International Journal for Information Security Research (IJISR), Volume 3, Issues 3 and 4, September/December 2013

Analysing the EAP-TLS Handshake and the 4-Way Handshake of the 802.11i Standard

Abdullah Alabdulatif[1], Xiaoqi Ma[2]
Department of Computer, College of Sciences and Arts, Qassim University, Al-Rass, Saudi Arabia[1]
School of Science and Technology, Nottingham Trent University, Nottingham, UK[2]

Abstract

The IEEE 802.11i standard has been designed to enhance security in wireless networks. The EAP-TLS handshake aims to provide mutual authentication between supplicant and authentication server, and then derive the Pairwise Master Key (PMK). In the 4-way handshake the supplicant and the authenticator use PMK to derive a fresh pairwise transient key (PTK). The PMK is not used directly for security while assuming the supplicant and authenticator have the same PMK before running 4-way handshake. In this paper, the EAP-TLS handshake and the 4-way handshake phases have been analysed with a proposed framework using Isabelle tool. In the analysis, we have found a new Denial-of-Service (DoS) attack in the 4-way handshake. The attack prevents the authenticator from receiving message 4 after the supplicant sends it out. This attack forces the authenticator to re-send the message 3 until time out and subsequently to de-authenticate supplicant. This paper has proposed improvements to the 4-way handshake to avoid the Denial-of-Service attack.

1. Introduction

One of the great challenges for wireless environments is to provide enough strong protection to the data packages exchanged over WLANs. Eavesdropping attacks can be conducted in WLANs by potential attackers with suitable radio receivers and little effort. So attackers can attack a WLAN with difficult detection or prevention [1]. The wired equivalent privacy protocol (WEP) has been the first attempt proposed to protect the data packages exchanged over WLANs. However, WEP does not provide strong protection to the data packages exchanged over WLANs, especially in encryption. In June 2004, the IEEE task group *i* developed a new standard called 802.11i to avoid the weaknesses in WEP and to enhance confidentiality, integrity and mutual authentication [2].

The 802.11i standard involves three entities called supplicant (wireless device), authenticator (access point) and authentication server. All six phases of the 802.11i standard are important to achieve authentication, especially for the EAP-TLS handshake and the 4-way handshake. The EAP-TLS Handshake includes a series of message exchange between the entities in specific order. The order of messages is significant in EAP-TLS handshake, whereas a number of options are available. The access point participates the EAP-LTS handshake as a reply without checking the content of messages [3]. The 4-way handshake aims to establish a fresh session key between the access point and the wireless device. There are three tasks for the access point and the wireless device to achieve successfully in the 4-way handshake phase. Firstly, establish random nonces to verify the liveliness of each other. Then, confirm the existence of the PMK at the access point and the wireless device. Finally, generate the group transient key (GTK) by the access point and transfer the GTK to the wireless device [4].

The phases of IEEE 802.11i Standard can be analysed using linear temporal logic. Alabdulatif *et al*. have proposed a framework which can be used to investigate and analyse the EAP-LTS handshake and the 4-way handshake [5,6]. This framework can be classified as a theorem proving method, which is used to analyse all possible behaviours of a protocol to ensure they meet a set of correctness conditions [7]. There are a number of general rules and assumptions in the framework that can be used to analyse many protocols. Isabelle is one of the tools that can be used to implement the framework and to analyse protocols. In this paper, we use the proposed framework to successfully identify a DoS attack in the 4-way handshake.

The paper is structured as follows. Section 2 will introduce the IEEE 802.11i standard. Section 3 will provide the framework adjusted for analysing the 802.11i standard using Isabelle. Section 4 will show the analysis of EAP-TLS handshake. Section 5 will present the analysis of 4-way handshake. Section 6 will show how to defend against the denial of service attack on the 4-way handshake. Section 7 will present conclusions and future work.

2. IEEE 802.11i standard

The IEEE 802.11i standard provides confidentiality, integrity and mutual authentication of the WLANs security. There are two mechanisms used to achieve confidentiality and integrity of data, namely the Temporal Key Integrity Protocol (TKIP) and the Counter Cipher Mode with Block Chaining Message Authentication Code Protocol (CCMP).

The TKIP is a temporary solution for the WEP flaws, whereas the CCMP is a comprehensive solution requiring specific hardware features. For the mutual authentication the 802.11i standard defines a Robust Security Network (RSN) with two new protocols, the first of which is the 4-Way Handshake and the other is Group Key Handshake [4,8]. The aim of the paper focuses on analysing the authentication aspect, especially in the EAP-TLS Handshake and the 4-Way Handshake phase.

2.1. Overview of the 802.11i standard phases

The 802.11i standard has six sequential phases to achieve authentication among the authentication server, the access point and the wireless device. In each phase there are some tasks that should be achieved successfully to meet the security target of the phase. The success of authentication means the wireless device and the access point are identified and verified by each other and a secret key is established for exchanging encrypted data over WLANs. Figure 1 shows that the authentication procedures consists of six phases as follows: a) discover phase, b) authentication and association phase, c) EAP/802.1x/RADIUS authentication, d) 4-way handshake, e) group key handshake, f) secure data communication [9].

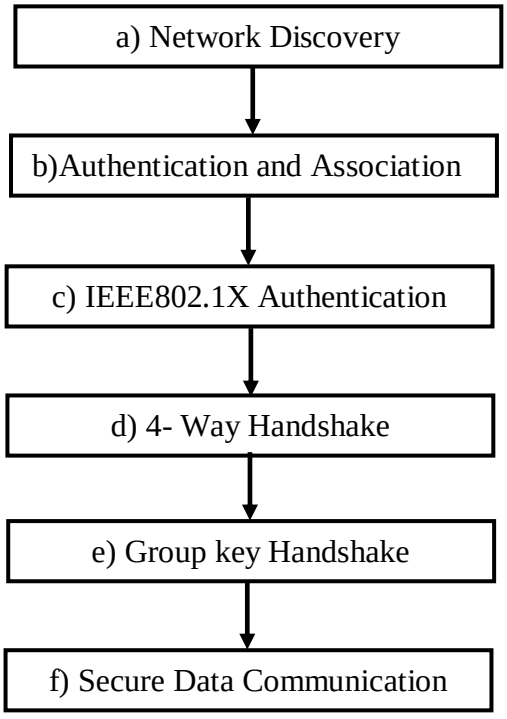

Figure 1. IEEE 802.11i authentication phases

The first phase in the IEEE 802.11i standard is the network discovery phase. The aim of this phase is that a wireless device selects one of the access points available and corresponding security capabilities. There are two ways to choose appropriate access points. One way is that the wireless device observes the Beacon frames and identifies the access point by these frames. The other way is that the wireless device can send a probe request frame to obtain a response frame from all access points available [10,11].

The second phase is authentication and association. It aims to perform authentication and association with a specific access point for communication. The wireless device selects an access point from the list created in the previous phase. Then the association between the wireless device and the access point is built up through negotiating the security capabilities. The open system authentication is used to indicate backward compatibility. In this phase, the authentication between the wireless device and access point is insecure and the secure mutual authentication will be in the next phase [10,11].

The IEEE802.1X authentication phase is the third phase. This phase aims to provide mutual authentication. The wireless device and the authentication server have to authenticate each other. The wireless device and the authentication server establish a Master Session Key (MSK). The wireless device uses the MSK to derive the PMK where the authentication server transfers the key material to access point to derive the same PMK. The wireless device and the access point may ignore the third phase if both of them have pre-Shared Key (PSK) or cashed PMK used in re-association[10,11] .

The 4-way handshake is the fourth phase. It aims to verify that the access point is legitimate and establish a fresh session key PTK between the access point and the wireless device [2]. The wireless device may request to run the 4-way handshake protocol or the access point may start by itself and both have the same PMK before running the 4-way handshake protocol [12].

The fifth phase is group key handshake phase. It aims to distribute a fresh Group Temporal Key (GTK) to wireless devices. The access point is able to generate the GTK and multicast to the wireless devices in this phase. When the GTK has been distributed in the previous phase the group key handshake phase will be unnecessary. The GTK distribution may be repeated multiple times from the same access point [10,11].

The last phase is secure data exchange phase and it aims to establish a secure channel between the access point and the wireless device.

After all necessary phases are achieved successfully, the wireless device can connect with access point using fresh PTK or GTK to protect data packets.

2.1.1. The EAP-TLS handshake. The EAP-TLS represents the integration of the EAP framework and

TLS protocol [13]. There are ten steps to achieve the EAP-TLS handshake. The EAP messages start and end with same sequence. As shown in figure 2, we assume EAP messages are secure and will focus on EAP-TLS handshake which includes the messages from 4 to 9. There are a number of optional messages and we will ignore them in our analysis. The full handshake is sometimes unnecessary, where the wireless device can resume a session with a fresh nonce and an existing session identifier. Then the authentication server will reply a fresh nonce to resume the specific session. The wireless device and authentication server have stored a master key from which they can derive the session keys. They finish messages exchange between them by confirming these keys [14].

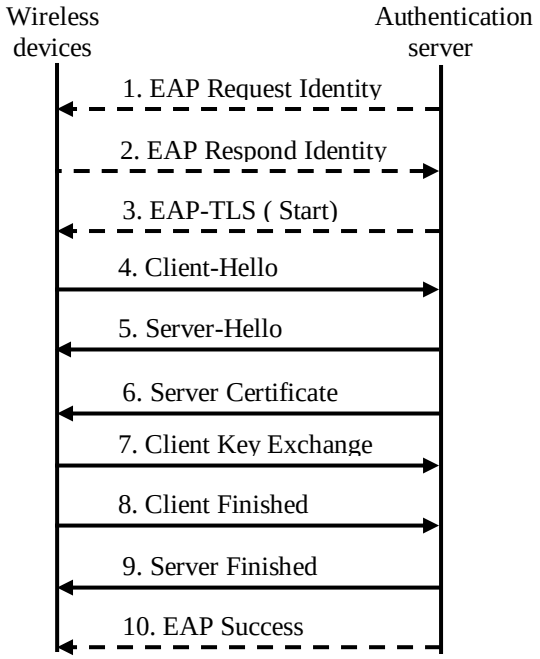

Figure 2: EAP-TLS handshake.

The first message in TLS handshake is **Client Hello**. It contains the client name A, client random number Na and a session identifier Sid [14]. It represents as follows:

Client Hello: A, Na, Sid

The second message in TLS handshake is the **Server Hello** containing server random number Ns and the session identifier Sid [14]. It represents as follows:

Server Hello : Ns; Sid

The third message in TLS handshake is the **Server Certificate** including a certificate signed by a trusted third party containing server's public key [14]. It represents as follows:

Server Certificate: Certificate(B, Kb)

The fourth message in TLS handshake is the **Client Key Exchange** including the random number generated by client called Pre-master secret (PMS) encrypted by server's public key [14]. It represents as follows:

Client Key Exchange : {PMS}

The last two messages are **Client Finished** and **Server Finished**. The Client Finished message is the hash of all previous messages encrypted by the symmetric key *clientK*. The Server Finished message is the hash of all previous messages encrypted by the symmetric key *serverK [14]*. They represent as follows:

Client Finished : {Finished}$_{ClientK}$

Server Finished : {Finished}$_{ServerK}$

2.1.2. The 4-way handshake phase. The 4-way handshake is essential in the IEEE 802.11i protocol, aiming to verify that the access point is legitimate to generate the PMK. Figure 3 shows that the 4-way handshake exchanges messages at abstract level, where AA and SPN represent the MAC address of the access point and wireless device, respectively. SNonce represents the access point nonce. and ANonce represents the nonce of the wireless device. The msg1, 2, 3, 4 refer to several message types; *sn* is sequence number. MIC_{PTK} {} refers to the Message Integrity Code (MIC) that uses the fresh PTK to calculate the integrity code of contents between the braces. MIC is used instead of Message Authentication Code (MAC) for cryptography because the meaning of MAC in network is medium access control [15].

PTK = PRF-X(PMK, Pairwise key expansion || Min {AA, SPA} || Max {AA, SPA} || Min {ANonce, SNone}|| Max {ANonce, SNone})

The fresh PTK is divided into three keys. The first key is the Key Confirmation Key (KCK), which is only used to calculate MIC. The second key is the Key Encryption Key (KEK) and the third key is the Temporary Key (TK). The KEK and TK are not used in the authentication process, so they will be ignored in this paper [3].

The wireless device and the access point can discard a message in the 4-way handshake when receiving a message with unexpected sequence number or invalid MIC. Message 1 is unacceptable for the wireless device when it is received after the

time interval of successful 802.11i authentication. In this case, the wireless device tries to authenticate with same access point or another one after disassociating and de-authenticating the current access point. On the other side, if the access point has not received a message before time out then it will re-send within configured time intervals. Moreover, the access point will de-authenticate the wireless device if it has never received any reply from the wireless device [3].

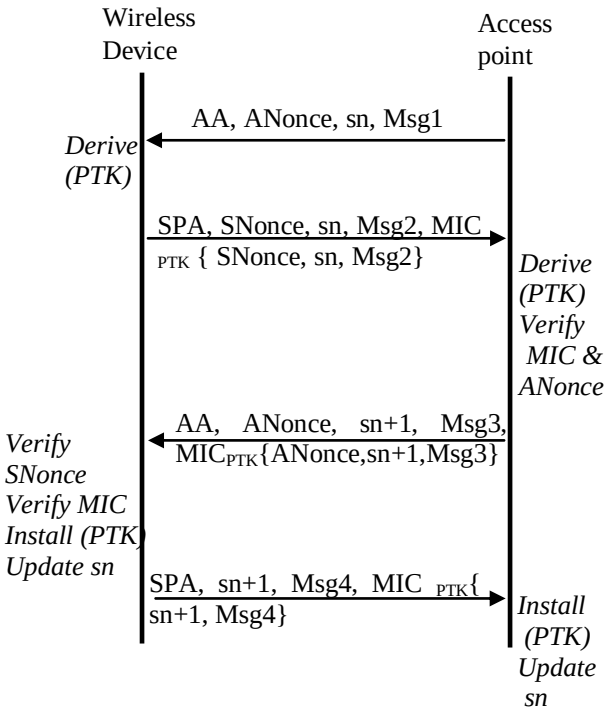

Figure 3: 4-way handshake.

3. A framework implemented in Isabelle

Isabelle tool is widely used to reason a formal system based on higher order logic. Bella defined Isabelle tool as "a generic, interactive theorem prover" [16]. Isabelle provides a high level automation, which means human intervention required is lower than many other tools. Paulson is one of the researchers who have used Isabelle to prove the correctness of a number of protocols, such as the internet protocol TLS [14]. In this paper, Isabelle is used to analyse the 4-way handshake phase of 802.11i standard.

3.1. A Framework for analysing protocols using linear temporal logic

Isabelle can be used to verify and prove the correctness of security protocols. Four steps are followed to analyse protocols using Isabelle tool. First, adjust the framework slightly for the protocol to be verified. The reason is that the framework is a template and requires accommodating the minor differences amongst various security protocols. Then, model the protocol steps by rewriting the protocol to make it compatible with the language used in the framework. After that, prove basic and essential properties of the protocol, which can be reused for other protocols. Finally, prove security properties of the protocol based on the proof of the basic properties mentioned above. In the next section, we will follow these steps and use the framework proposed by Alabdulatif *et al* [5, 6].

3.2. Framework adjustment

The framework requires a slight amendment to be appropriate for analysing the EAP-TLS handshake and 4-way handshake protocol. The access point AP and the wireless device SP are honest agents and the attacker here is called Spy. Also, the trusted third party is denoted as TTP. The definition of agent will be modified as:

datatype *agent = SP | AP | Spy | TTP*

In the EAP-TLS handshake, the wireless device and the authentication server exchange some random values. The random type consists of PMS and MASTERKEY representing the pre-master secret and the master key, respectively. Also, Na represents the random value generated by wireless device for the current session and the Ns represents the random value generated by authentication server for the current session. Sid is the random value representing the session identifier generated by the wireless device. The random type can be defined as follows:

datatype *random = PMS | MASTERKEY | Na | Ns | Sid*

The wireless device uses Na, Ns and MASTERKEY random values in the EAP-TLS handshake to generate the symmetric key SP_K to encrypt Client Finished message. Whereas, the authentication server uses the same values to generate the symmetric key AS_K to encrypt Server Finished message. The SP_K and AS_K can be defined as follows:

SPK :: *random * random * random => nat*
ASK :: *random * random * random => nat*

Similarly, in the 4-way handshake four new nonces are used, with *SN* and *SN1* representing the sequence number and the sequence number +1, respectively. The SNonce and ANonce are fresh nonces chosen by agents SP and AP, respectively.

Therefore, the definition of nonce will be modified as:

datatype *nonce = SN | SN1 | SNonce | ANonce*

In addition, a new type will be defined for Msg1, Msg2, Msg3 and Msg4. This type will be called Messages and added as follows:

datatype *Masseges = Msg1 | Msg2 | Msg3 | Msg4*

Since the 4-way handshake uses the Message Integrity Code and typed messages, we need to add two constructors for datatype msg. They can be defined in msg datatype:

datatype *msg =Mag Masseges*
| MIC msg key

Besides the type definitions, the analysis requires several new actions to represent their behaviours during the authentication process. The three new actions are added as follows:

Discard :: *agent => msg => Formula*
Block :: *agent => msg => Formula*
RRcv :: *agent => msg => Formula*

The Discard action represents the behaviour of an agent when ignoring received message. The Block action represents the behaviour of an agent when removing the message from the network so that the recipient cannot receive the message. The behaviour of an agent receiving the same message more than once can be represented by the RRcv action.

Since new definitions and actions have been added into the framework, it is necessary to introduce a set of new rules to describe new properties:

Rule 1.1 : S |= RRcv A M ∧ (S < t)) => t |= □ (Discard A M).

This rule says that if an agent receives the same message more than once, then the agent will always discard this message.

Rule 1.2 : (S |= Rcv A M) ∧ (t |= Rcv A M)) => (t |= RRcv A M) ∧ (S < t).

This rule says that if an agent receives a message at moment *S* and receives the same message at moment *t*, then the agent receives the message more than once.

Rule 1.3 : (S |= Rcv Spy M) ∧ (S |= Block Spy M)) => ∀X:(S |= (Neg (Rcv X M))).

This rule says that if the attacker receives a message and blocks it, then other agents in the network cannot receive this message.

After adjusting the framework is suitable for analysing the EAP-TLS handshake and 4-way handshake stages. We will analyse these stages separately.

4. Analysing EAP-TLS handshake

The next steps to analyse the EAP-TLS handshake protocol are modelling the protocol and proving the basic and security properties.

4.1. Modelling the EAP-TLS handshake

Normally, a protocol is written in informal language as shown in figure 2. The EAP-TLS handshake protocol steps for the honest agents should be formalised. The six steps of EAP-TLS handshake can be formalised in the framework as follows:

ClientHello: S |= Send SP AS ({Agent SP,{Random Na, Random Sid}}) ∧ (S < t) => t |= Rcv AS ({Agent SP, {Random Na, Random Sid}})

ServerHello: S |= Rcv AS ({Agent SP, {Random Na,Random Sid}}) => t|=Send AS SP ({Random Ns, Random Sid}) ∧ (S < t).

SendServerCertificate: S |= Send AS SP ({Random Ns, Random Sid}) ∧ (S < t) => t |= Send AS SP (Cert({Agent AS, Key (Kpb (AS))}) (Kpb (TTP)))

SendClientKeyExchange: S |= Know SP (Encrypt (Random (PMS)) ((Kpb (AS)))) => t |= Send SP AS (Encrypt (Random (PMS)) ((Kpb (AS)))) ∧ (S < t)

SendClientFinished: S |= Send SP AS (Encrypt (Hash {Finished})((Ksym SP AS(SP_K (Na, Ns, MASTERKEY))))) => t |= Rcv AS (Encrypt (Hash {Finished})((Ksym SP AS (SP_K (Na, Ns, MASTERKEY))))) ∧ t |= Verify AS (Encrypt (Hash {Finished})((Ksym SP AS(SP_K (Na, Ns, MASTERKEY))))) ∧ (S < t)

SendServerFinished: S |= Rcv AS (Encrypt (Hash {Finished})((Ksym SP AS(SP_K (Na, Ns, MASTERKEY))))) ∧ S |= Verify AS (Encrypt (Hash {Finished})((Ksym SP AS(SP_K (Na, Ns, MASTERKEY))))) ∧ (S < t) => t |= Send AS SP (Encrypt (Hash {Finished})((Ksym SP AS (AS_K (Na, Ns, MASTERKEY)))))

4.2. Verifying basic properties

In proving the EAP-TLS handshake, basic properties proved for Needham-Schroeder public key protocol can be reused. Thus, there are not any basic properties that should be specifically proved for EAP-TLS handshake.

4.3. Verifying security properties

The first security property is *knowMasterkey* and it says that if the wireless device receives the authentication server's public key in the certificate signed by a trusted third party, then the authentication server will know the master key. This property can be proved as follows:

lemma *knowMasterkey* : S |= Rcv SP (Cert ({Agent AS, Key (Kpb (AS))}) (Kpb (TTP))) ∧ S |= Generate SP (Random (PMS)) ∧ (S < t) => t |= Know AS (Random (MASTERKEY))

apply(rule KnownMasterKey)
apply(rule KnownEncryptMessage)
apply(rule SendClientKeyExchange)
apply(rule Rule 1)
apply(rule KnowcertMessage)
apply(auto)
done

The second security property is *LTShandshake* which says that when the wireless device sends client hello message at moment *S*, then the authentication server will send server finished message at moment *t*. Similarly, this property can be proved as follows:

lemma *LTShandshake*: S |= Send SP AS ({Agent SP, {Random Na, Random Sid}}) ∧ (S < t) => t |= Send AS SP (Encrypt (Hash {Finished})((Ksym SP AS (AS_K (Na, Ns, MASTERKEY)))))

apply(rule SendServerFinished)
apply(rule SendClientFinished)
apply(rule KnowserverMaster)
apply(rule knowMasterkey)
apply(rule RcvServerCertificate)
apply(rule SendServerCertificate)
apply(rule ServerHello)
apply(rule ClientHello)
apply(auto)
done

In the verification of the EAP-TLS handshake phase, the pre-master secret is difficult to know by any agent except the authentication server and the wireless device. The difficulty is because the pre-master secret is encrypted by the public key of the authentication server when sent over the network. This has been shown in the proof of the lemma *KnownMasterKey*. The lemma *LTShandshake* proves the correctness of EAP-TLS handshake and that it meets the targets of security. Therefore, we can conclude that the EAP-TLS handshake phase of 802.11i standard is secure.

5. Analysing 4-way handshake

The next steps to analyse the 4-way handshake protocol are modelling the protocol and prove the basic and security properties.

5.1. Modelling the 4-way handshake

Normally, a protocol is written in informal language as shown in figure 3. In this part we will therefore formalise the steps of the 4-way handshake as four formal formulas for all honest agents as follows:

FHShake1: S |= Send AP SP ({Agent AP,{Nonce ANonce, {Mag Msg1, Nonce (*SN*)}}}).

FHShake2: (S |= Send AP SP({Agent AP, {Nonce ANonce, {Mag Msg1{ Nonce (*SN*)}}})) => t |= Send SP AP ({Agent SP, { Nonce SNonce, {Mag Msg2, {Nonce (*SN*), MIC {Nonce SNonce, {Mag Msg2, Nonce (*SN*)}} k}}}}) ∧ (S< t).

FHShake3: S |= Send SP AP ({Agent SP, { Nonce SNonce, {Mag Msg2, {Nonce (*SN*), MIC {Nonce SNonce, {Mag Msg2, Nonce (*SN*)}} k}}}}) ∧ (S < t) => t |= Send AP SP({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce, {Mag Msg3, Nonce(*SN1*)}} k}}}}).

FHShake4: S |= Send AP SP({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce, {Mag Msg3, Nonce (*SN1*)}} k}}}}) => t |= Send SP AP ({Agent SP, {Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}}}) ∧ (S< t).

The access point will re-send message 1 and message 3 if it did not receive the reply during the per-defined time interval. The access point will continue to re-send and, after timeout, will de-authenticate the wireless device if there is no reply from it. There are two rules for re-sending the message 1 and message 3, as described below:

ReplayMessage1: S |= ¬(Rcv AP ({Agent SP,{ Nonce SNonce, {Mag Msg2, {Nonce (*SN*), MIC {Nonce SNonce, {Mag Msg2, Nonce (*SN*)}} k}}}})) ∧ (S< t) => S |= Send AP SP ({Agent AP, {Nonce ANonce, {Mag Msg1, Nonce (*SN*)}}})

ReplayMessage3: S |= ¬(Rcv SP ({Agent SP,

{Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}})) ∧ (S < t) => t |= Send AP SP ({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce, {Mag Msg3, Nonce (*SN1*)}} k}}}}) ∧ (t < outtime) ∧ (intervaltime < t)

The attacker has the ability to block any messages over the network. So if any agent sends a message, the attacker can block it and the recipient will not be able to receive it. The rule for blocking message 4 can be represented in the framework as follows:

BlockMessage4: S |= Send SP AP ({Agent SP, {Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}}}) => S |= Block Spy ({Agent SP, {Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}}}).

5.2. Verifying basic properties

The first basic property is discarding the received messages. The reason for an agent discarding a received message is because the same message is received more than once. So for security reasons the agent should discard the duplicate copies of a message. The first property says that when agent A sends a message to agent B more than once, then agent B will always discard the message:

lemma *DiscardReceivedMessage* : (S |= Send A B M) ∧ (S < t) => (t |= □ (Discard B M))

apply (rule Rule 1.1)
apply (rule Rule 1.2)
apply (rule conjI)
apply (rule Rule 8)
apply (auto)
apply (rule Rule 8)
apply (auto)
done.

The second basic property is a special case of the blocked message 4 by the attacker. This property says that when the wireless device sends message 4, then the message may not be received by the access point if it is blocked by the attacker.

lemma *NotReceivedMessage4FromSender* :
S |= Send SP AP ({Agent SP, {Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}}})
∧ (S < t) => S |= ¬ (Rcv AP ({Agent SP, {Mag Msg4, {Nonce (*SN1*), MIC {Mag Msg4, Nonce (*SN1*)} k}}}))

apply(rule allE)
apply(rule Rule 1.3)
apply(auto)
apply(rule Eavesdropping rule)
apply (auto)
apply (rule BlockMessage4)
apply (auto)
done

6. Denial of service attack on the protocol

In a DoS attack, the adversary prevents or inhibits protocols from completing successfully. Simply speaking, it involves disabling or preventing servers who are required to interact with participants. Most protocols have the potential to be attacked by DoS; however, the design of a protocol could improve prevention, or make such attacks more unlikely [17]. It is impossible to fully protect protocols against DoS attacks.

6.1. DoS Attack on the 4-way handshake

The sequence number (*sn*) is a technique used to prevent reply attacks in the 4-way handshake. *sn* is a counter set to 0 when establishing PMK then incremented with successive messages. The wireless device and the access point assume that they have the same *sn* value before running the 4-way handshake. During the running of the 4-way handshake the wireless device should update the *sn* value when receiving the message 3, while the access point should update the *sn* value after receiving the message 4 as shown in figure 1. As a result, at the end of the 4-way handshake we assume that they will have the same *sn* value. If the wireless device and access point have different *sn* values at the end of the 4-way handshake they will de-authenticate each other and cannot start future sessions.

The *sn* value can be a potential vulnerability in the 4-way handshake. The wireless device and the access point will continue running the 4-way handshake until time out without knowing the attacker having blocked message 4. The access point will re-send message 3 if it does not receive message 4 while the wireless device discards these messages as shown in figure 4. This attack happens because each side has different values of *sn*. A simple effort of the attacker, which blocks message 4 once then lets the protocol run as usual, can destroy the authentication between the wireless device and the access point. It is easy for the attacker to detect message 4 over the network because the 4-way handshake exchanges messages without encryption.

One of the security properties in the 4-way handshake is synchronising the installation of session keys. The wireless device installs the session key after receiving message 3 and the access point will install it after receiving message 4. In Isabelle tool, a lemma shows that the wireless device will discard message 3 resent by the access point. As a result, if the wireless device discards message 3, message 4

will not be received by the access point and the access point cannot install the session keys. The proving scripts of this lemma are as follows:

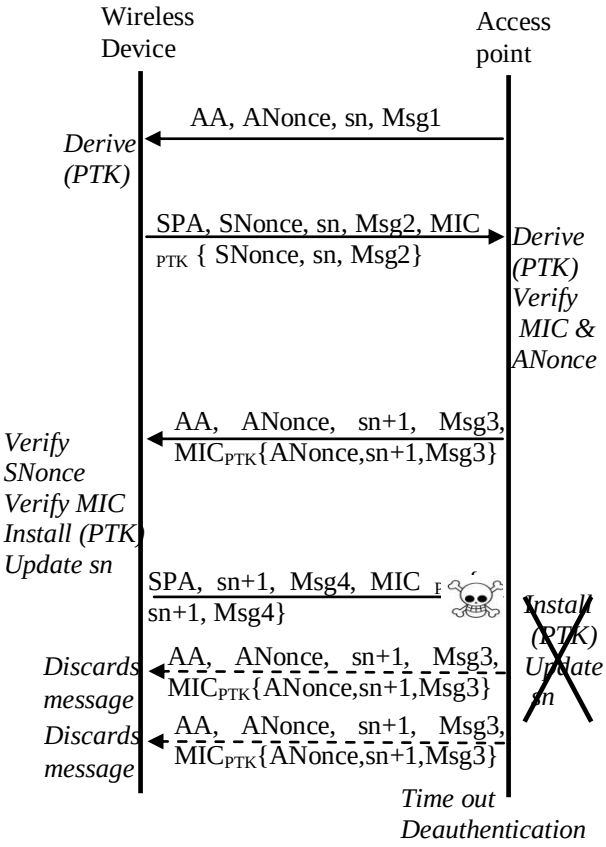

Figure 4: An Attack on the 4-way

lemma *FindAttackinFourhandshack* : S |= □ (
Discard SP ({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce, {Mag Msg3, Nonce (*SN1*)}} k}}}}))

apply(rule DiscardReceivedMessage)
apply(rule ReplayMessage3)
apply(rule NotReceivedMessage4FromSender)
apply(rule FHShake4)
apply(rule FHShake3)
apply(rule FHShake2)
apply(rule FHShake1)
done

PMK is important to reduce the authentication process costs that occur every time PTK is established or updated. If PMK has already existed they can run the 4-way handshake to obtain new PTK for transferring the data over the network. If the attacker can block message 4 then at the end of the 4-way handshake, PMK will be invalid and 802.1X authentication need to be run every time. As we know, the attacker has the ability to block any messages over the network, therefore it is easy for the attacker to block specifically message 4. Consequently, the attacker can de-authenticate the access point with all wireless devices wanting to connect during the 4-way handshake phase. The DoS attack identified in this paper is easy to implement over the network and it is difficult to prevent. A number of attempts to prevent DoS proposed by some researchers failed to defend against all DoS in the 4-way handshake.

6.2. Preventing the DoS attacks on message 4

A number of researchers have already discussed and proposed some solutions to avoid the DoS attack in the 4-way handshake. He and Mitchell provided two solutions to avoid DoS attacks on the wireless device side [11,18]. In addition, Rango *et al.* discussed the He and Mitchell solutions and introduced two new solutions to prevent DoS attacks [19]. Unfortunately, all these solutions are unsuitable to prevent the DoS attack identified in this paper. Therefore, we are going to introduce a new solution for this attack.

The *sn* value plays an important role in preventing replay attacks. The access point usually checks the *sn* value of received message corresponding to the outstanding message. Whereas, the wireless point checks the *sn* value used before with current PMK. Moreover, when the access point does not receive reply message during the timeout interval, it will keep re-sending the message until time out. The re-sent message has the same content as the original message and is valid from the point of view of the access point. The wireless device will likely discard the re-sent message if it has seen the original message, which is valid for the access point. So there is a contradiction between using *sn* value and re-sending the original message. In other words, the recipient will be confused with the re-sent message and the original message where both are valid.

In order to prevent from discarding the valid messages by the wireless device, the access point should update the *sn* value immediately after sending the message. So, when the access point wants to re-send the original message, a new *sn* value will be used. The *sn* value makes the resent message different from the original message and therefore the wireless device is not going to discard the resent message. As shown in figure 5, if the access point has not received message 4 during interval time out, then message 3 will be re-sent with the new *sn* value. As a result, the simple amendment will reduce the chance of message 4 being attacked. Also, the wireless device is not going to discard the valid messages [20].

6.3. Proving the fixed protocol using Isabelle

In order to prove that the DoS attack on the 4-way handshake can be prevented the replayed message 3 should be modified according to the proposed solution. Suppose that the access point should rename the *sn*1 value to become *sn* after sending message 3. Meanwhile, the updated *sn* value will become *sn1*. So, the replayed message 3 will be changed every time it is re-sent; therefore, it is not going to be discarded by the wireless device [20]. The replayed message 3 can be re-written as follows:

ReplayMessage3New : S |= :(Rcv SP ({Agent SP, {Mag Msg4, {Nonce (*SN*), MIC {Mag Msg4, Nonce (*SN*)} k}}})) ∧ (S< t) => t |= Send AP SP ({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce,{Mag Msg3, Nonce (*SN1*)}} k}}}}) ∧ (t < outtime) ∧ (intervaltime < t)

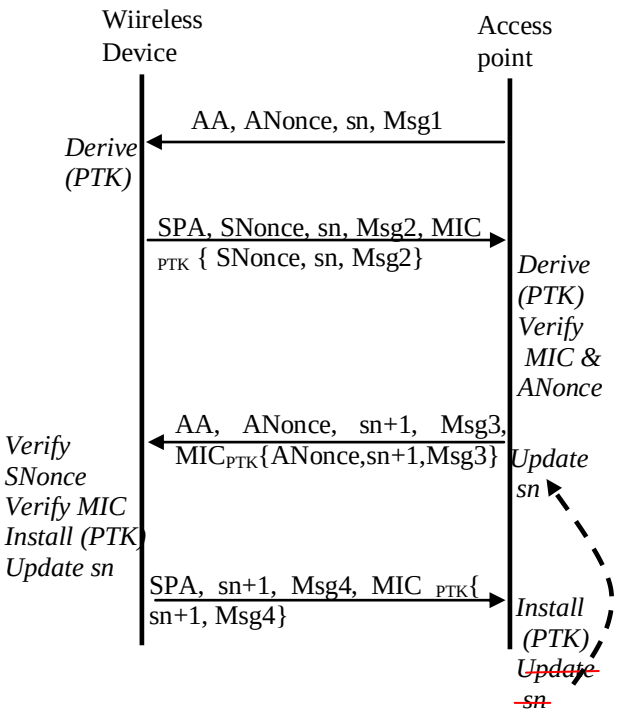

Figure 5: Prove the Updated 4-Way Handshake.

The following script shows that the access point will keep sending the message 3 until receiving message 4 or the finish time of the session. Whereas, the wireless device is not going to discard the re-sent message 3 because it is not the same as the previous message 3 which has been received.

FixDoSAttack : t |= Send AP SP ({Agent AP, {Nonce ANonce, {Mag Msg3, {Nonce (*SN1*), MIC {Nonce ANonce, {Mag Msg3, Nonce (*SN1*)}} k}}}}) ∧ (t < outtime) ∧ (intervaltime < t)

apply(rule ReplayMessage3New)
apply(rule NotReceivedMessage4FromSender)
apply(rule FHShake4)
apply(rule FHShake3)
apply(rule FHShake2)
apply(rule FHShake1)
done

7. Conclusion and future work

The 4-way handshake phase in the IEEE 802.11i standard has been analysed and a DoS attack has been identified. Isabelle tool has been used to implement the linear temporal logic framework. The adjustment of the framework, the modelling of the protocol and the proving of basic properties have been used for analysing the 4-way handshake. More importantly, a new effective DoS attack by blocking message 4 has been identified and analysed.

The protocol uses the *sn* value to avoid replay attacks in the 4-way handshake. However, the analysis has shown that the *sn* value will be a flaw if message 4 is not received by the access point. Non-receipt of message 4 can be caused by the attacker or anything else. In this case, the authentication between the wireless device and the access point will fail. Simply updating the *sn* value after sending message 3 can prevent the attack. Moreover, it is possible for the access point to obtain the reply message for message 3. A fixed version of the protocol has been proposed and the security of it has been proved using the framework with Isabelle.

8. Acknowledgements

This research is supported by Saudi Arabian Cultural Bureau in London, the Ministry of Higher Education in Saudi Arabia and Qassim University.

9. References

[1] Ma, X.; McCrindle, R.; Cheng, X. Verifying and Fixing Password Authentication Protocol. In Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006. SNPD 2006. Seventh ACIS International Conference On; pp. 324-329.

[2] Zha, X.; Ma, M. Security Improvements of IEEE 802.11 i 4-Way Handshake Scheme. In Communication Systems (ICCS), 2010 IEEE International Conference On; pp. 667-671.

[3] Edney, J.; Arbaugh, W.A. *Real 802.11 Security: Wi-Fi Protected Access and 802.11 I.*; Addison-Wesley Professional, 2004.

[4] Dong, L.; Chen, K.F.; Lai, X.J. Formal Analysis of Authentication in 802.11 I. Journal of Shanghai Jiaotong University (Science) **2009**, *1*, 023.

[5] Alabdulatif, A.; Ma, X.; Nolle, L. A Framework for Cryptographic Protocol Analysis using Linear Temporal Logic. In Information Society (i-Society), 2012 International Conference On; pp. 525-530.

[6] Alabdulatif, A.; Ma, X.; Nolle, L. A Framework for Proving the Correctness of Cryptographic Protocol Properties by Linear Temporal Logic. International Journal of Digital Society (IJDS) **2013**, *4*, 749-757.

[7] Boyd, C.; Mathuria, A. *Protocols for Authentication and Key Establishment*.; Springer Verlag, 2003.

[8] Junaid, M.; Mufti, M.; Ilyas, U.M. Vulnerabilities of IEEE 802.11 i Wireless LAN CCMP Protocol. Transactions on Engineering, Computing and Technology V **2006**, *11*.

[9] Xing, X.; Shakshuki, E.; Benoit, D.; Sheltami, T. Security Analysis and Authentication Improvement for IEEE 802.11 i Specification. In Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE; pp. 1-5.

[10] He, C. Analysis of Security Protocols for Wireless Networks. PhD thesis **2005**, *Stanford University*.

[11] He, C.; Mitchell, J.C. Security Analysis and Improvements for IEEE802.11i. In 11th Annual Network and Distributed System Security Symposium (NDSS'05), Feb.

[12] Wang, L.; Srinivasan, B.; Bhattacharjee, N. Security Analysis and Improvements on WLANs. Journal of Networks **2011**, *6*, 470-481.

[13] Latze, C.; Ultes-Nitsche, U.; Baumgartner, F. Strong Mutual Authentication in a User-Friendly Way in EAP-TLS. In Software, Telecommunications and Computer Networks, 2007. SoftCOM 2007. 15th International Conference On; pp. 1-5.

[14] Paulson, L.C. Inductive Analysis of the Internet Protocol TLS. ACM Transactions on Information and System Security (TISSEC) **1999**, *2*, 332-351.

[15] IEEE Standard for Information Technology-Telecommunications and Information Exchange between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Std 802. 11i-2004 **2004**.

[16] Bella, G. *Formal Correctness of Security Protocols*.; Springer Verlag, 2007.

[17] Eronen, P. Denial of Service in Public Key Protocols. In Proceedings of the Helsinki University of Technology Seminar on Network Security (Fall 2000).

[18] He, C.; Mitchell, J.C. Analysis of the 802.11 i 4-Way Handshake. In Proceedings of the 3rd ACM Workshop on Wireless Security; pp. 43-50.

[19] Rango, F.D.; Lentini, D.C.; Marano, S. Static and Dynamic 4-Way Handshake Solutions to Avoid Denial of Service Attack in Wi-Fi Protected Access and IEEE 802.11 I. EURASIP Journal on Wireless Communications and Networking **2006**, *2006*, 73-93.

[20] Alabdulatif, A.; Ma, X.; Nolle, L. Analysing and Attacking the 4-Way Handshake of IEEE 802.11i Standard. In The 8th International Conference for Internet Technology and Secured Transactions (ICITST-2013), London, Dec 2013; pp. 387-392.

