of client
4. Session key
- trying to capture Master Key
Attacker
1. ECC Parameters
2. Public key of the AP

Fig. 4: Network model showing Network entities, their capabilities, attacker's objective and system setup used in SAP.

3.2.1 *Eavesdropping*

In SAP, all the exchanged messages are either encrypted using ECC or symmetric encryption to maintain end-to-end confidentiality of exchanged messages in the network. If the attacker eavesdrops the communication, he gets the encrypted frames which cannot be decrypted without the knowledge of private and decryption key (ECC), and MK and session keys (symmetric encryption). As already discussed, the attacker cannot obtain private and decryption keys because they are never transmitted. Thus, the attacker cannot generate MK and session keys as they are exchanged through messages encrypted with ECC keys. Hence, the transmitted messages in the network are secure from eavesdropping in the presence of the proposed protocol.

3.2.2 *Replay Attack*

SAP is resistant to replay attack as it sends timestamp values $(T_c, T_{MK}, T_A, T_{sk}, T_{AB}, T_B)$ with every message in the registration and authentication phase. Further, SAP also uses nonces in the authentication phase to prove the continuity of messages.

3.2.3 *ET Attack*

The objective of the attacker in ET attack is to force clients to get disconnected from the genuine AP and get connected to the ET so that the attacker can control the network traffic of the client. Suppose, in a network equipped with SAP; an ET disconnects a client from a genuine AP by sending deauthentication frames. The client tries to reauthenticate by sending Auth request frame. Although, the Auth request sent by the client is received by the ET, the ET is unable to read the contents of the message as it is encrypted with MK. Therefore, ET is unable to send correct Auth response message. An ET attack cannot be successful in the presence of SAP as an attacker needs private and public key of the AP and encryption key of the client to capture MK for successfully launching an ET attack.

3.2.4 *MITM Attack*

In this attack, the attacker tries to locate itself between client and AP, such that all the communication between them is through the attacker. Thus, the attacker can either intercept, replay or inject messages in the ongoing communication between the two parties. In our context, an attacker can perform MITM attack in two ways - (1) by launching ET attack and (2) by performing registration phase with the client. We have already proved that the ET attack cannot be launched in a network implementing SAP. Suppose, the client sends the probe request message to the AP. Attacker eavesdrops the communication and tries to forge the message to make the client connect to itself. However, the attacker cannot decrypt the probe request encrypted by the public key of AP as it does not have access to the private key of the AP. Nowhere in the exchanged messages, the private keys are shared. Moreover, we provide a public key certificate to the AP which makes the protocol resistant to MITM attack.

3.2.5 *Message Tampering Attack*

The proposed protocol provides one-to-one and end-to-end security to the transmitted messages between client and AP. To tamper the transmitted messages and breach the integrity of messages, an attacker needs to decrypt the message which is not possible without the keys. As proved earlier, the attacker cannot obtain any type of key and thus, the proposed protocol is safe from message tampering attack.

4 FORMAL VERIFICATION OF SAP

The formal verification of security protocols can be performed using model checking approach to automatically verify the security properties of a protocol. It is based on evaluating the protocol by exploring all possible states and behaviors of the protocol. It runs multiple instances of the protocol simultaneously and analyzes whether the protocol satisfies security properties in all the instances or not.

To verify and analyze the security properties of the proposed protocol, we use Scyther. The reasons being manifold - (1) Scyther utilizes the unbounded model checking approach with confirmed termination which allows it to verify all possible states and behaviors of the protocol [14], (2) Scyther uses backward symbolic state search technique which empowers it to explore all type flaws and infinite state spaces [14] and (3) According to a study conducted in [18], Scyther is the fastest tool among the existing state-of-the-art tools. In case of an attack, it gives an attack scenario which provides a better understanding of the flaws in the protocol.

In the subsequent subsections, we explain the adversary model and security claims used in Scyther, and further, we discuss the modeling and verification of SAP using Scyther.

4.1 Adversary Model

Scyther uses Dolev-Yao model as an adversary model which allows the adversary to replay, delete, breach, reroute, eavesdrop and process the content of the messages exchanged through the network. This model is predefined in the semantics of Scyther and thus, there is no need to define capabilities of an adversary for analyzing protocols in Scyther.

4.2 Security Claims

In Scyther, security properties are represented in the form of claims known as security claims. The adherence of a claim is checked by verifying whether the claim state is reachable or not during the protocol execution. Security claims in Scyther include:

- *Secrecy:* According to this claim, the messages exchanged over the network are not exposed to the attacker, even when the network is under full control of the attacker. The secrecy claim is expressed as $claim_L(R, secret, rt)$ which denotes, for the role R, rt should not be known to the adversary [14]. If rt is a session key, Scyther uses $claim_L(R, SKR, rt)$ to represent the secrecy of rt, where SKR stands for *Session Key Reveal*.
- *Mutual Authentication:* According to this claim, the communication must happen with the intended communication partner and not with the adversary. For verifying the authenticity of the communicating party, Scyther introduces the notion of *Synchronization*. This property states that the communication should occur between the expected, intended and genuine partners, and the protocol events should execute in the same way as described in the protocol specification. The *synchronization* claim is expressed as $Nisynch(R, Nisynch)$, where Nisynch stands for *Non-injective Synchronization*.
- *Agreement over exchanged messages*: Mutual authentication is not sufficient to judge whether the sent message is exactly same as the received message or not. It is also crucial to verify the integrity of the exchanged messages by checking the agreement of both the parties on the contents of exchanged messages. Scyther uses *commit* signal to verify the integrity of exchanged messages during protocol execution.

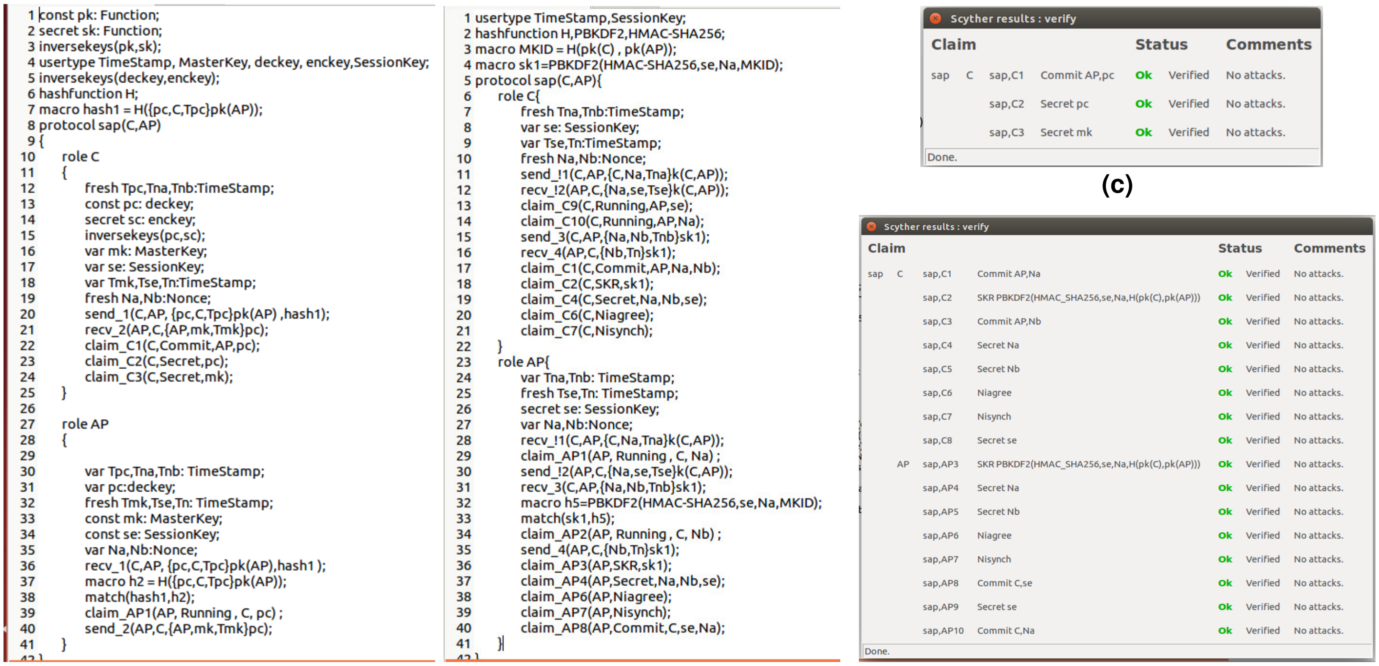
4.3 Verification of the Proposed Protocol

The proposed protocol involves two parties - $AP = Access\ point$ and $C = Client$. Table 2 represents the notations with their descriptions used in the verification of the proposed protocol using Scyther. The messages exchanged during the execution of the proposed protocol are represented according to the notations described in Table 2.

Registration

- $C \rightarrow AP : m_0 = \{pc,\ Tpc\}pk(AP),\ hash(m_0)$
- $AP \rightarrow C : m_1 = \{mk,\ Tmk\}pc,\ hash(m_1)$

Authentication



```
const pk: Function;
secret sk: Function;
inversekeys(pk,sk);
usertype TimeStamp, MasterKey, deckey, enckey,SessionKey;
inversekeys(deckey,enckey);
hashfunction H;
macro hash1 = H({pc,C,Tpc}pk(AP));
protocol sap(C,AP)
{
    role C
    {
        fresh Tpc,Tna,Tnb:TimeStamp;
        const pc: deckey;
        secret sc: enckey;
        inversekeys(pc,sc);
        var mk: MasterKey;
        var se: SessionKey;
        var Tmk,Tse,Tn:TimeStamp;
        fresh Na,Nb:Nonce;
        send_1(C,AP, {pc,C,Tpc}pk(AP) ,hash1);
        recv_2(AP,C,{AP,mk,Tmk}pc);
        claim_C1(C,Commit,AP,pc);
        claim_C2(C,Secret,pc);
        claim_C3(C,Secret,mk);
    }

    role AP
    {

        var Tpc,Tna,Tnb: TimeStamp;
        var pc:deckey;
        fresh Tmk,Tse,Tn: TimeStamp;
        const mk: MasterKey;
        const se: SessionKey;
        var Na,Nb:Nonce;
        recv_1(C,AP, {pc,C,Tpc}pk(AP),hash1 );
        macro h2 = H({pc,C,Tpc}pk(AP));
        match(hash1,h2);
        claim_AP1(AP, Running , C, pc) ;
        send_2(AP,C,{AP,mk,Tmk}pc);
    }
}
```

(a)

```
usertype TimeStamp,SessionKey;
hashfunction H,PBKDF2,HMAC-SHA256;
macro MKID = H(pk(C) , pk(AP));
macro sk1=PBKDF2(HMAC-SHA256,se,Na,MKID);
protocol sap(C,AP){
    role C{
        fresh Tna,Tnb:TimeStamp;
        var se: SessionKey;
        var Tse,Tn:TimeStamp;
        fresh Na,Nb:Nonce;
        send_!1(C,AP,{C,Na,Tna}k(C,AP));
        recv_!2(AP,C,{Na,se,Tse}k(C,AP));
        claim_C9(C,Running,AP,se);
        claim_C10(C,Running,AP,Na);
        send_3(C,AP,{Na,Nb,Tnb}sk1);
        recv_4(AP,C,{Nb,Tn}sk1);
        claim_C1(C,Commit,AP,Na,Nb);
        claim_C2(C,SKR,sk1);
        claim_C4(C,Secret,Na,Nb,se);
        claim_C6(C,Niagree);
        claim_C7(C,Nisynch);
    }
    role AP{
        var Tna,Tnb: TimeStamp;
        fresh Tse,Tn: TimeStamp;
        secret se: SessionKey;
        var Na,Nb:Nonce;
        recv_!1(C,AP,{C,Na,Tna}k(C,AP));
        claim_AP1(AP, Running , C, Na) ;
        send_!2(AP,C,{Na,se,Tse}k(C,AP));
        recv_3(C,AP,{Na,Nb,Tnb}sk1);
        macro h5=PBKDF2(HMAC-SHA256,se,Na,MKID);
        match(sk1,h5);
        claim_AP2(AP, Running , C, Nb) ;
        send_4(AP,C,{Nb,Tn}sk1);
        claim_AP3(AP,SKR,sk1);
        claim_AP4(AP,Secret,Na,Nb,se);
        claim_AP6(AP,Niagree);
        claim_AP7(AP,Nisynch);
        claim_AP8(AP,Commit,C,se,Na);
    }
}
```

(b)

```
Scyther results : verify
Claim                                  Status          Comments
sap   C   sap,C1   Commit AP,pc        Ok   Verified   No attacks.
          sap,C2   Secret pc           Ok   Verified   No attacks.
          sap,C3   Secret mk           Ok   Verified   No attacks.
Done.
```

(c)

```
Scyther results : verify
Claim                                                                     Status          Comments
sap   C    sap,C1    Commit AP,Na                                         Ok   Verified   No attacks.
           sap,C2    SKR PBKDF2(HMAC_SHA256,se,Na,H(pk(C),pk(AP)))        Ok   Verified   No attacks.
           sap,C3    Commit AP,Nb                                         Ok   Verified   No attacks.
           sap,C4    Secret Na                                            Ok   Verified   No attacks.
           sap,C5    Secret Nb                                            Ok   Verified   No attacks.
           sap,C6    Niagree                                              Ok   Verified   No attacks.
           sap,C7    Nisynch                                              Ok   Verified   No attacks.
           sap,C8    Secret se                                            Ok   Verified   No attacks.
      AP   sap,AP3   SKR PBKDF2(HMAC_SHA256,se,Na,H(pk(C),pk(AP)))        Ok   Verified   No attacks.
           sap,AP4   Secret Na                                            Ok   Verified   No attacks.
           sap,AP5   Secret Nb                                            Ok   Verified   No attacks.
           sap,AP6   Niagree                                              Ok   Verified   No attacks.
           sap,AP7   Nisynch                                              Ok   Verified   No attacks.
           sap,AP8   Commit C,se                                          Ok   Verified   No attacks.
           sap,AP9   Secret se                                            Ok   Verified   No attacks.
           sap,AP10  Commit C,Na                                          Ok   Verified   No attacks.
Done.
```

(d)

Fig. 5: (a) spdl script for registration phase, (b) spdl script for reauthentication phase, (c) Scyther execution results for registration phase, and (d) Scyther execution results for reauthentication phase.

TABLE 2
Notations with their descriptions used in the verification of the proposed protocol using Scyther

S.No.	Notation	Description
1.	sc	Decryption key of client
2.	pc	Encryption key of client
3.	$sk(AP)$	Private key of AP
4.	$pk(AP)$	Public key of AP
5.	Na, Nb	Nonces
6.	se	Seed key
7.	$sk1$	Session key
8.	mk	Master key
9.	T	Timestamp

- $C \rightarrow AP : m_2 = \{Na,\ Tna\}mk$
- $AP \rightarrow C : m_3 = \{Na,\ se,\ Tse\}mk$
- $C \rightarrow AP : m_4 = \{Na,\ Nb,\ Tnb\}sk1$
- $AP \rightarrow C : m_5 = \{Nb,\ Tn\}sk1$

We assume that revelation of $sk(AP)$ and sc to the attacker is not possible, as nowhere in the protocol specification, private and decryption keys are exchanged. Fig. 5(a) and 5(b) represent the *spdl* scripts of registration and authentication phase of the proposed protocol, respectively. In Fig. 5(b), the authentication phase represents the script used in reauthentication, where mk is replaced with $k(C, AP)$ as in reauthentication phase mk acts as a symmetric key shared between client and AP.

The following security *claims* are made in *spdl* scripts of registration and authentication phases of SAP:

- $claim(C, Secret, pc/mk)$: The encryption key of the client and master key generated by AP should not be revealed to the attacker for preventing MITM and ET attacks.
- $claim(C, Commit, AP, pc)$: Client C and AP should agree on the value of pc to proceed towards the authentication phase.
- $claim(AP/C, Secret, se/Na/Nb)$: The seed key and nonces used as an input to generate session key should not be leaked to the adversary to forbid attacker from generating the session key.
- $claim(AP/C, SKR, sk1)$: The generated session key $sk1$ should not be revealed to the attacker to avoid eavesdropping, MITM and message tampering attacks.
- $claim(C/AP, Commit, AP/C, Na/Nb/se)$: The AP and client should agree on the values of nonces Na and Nb, and seed key se, to ensure prevention from tampering attack.
- $claim(C/AP, Nisynch)$: For both the roles, the claim of synchronization should hold to ensure prevention from replay, ET and MITM attacks.

Fig. 5(c) and 5(d) represent the Scyther execution results of the proposed protocol, which shows that no attack has been found in the protocol. Hence, the proposed protocol is secure from the network attacks.

5 PRACTICAL PERSPECTIVE: SIMULATION OF PROPOSED PROTOCOL

The proposed protocol is simulated using the INET Framework extension of broadly accepted OMNeT++ simula-

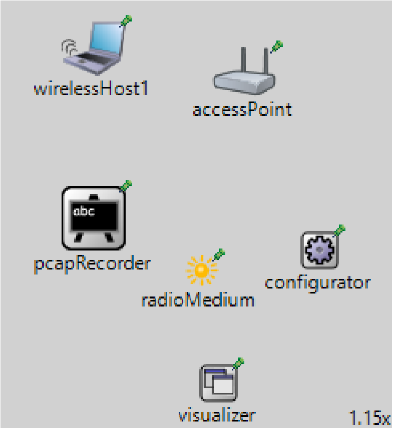


Fig. 6: Simulation setup for the proposed protocol.

tor [15] on Windows 10 platform. The INET framework of OMNeT++ represents network devices such as hosts, switches, routers, APs, etc., as modules written in C++. It also contains devices configured with IEEE 802.11 network interfaces and represents several layers of the IP suite such as UDP, TCP, ARP and IPv4 protocols. We embed OpenSSL APIs (Application Program Interface) in INET framework to model the cryptographic operations (ECC, AES-CCMP and PBKDF2) of the proposed protocol.

5.1 Simulation Setup

The setup for simulation consists of a network containing an AP modeled using *AccessPoint* compound module of INET, client modeled using *WirelessHost* compound module of INET, *configurator* module to assign IP address to the network entities, *radioMedium* module to send and receive packets for wireless nodes, *visualizer* module to visualize the packet transmission in the network, and *pcapRecorder* module to record the packets and further analyze them using packet analyzer tools such as Wireshark. Fig. 6 shows the network setup used for simulation.

We modified the management layer frames of *AccessPoint* and *WirelessHost* modules to include the ECC encryption and decryption operations in beacon, probe request and probe response frames; AES encryption and decryption functions in authentication frames; and PBKDF2 function for session key generation. The UDP protocol is used for exchanging messages between the modules.

5.2 Simulation Results and Discussion

The INET framework of OMNeT++ already contains a default implementation of the scanning and authentication process performed in IEEE 802.11 open networks. Since we modified the packet contents of the default implementation of IEEE 802.11 network protocol, we compared the performance of proposed protocol with the default open network protocol. The comparison is performed on various parameters such as computation overhead analysis (in ms) and packet size (in $bytes$).

5.2.1 Computation Overhead Analysis

Since the proposed protocol performs key generation and encryption-decryption functions in registration and authentication phases, it incurs an additional computation time. In the simulation of the proposed protocol, the approximate computation time spent on key generation and encryption-decryption operations while using various cryptographic algorithms is shown in Table 3. For ECC operations, we use the standard NIST curve *Secp384r1* owing to the reason that for a highly secure system, a minimum of 384-bit key size is required [19].

TABLE 3
Timing Analysis

Algorithm.	**Action**	**Approximate Time Taken (in μs)**
ECC	Encryption	200
	Decryption	300
	Key Generation	50
AES-CCMP	Encryption	400
	Decryption	500
PBKDF2	Key Generation	100

Let the time taken to generate a key, encrypt and decrypt a message using ECC be represented as α, β and γ, respectively; time taken to encrypt and decrypt a message using AES-CCMP be denoted as δ and ϵ, respectively; and time taken to generate a key using PBKDF2 be represented as η.

During the registration phase in SAP, the client generates keys using ECC and all the message transmissions are encrypted and decrypted through ECC. Let the computation time spent during the registration phase be denoted as T_{reg}. It is calculated as:

$$T_{reg} = \alpha + 2 \times \{\beta + \gamma\} \tag{15}$$

Using the values in Table 3, T_{reg} is evaluated as approximately 1050 μs. During the authentication phase, SAP encrypts and decrypts the messages using AES-CCMP and produces session key using PBKDF2 algorithm. Let the computation time consumed during the authentication phase be represented as T_{auth}. It is computed as:

$$T_{auth} = 2 \times \eta + 4 \times \{\delta + \epsilon\} \tag{16}$$

The approximate value of T_{auth} is obtained as $3800\mu s$. Thus, the total overhead incurred while connecting to SAP is $4850\mu s$ (T_{reg}+T_{auth}). So, overhead approximates to 4.85ms. However, the registration phase is a one-time process, so the overhead for reauthentication process of the proposed protocol is 3.8ms only. For many applications, low latency is of utmost importance, such as factory automation applications require latency between $0.25 - 10$ms, ITS applications between $10 - 100$ms, heathcare applications between $1 - 10$ms and education applications require less than 10ms [20]. Since the proposed protocol incurs a very low computational overhead, it can be a suitable choice for such applications.

5.2.2 Packet Size

Since the packets transmitted during registration and authentication phases in SAP implementation carries additional encrypted information for safely exchanging master and session keys, a comparison between the packet sizes of various frames under SAP and default open network implementation is required. Fig. 7 shows the packet size comparison. The difference in packet sizes of frames under SAP and default open network INET implementation during registration and authentication phases is approximately

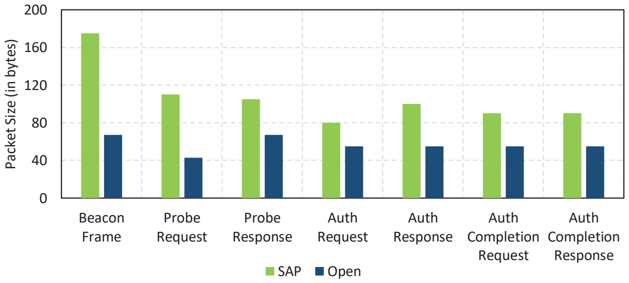


Fig. 7: Packet Size.

71B and 35B. We agree that the difference is not negligible, but if the trade-off between security offered by SAP in open networks and the packet sizes is considered, the difference can be ignored.

6 COMPARISON OF THE PROPOSED PROTOCOL WITH EXISTING STANDARD PROTOCOLS

We compared SAP with the most widely used protocols WEP, WPA, WPA2, WPA3, and 802.1x based on the following characteristics in Table 4 - **Attack methods** successfully used against the protocols, **number of parties** involved in the connection establishment and the **total number of messages** exchanged during the process. According to Table 4, the open authentication protocol is vulnerable to all the attacks as it does not incorporate any type of authentication and encryption. WPA is susceptible to ET, MITM and message tampering attacks due to the use weaker encryption standards such as RC4. WPA2 is vulnerable to ET and MITM attacks due to the unencrypted transfer of nonces making it a prey to offline dictionary attack. WPA3-OWE is considered as the most secure protocol for public networks at the time of writing this paper. It is found secure against all attacks except message tampering and ET attacks because of unencrypted transmission of authentication messages and the use TOFU model for authentication. WPA3-personal is the most secure standard for personal networks as of today, but due to its proneness to downgrade and dictionary attack, it is vulnerable to other network attacks as well. The computational overhead of WPA3-personal is very high given the complexity of the SAE handshake [10]. Moreover, it undergoes two rounds of four-way handshakes before authentication leading to a high connection establishment time and thus, a high latency protocol. The credential-based 802.1X protocol is vulnerable to sniffing attack due to unencrypted over-the-air transfer of credentials. To overcome this problem, 802.1X introduced certificate-based protocol which uses unique client certificates for authentication. RADIUS server authentication by clients is not implemented in many networks due to the increased overhead caused by the certificate management, which makes it vulnerable to ET attacks. In addition, 802.1X exchanges a higher number of messages for authentication compared to SAP. The costs of implementing and operating an 802.1x-based network are significant, as additional servers are required and the complexity of administration and certificate management is high.

From the Table 4, it can be seen that SAP exchanges the least number of messages compared to the presented protocols (except for open), which for directly results in reduced authentication and re-authentication time. In addition to this property, which is especially required in the area of low-latency applications, SAP offers very good protection against all evaluated threats. It assures end-to-end encrypted message transmission without any high-level expertise requirement for deployment. Although the protocol introduces the use of certificates for AP authentication which adds additional cost to AP setup, low latency, small computation overhead and security from network attacks makes it a reliable choice for sectors requiring both security and availability as their key parameters. The use of ECC makes SAP suitable for embedded and IIoT devices as ECC incurs low computational overhead owing to its small key size.

7 CONCLUSIONS

In this paper, we have discussed the severe consequences of ET attacks in a wireless network and studied the existing protocols in terms of their security against such attacks. In addition, a particularly important consideration for low-latency systems has been carried out with respect to the number of packets required for the authentication and reauthentication. We proposed a protocol that mitigates the considered attacks, incurs low computation overhead and a lower authentication time compared to the existing protocols. SAP ensures secure transmission of keys used for authentication, reauthentication and encrypted sessions. By using formal verification, we have justified resistance of SAP against network attacks and by performing simulation,

TABLE 4
Comparative analysis of SAP with existing standard protocols.

Protocol	Eavesdropping	Replay attack	ET attack	MITM attack	Message Tampering attack	No. of parties involved	Total messages exchanged
Open	✗	✗	✗	✗	✗	2	7
SAP	✓	✓	✓	✓	✓	2	9
WPA	✓	✓	✗	✗	✗	2	11
WPA2	✓	✓	✗	✗	✓	2	11
WPA3-OWE	✓	✓	✗	✓	✗	2	13
WPA3-personal	✓	✓	✗	✓	✓	2	15
Credential-based 802.1X	✗	✓	†	✓	✓	4	22
Certificate-based 802.1X	✓	✓	†	✓	✓	4	15

✗= The protocol is vulnerable to the attack, ✓= The protocol is resistant to the attack, †= The protocol may or may not be vulnerable to the attack we demonstrate that the proposed protocol achieves better latency and thus lesser delay in connection establishment process compared to other IEEE 802.11 standard protocols.

ACKNOWLEDGMENT

This work is supported by Federal Ministry of Education and Research of the Federal Republic of Germany (BMBF) within the PENTA project "SunRISE" (https://www.project-sunrise.eu/) under the Project Number 16ES0974.

REFERENCES

[1] "Cisco visual networking index: Forecast and trends, 2017 to 2022 white paper," https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html, [Online; accessed 05-March-2020].

[2] A. e. a. Frotzscher, "Requirements and current solutions of wireless communication in industrial automation," in *2014 IEEE Intl. Conf. on Commun. Workshops (ICC)*, 2014, pp. 67–72.

[3] V. e. a. Jain, "Etguard: Detecting d2d attacks using wireless evil twins," *Comput. Security*, 2019.

[4] "Russian wi-fi hacking - evil twin attacks," https://www.secplicity.org/2018/10/07/russian-wi-fi-hacking-evil-twin-attacks-explained/, [Online; accessed 30-October-2018].

[5] C. Kohlios and T. Hayajneh, "A comprehensive attack flow model and security analysis for wi-fi and wpa3," *Electronics*, vol. 7, no. 11, p. 284, 2018.

[6] J. S. Park and D. Dicoi, "Wlan security: current and future," *Internet Comput.*, vol. 7, no. 5, pp. 60–65, 2003.

[7] M. Waliullah and D. Gan, "Wireless lan security threats & vulnerabilities," *Intl. J. of Adv. Comput. Sci. Applicat.*, vol. 5, no. 1, 2014.

[8] O. e. a. Nakhila, "Parallel active dictionary attack on wpa2-psk wi-fi networks," in *MILCOM 2015-2015 IEEE Military Commun. Conf.* IEEE, 2015, pp. 665–670.

[9] S. Kwon and H.-K. Choi, "Evolution of wi-fi protected access: security challenges," *IEEE Consum. Electron. Mag.*, vol. 10, no. 1, pp. 74–81, 2020.

[10] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the dragonfly handshake of wpa3 and eap-pwd," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 517–533.

[11] M. Ghering and E. Poll, "Evil twin vulnerabilities in wi-fi networks," *Bachelor thesis, Radboud University, Nijmegen, Netherlands*, 2016.

[12] M. H. e. a. Hue, "All your credentials are belong to us: On insecure wpa2-enterprise configurations," in *Proc. of the 2021 ACM SIGSAC Conf. on Comput. and Commun. Security*, 2021, pp. 1100–1117.

[13] "Industrial wireless part 3: Choosing between wi-fi and cellular," https://www.automate.org/industry-insights/industrial-wireless-part-3-choosing-between-wi-fi-and-cellular, [Online; accessed 14-October-2021].

[14] C. J. F. Cremers, *Scyther: Semantics and verification of security protocols*, Ph.D. dissertation, Inst. Program. Res. Algorithmics, Eindhoven Univ. Technol., Eindhoven, The Netherlands, 2006.

[15] "Omnet++ discrete event simulator," https://omnetpp.org/, [Online; accessed 14-March-2020].

[16] N. Mehibel and M. Hamadouche, "A new approach of elliptic curve diffie-hellman key exchange," in *Proc. 5th Int. Conf. Elect. Eng.*, Boumerdes, Algeria, 2017, pp. 1–6.

[17] "PBKDF2," https://en.wikipedia.org/wiki/PBKDF2, [Online; accessed 15-March-2018].

[18] C. Cremers and P. Lafourcade, "Comparing state spaces in automatic security protocol verification," *Proc. 7th Int. Workshop on Automated Verification of Critical Systems (AVoCS'07)*, vol. 558, Electron. Notes Theor. Comput. Sci., Elsevier Science Publishers B. V. 2007.

[19] "Elliptic curve cryptography," https://www.linuxjournal.com/content/elliptic-curve-cryptography, [Online; accessed 15-April-2018].

[20] I. Parvez and R. et al., "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3098–3130, 2018.

Dr. Vineeta Jain is currently pursuing her post-doctoral study at Fraunhofer Institute of Integrated Circuits IIS Division Engineering of Adaptive Systems EAS, Germany. Her research interests include the area of security and privacy with a special emphasis on wireless networks and mobile security. She received her Ph.D. in computer science from Malaviya National Institute of Technology Jaipur, India. She is student member of IEEE. Contact her at vineeta.jain@eas.iis.fraunhofer.de.

Ulf Wetzker is a Research Scientist with the Division Engineering of Adaptive Systems (EAS), Fraunhofer Institute for Integrated Circuits IIS, Germany. His research interests are in the areas of wireless communication systems, data analytics, and machine learning, with a special focus on anomaly detection in wireless networks. He received the Diploma degree in computer science and systems engineering from the Ilmenau University of Technology, Ilmenau, Germany. Contact him at ulf.wetzker@eas.iis.fraunhofer.de

Prof. Vijay Laxmi is a faculty in Department of Computer Science and Engg., Malaviya National Institute of Technology Jaipur, India. Her research interests include Information security, Malware analysis, Security and QoS provisioning in wireless Networks. She received her PhD from University of Southampton, UK. Contact her at vlaxmi@mnit.ac.in.

Prof. Manoj Singh Gaur has been a faculty in Department of Computer Engg., Malaviya National Institute of Technology Jaipur, India and currently Director, IIT Jammu, India. His research areas include Networks-on-Chip, Computer and network security and wireless networks. He received PhD from University of Southampton, UK. He is a member of IEEE. Contact him at gaurms@mnit.ac.in.

Prof. Mohamed Mosbah is a full Professor in computer science at the Polytechnic Institute of Bordeaux, France. His research interests include distributed algorithms and systems, formal models, security, and ad hoc and sensor networks. He obtained his Ph.D. from the University of Bordeaux, in 1993. Contact him at mohamed.mosbah@labri.fr

Prof. Dominique Mery is a full professor of computing science at University of Lorraine since September 1993. His research interests include proof-oriented system development for computer-based systems with higher levels of reliability and correctness, proof- based development of distributed algorithms and the proof-based modelling of medical devices. He is a member of IEEE. Contact him at dominique.mery@loria.fr.