A New EAP Authentication Method for IEEE 802.11 Wireless

Younes El Hajjaji El Idrissi, Noureddine Zahid, Mohamed Jedra

Laboratory of Conception and System, Faculty of Science, Avenue Ibn Batouta, B.P.1014, Rabat, Morocco

Summary
Currently the wireless networks are facing more and more problems linked to security threats, which expose legitimate users to increased risk. User authentication is an important aspect in wireless network security, and Extensible Authentication Protocol (EAP) has been widely used. In this paper, we will analyze the existing EAP methods, and will propose a new EAP method: Extensible Authentication Protocol Public Key "EAP-PK". This method combines between the simplicity of deployment and management of password methods, and the robustness of certificated ones. It can also be used in the same application domains like other existing methods. We have checked the EAP-PK security properties (secrecy and authentication) using the specialized model checker AVISPA, which provides formal proofs of the security protocols.

Key words:
Wireless network; security protocol; Access control; EAP; HPSL.

1. Introduction

During the last three decades, the use of wireless communication technologies has been growing. This is due to the new applications (skype, voip, MSN, facebook, etc…) introduced in the multiple high technologies solutions such as laptops, smartphones and tablets. Most of these solutions use unsecured public networks to transmit confidential information, like user name, password, private or sensible data that require high security levels.

The first generation of wireless technologies had a bad reputation, due to their poorly designed security mechanisms WEP (Wired Equivalent Privacy). To address all serious weaknesses found, the IEEE has developed the 802.11i standard, which offers a strong security level by using the WPA (Wi–Fi Protected Access).

The WPA was invented by the Wi-Fi Alliance to secure the wireless network. Unlike in the WEP, the encryption key is recalculated more frequently, which leaves no time for an attacker to perform his attack. This is based on the TKIP (Temporal Key Integrity Protocol).

To reinforce the security process and to give more flexibility to the wireless network users, the IEEE invented the 802.1x in June 2001. This new release provides an intelligent authentication mechanism based on the authentication protocol EAP (Extensible Authentication Protocol) [1], which is defined by the IETF (Internet Engineering Task Force) [26].

The success of the EAP is the distinction between the EAP protocol and the EAP methods that are used. The principal function of the EAP protocol is to encapsulate the confidential data (login, password, certificate, etc.) used for the authentication. And the EAP methods take in charge the authentication process. As a result, protocols using the EAP are not attached to a particular EAP method, and in case a security fail is discovered, we can simply change this method without changing all the protocol or platform.

Currently, more than forty EAP methods exist, but only six of them are standardized in the IETF. Most of these methods suffer from several problems that make them vulnerable to several types of attacks.

This paper analyses the EAP based authentication methods, and proposes a new EAP method EAP-PK (Public Key), which combines between the simplicity of deployment and management of password methods (EAP-MD5, LEAP, etc), and robustness of certificated ones (EAP-TLS, EAP-TTLS, etc).

The remainder of this paper is organized as follows: section 2 and 3 describe the EAP authentication framework, section 4 briefly reviews the possible wireless attacks, section 5 provides overviews of a variety of EAP authentication methods, such as EAP-MD5 and EAP-TLS followed by a critical analysis, section 6 and 7 introduce and analyse the new proposed method, and section 9 illustrates the validation results of the EAP-PK by using the tool AVISPA presented in section 8. At last, we draw our concluding remarks in section 10.2. Tables, Figures and Equations

2. Extensible authentication protocol

EAP (RFC 2284) is an authentication protocol defined by the Internet Engineering Task Force (IETF) that typically rides on top of other protocols such as 802.1x, RADIUS, PPP. Through the use of EAP support, a number of authentication schemes may be added, including smart cards, Kerberos, Public Key Infrastructure (PKI), One Time Passwords (OTP), and others.

Manuscript received June 5, 2011
Manuscript revised June 20, 2011

The protocol 802.1x supports the EAP protocol as an authentication support protocol between the supplicant and the authenticator [1]. EAP typically runs directly over data link layers such as Point to Point Protocol (PPP) or IEEE 802, without requiring an IP address. The main advantage of the EAP architecture is its flexibility, because it is independent from the used authentication method. This protocol is named extensible protocol.

Some authentication methods are predefined like MD5, OTP, TLS [9], TTLS an SIM. These methods support authentication credentials that include digital ID, Password, certificates, secure tokens, and SIM secrets… Other methods can be added without changing the network protocol or defining new ones.

The RFC 4017[2] and RFC 3748 [1] define some mandatory, recommended and optional requirements for the EAP methods used in IEEE 802.11 wireless:

The mandatory requirements

The mandatory requirements can be considered as the base level functionality, which is required from an authentication method to provide basic security to the wireless network. The EAP method must support the following features:

- Generation of symmetric keying material

The EAP method must have the ability to generate an exportable key like a session key which can used for further key generation or data encryption.

- Key strength

The key strength is an important security parameter, the EAP method must have diverse key size generation (at least 128 bits).

- Mutual authentication support

The EAP methods must provide a mechanism by which the supplicant authenticates the network and the network authenticates the supplicant.

- Shared state equivalence

The EAP methods must provide a mechanism by which both parties (supplicant and server) can share some state attributes such as the method version number, the cryptographic keys shared, contributor names and ciphersuites.

- Resistance to dictionary attacks

When the EAP method uses a password authentication, the method must provide a protection to dictionary attacks.

- Protection against man-in-the-middle attacks

The EAP method should provide sturdiness to the man in the middle which tries to convince the user to connect to a rouge access point. In order to prevent this attack, no authentication functionality must be provided to the access point.

- Protected ciphersuite negotiation

The EAP method should protect the pre-negotiation used to select the cipher type to protect the EAP conversation.

Recommended requirements

Below a number of recommended requirements which should be supported by the EAP methods:

- Fragmentation

The EAP method should be able to segment or reassemble its transactions if the EAP message exceeds minimum MTU of 1020 octets.

- End-user identity hiding

As like the protected ciphersuite the EAP should hide the user identity by encryption or another method.

Optional requirements

The EAP authentication methods used for authentication may support the following features:

- Channel binding

The EAP method should support a mechanism by which server authenticator and supplicant can be conveyed to out of band devices such as lower-layer protocols.

- Fast reconnect

In the case of a lost a security association, the EAP methods should offer a mechanism to re-authenticate a client with a reduced number of message exchanges.

3. EAP Models

The 802.1X authentication framework provides Extensible Authentication to Link layer, which is transparent to upper layer (figure 1).

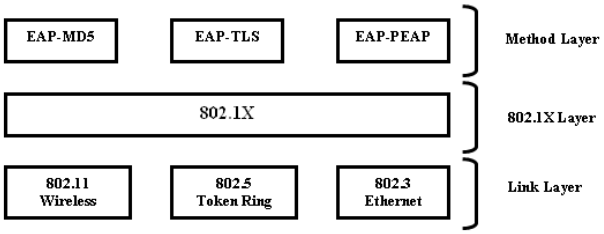

Fig. 1 EAP messages flow.

The EAP protocol introduces three principal entities. The EAP peer is the client to authenticate. The EAP authenticator corresponds to the entity that has control of the service (such as access point). And the EAP server is

the entity capable of authenticating the EAP client. The RFC 3748 [1] define two types of EAP models:

EAP multiplexing model

Using a layered model, the EAP multiplexing model can be illustrated in the figure 2

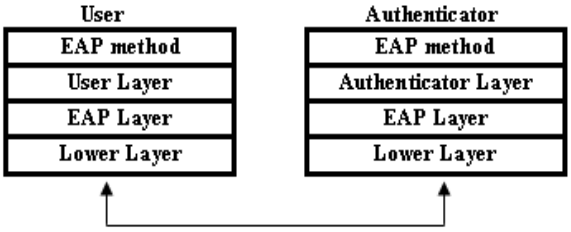

Fig. 2 EAP layered-model.

Lower layer is responsible for transmitting and receiving EAP frames between the user and the authenticator. This layer supports PPP, wired LAN, wireless LAN, etc. EAP layer detects and transmits the EAP packets from and to EAP peer or authenticator. EAP method layer implements the authentication algorithm.

In this model, the EAP server and the EAP authenticator are combined, the authenticator will implement all the authentication methods, or we could say that the authentication service is embedded in the authenticator.

Pass-Through behaviour model

When the EAP server is separated from the EAP authenticator, the EAP authenticator is said to act as an EAP pass-through authenticator, and the EAP server is referred as Back-end EAP authentication server. The EAP pass-through authenticator forwards EAP packets between the EAP peer and the EAP server and waits for a message called EAP-Success or a message called EAP-Failure messages from the EAP server indicating respectively the success or the failure of the EAP authentication. In case of successful authentication, the EAP authenticator grants the network access service to the client, in the opposite case, the client is rejected. As a result the authenticator only needs to understand the EAP success and failure message (EAP message with code with 3 (success) or 4 (failure)) and allowing EAP requests and responses to pass without understanding their content. The figure 3 illustrates the EAP pass-through authenticator layer architecture.

The EAP sets 4 types of EAP messages (request, response, success and failure). The figure 4 illustrates a typical exchange of a user authentication. The communication starts by the user which broadcast an EAPOL-Start frame. The authenticator reply by asking the user for user identity through an EAP-Request identity, encapsulated inside an EAPOL packet frame. After user identification, the peer and the EAP server agree to choose an authentication method, and the authentication process starts. The authentication server sends an EAP request with a specific type and the supplicant reply with an EAP response message of the same type. All types of EAP messages (EAP request, EAP response, EAP success and EAP failure) are encapsulated in EAP-Packet. The EAP request (from EAP server to supplicant) and EAP response (from supplicant to EAP server) transmit data, while the result of authentication is carried by EAP success and EAP failure messages.

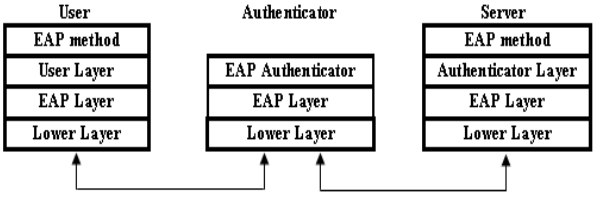

Fig. 3 Pass-through Authenticator

The authentication methods are facing to several types of attacks due to the open nature of the wireless network environment. In the next section we present a number of possible attacks.

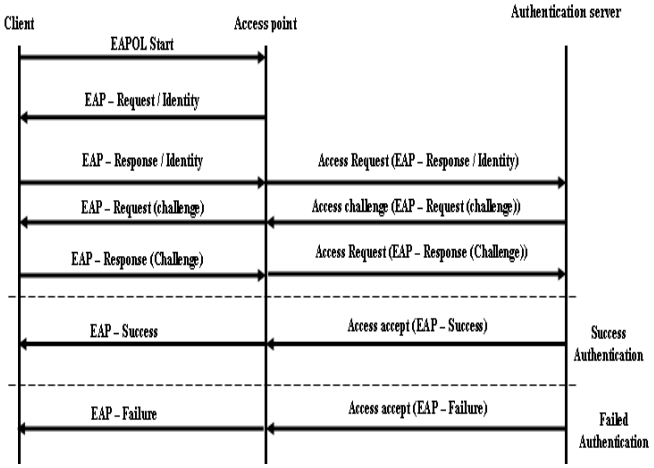

Fig. 4 EAP messages flow.

4. Possible Attacks

The possible attacks on the EAP method include [8]:

- Find user identities by reading unencrypted authentication exchanges,

- Spoofing EAP packets in order to collect such information as SSID and the channel of the AP. Denial of Service (DOS) attacks can use and modify the spoofed authentication responses, replay attacks and can also cut the session between client and legal access point or packets with overlapping identifiers.
- Dictionary attack or using a list of common passwords in order to attempt to gain access by simulating the authentication exchange offline.
- Man-In-The-Middle (MITM) attack in which an attacker mount a rouge access point between the client and the authentication into a trusted network [7].
- Interfering with negotiation of encryption parameters including the encryption type used in order to negotiate a less secure type which is easier to launch a subsequent attack against,
- Simulating an authenticator and providing false information to either the client or the EAP authenticating server and the encryptions algorithm flaws.

Generally these attacks take advantage of some flaws presented in the wireless technology and the EAP methods such as the shared communication channel, the lack of mutual authentication between the user and the authentication server and the weak cipher algorithm. To address these issues a number of authentication mechanisms are proposed like:

- ID/Password: it is the simplest method in which the user proofs his legitimate identity just by sending his user name and his password in clear.

- One-time passwords: to avoid several types of attacks in this mechanism each password is used one time by using a list of passwords or generating password by a card (SecureID card) that can calculate these passwords based on a predefined algorithm.

- Challenge/response [6]: the supplicant can use a pre-shared secret (password) to calculate the response to a challenge sent by the authentication server.

- Using a strong cipher algorithm: In this mechanism a secure channel is used to transmit the authentication between the two parties, the data is protected by adding encryption and integrity protection. This mechanism can support many authentication mechanisms such as password in the clear

- Zero-Knowledge password [25]: This mechanism allows authentications for both parties without pre-shared information (password, key) just by using the Diffie–Hellman algorithm.

- Certificates: This mechanism can be used to check the server / supplicant identity by using certificate provided by public key infrastructures (PKIs).

In the next section we present two type of EAP method using different authentication mechanism (EAP-MD5 and EAP-TLS).

5. EAP methods

EAP-MD5: This method is described in the RFC 2284[3] and illustrated by the figure 5. It is based on shared login - password between the supplicant and the authentication server, by using a CHAP mechanism (Challenge Handshake Authentication Protocol) - RFC 1994 [4]. To avoid the direct asking of the password, the server sends to the supplicant a random challenge value (message 4). And the supplicant confirms its identity by hashing the ID, password and the challenge number (message 5). The ID and the challenge request are 1 byte and 16 bytes in length respectively. The server should create a new challenge for the next challenge/response handshake. Each new challenge value must be accompanied by a challenge ID to distinguish that challenge from others.

The CHAP protocol defines challenge and response packets. Each including a CHAP header, the CHAP data, and other related fields as shown in table 1.

Table 1: CHAP challenge/ response packet

Field name	***Sub-Field***	***Description***
CHAP Header	CHAP Code CHAP Id CHAP length	01 for challenge / 02 for response 01 (first message) 00 23 (for 35 octets) / 00 18 (for 24 octets)
CHAP data	Value Size	10 (for 16 octets or 128 bits challenge or hash value)
	Value	A randomly generated 128 challenge or a 128 bits MD5 hash calculation (ID|| password || challenge value)
	Server or Peer name	Name authentication server or challenged user

This simple method provides some advantage like no need a PKI infrastructure, simplicity and fast authentication mechanism but it has several flaws like: No mutual authentication, no data privacy protection, no key generation over time (same issue as WEP) and no strong encryption method. For this it is vulnerable to many types of attacks like man in the middle and dictionary attacks.

The EAP MD5 does not derive the creation of Master Key, immediately after authentication the wireless client and access point would jump into WEP encrypted communications, which reduces the risks of eavesdropping, impersonation, or data corruption by a hostile attacker.

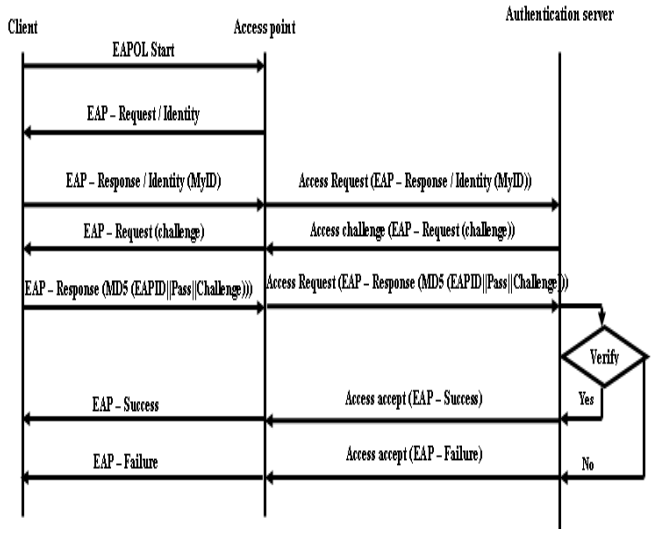

Fig. 5 EAP-MD5 messages flow.

EAP-TLS: To authenticate and to guarantee the mutual authentication between all participants (users and server) in the network, the EAP–TLS method uses client and server certificate. Managing certificates for all users can be the major downfall of this method, because the most organization does not have a Public Key Infrastructure (PKI).

The figure 6 illustrates the EAP-TLS message flow. After supplicant identity check (message 3) the communication continues with a handshake protocol, the server send the EAP-TLS/Start packet, and the supplicant will respond with a Client Hello message including the client's TLS version number, a session ID, a random number, and a set of cipher suites supported by the client (message 4). The server will choose a cipher suite from these supported by the supplicant and will reply by an EAP-TLS request packet which will contain a Server_hello message, server key exchange, public key and his certificate X509, (message 6). The supplicant verifies the certificate of the server and reply by his certificate and his public key. Based on the random numbers exchanged, the supplicant and the server define a cipher session key. The change_cipher_spec signals, the change of the key and the TLS_finished ends the hand check authentication. In the case of success check both parties send an EAP response message with no data.

In the face of the many advantages provided by this method there are some disadvantages as like:

Administration cost: This method requires a trusted third party to create and check the validity of the certificates. Therefore the cost of creation, maintenance and administration will be increased.

High protocol exchange: In order to complete the authentication process, this method requires a high number of protocol exchanges, between the user and the authentication server. This elongates the authentication delay for the user (which causes a problem in case of roaming and increase power consumption) and uses more computing resources on the authenticator.

EAP-TLS and EAP-MD5 are not ideal for operating in an environment IEEE 802.11 due to burdensome management, unilateral authentication and low resistance to several attacks (man in the middle, spoofing and dictionary attacks). Other EAP methods were standardized as EAP-OTP (One-Time-Password) and EAP-GTC (Generic Token Card) [1], which are simple to use, but have offered a unilateral authentication. EAP-SIM [20], and EAP-AKA (Authentication and Key Agreement) [21] have been specified to operate in GSM / UMTS environments but require smart cards.

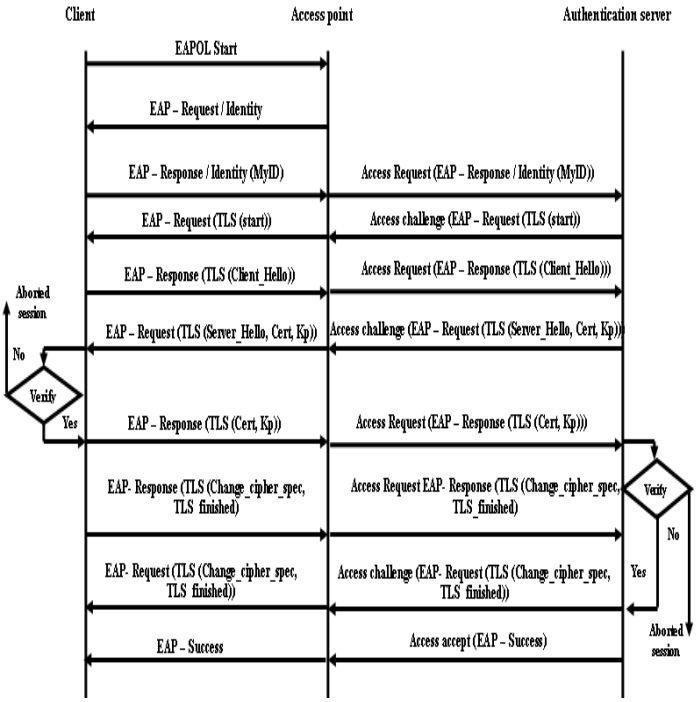

Fig. 6.EAP-TLS messages flow.

The table 2 presents a comparison between the most popular EAP methods and lists the attacks to which they may be vulnerable. As we see the choice of the authentication method has a strong impact on the

management system. All of the more sophisticated authentication methods (EAP-TLS, EAP-TTLS, EAP-

Tab 2: EAP method comparison

Method	*Encryption technologies*	*Vulnerability*	*Analyze*
EAP–MD5	One-way message digest	Dictionary attack Man-in-the-middle attack No mutual authentication	Easy to implement Supported by many server Use passwords in the clear No mutual authentication
EAP-TLS	Digital certificates X509	Strong authentication, resistant to attacks.	Using certificates in both side Strong but more complicated to manage Mutual authentication
EAP–TTLS	Encapsulation in a tunnel to complete authentication via login / password. Digital certificates x509 or Diffie–Hellman algorithm to generate keying material Symmetric key for data encryption	Strong authentication, resistant to attacks.	Creating a secure TLS tunnel Supports PAP, CHAP, MS-CHAP, MS-CHAPv2 Certificate mandatory server side, client side optional Mutual authentication
EAP–PEAP	Encapsulation in a tunnel to complete authentication via login / password. Digital certificates x509 or Diffie–Hellman algorithm to generate keying material Symmetric key for data encryption	Strong authentication, resistant to attacks.	Similar to EAP-TTLS Mutual authentication
EAP–LEAP	Diffie–Hellman algorithm to generate keying material • Symmetric key for data encryption	Dictionary attack.	CISCO Proprietary solution Not supported in Windows Mutual authentication

PEAP, etc.) necessitate a high infrastructure (PKI) and an administration service, which complicate the network design and increase the maintenance cost. In the other hand the weakest methods (EAP-MD5, LEAP, etc.) are easy to implement, does not require a complicated infrastructure and are simple to employ.

In the next section we propose a new EAP method called EAP-PK, which combines between the advantage of using an asymmetric cipher algorithm and the simplicity of using symmetric key management. It is backward compatible with the existing EAP protocol since it does not require any change in the 802.X standards and combines between the low cost and the high authentication security.

6. The proposed method : EAP-PK

To provide a mutual authentication and data integrity check this method uses an asymmetric cipher algorithm like RSA, Pohlig-Hellman, etc. Normally for each entity the asymmetric algorithms use a different pair of keys (Ke, Kd), Ke is the public key used for data encryption and Kd is the private key used for data decryption.

The mutual authentication method proposed in this paper, uses a unique pair of keys (Ke, Kd). This pair of keys is considered as a pre-shared secret key between the client and the EAP authentication server. Both parties can prove to each other that they know the secret key (Ke, Kd) for preventing third party access to the conversation. The EAP-PK can be classified as a password method. It does not require a PKI infrastructure, and it can be built into simple wireless access point (AP). The used pair of keys can be generated by the authentication server and communicated to the client via a secure channel independent of 802.11 WLAN (email, post mail, etc)

As shown in figure 7, the authentication process of EAP-PK consists of 5 steps described below:

1) After receiving the EAP_Start message, the authenticator (AP) replies by sending the EAP-Request/Identity message to the user.
2) The user station answers with an EAP-Response /Identity message, containing the user identity (MyID). This is the first actual exchange that informs the server on the identity of the user.
3) Then, the EAP server looks for the user information in its access data base, retrieves the user's authentication keys (Ke, Kd), and continues the authentication

protocol by sending an EAP Request/Challenge message. This message contains an authentication token AUTN = {challenge}$_{Ke}$ (generated challenge number encrypted with Ke) concatenated with server ID and a message integrity check (MAC), MAC = {challenge || ServId || UserId }$_{Ke}$ (encrypted with Ke).

4) When the EAP-Request/Challenge message is received, the user station decrypts the AUTN with the Kd key, calculates a local MAC'= {challenge || ServId || UserId }$_{Ke}$ and compares it with the received MAC. If they are not equal, the user station aborts the attempted session. If they are equal, this means the EAP server is legitimate. In the case of a positive verification the user station responds with an EAP Response/Challenge message that includes a generated random number RAND concatenated with RES= {RAND || challenge|| UserId}$_{Ke}$ (encrypted with Ke). The AUTN and MAC permit to the user station to authenticate the EAP server, and to check the message integrity.
5) After receiving the user station response, the EAP authentication server calculates a local RES'= {RAND || challenge|| UserId}$_{Ke}$ (encrypted with Ke), and compares it with the received RES from the user station. If they are equal, this means the user station is legitimate; the EAP server will send an EAP-Success packet to the access point (AP) to allow access to the supplicant. If they are not equal, the user station is not legitimate and the authentication server signals failure to the AP. The RES is used to check the supplicant authentication and the message integrity.

The server and the user station can use the shared key (Ke, Kd) to exchange a session key for future communication.

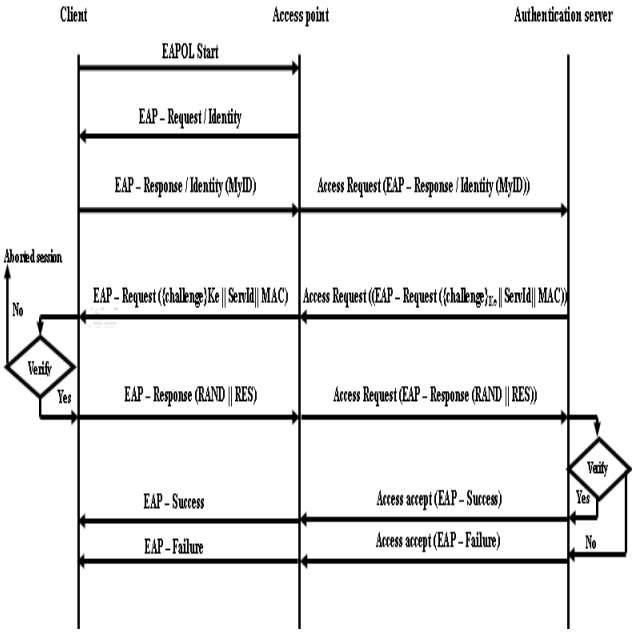

Fig 7: EAP-PK messages flow

7. EAP-PK Analysis

The EAP-PK does not only address all the mandatory features required by the RFC 4017 [2]; it also has several advantages in comparison to other EAP methods:

Mutual authentication: Compared to EAP-MD5, EAP-PK offers to the supplicant the possibility to authenticate the server. He can now detect more easily the rouge access point and check the message integrity by decrypting the received challenge and comparing the calculated and the received MAC.

Quick authentication: Unlike the EAP-TLS method, EAP-PK is based on challenge/ response mechanism with a reduced number of exchanged packets, which offers a quick authentication process.

Implementation cost: The EAP-PK method can be implemented without using a PKI infrastructure, therefore reducing implementation and maintenance costs.

Confidentiality: The confidentiality is guaranteed by using a strong encryption algorithm like RSA, Pohlig-Hellman, etc.

Protection against the man in the middle attack: Made possible by the strong cipher used and the privacy of the pre-shared key (Ke, Kd).

Protection to the replay attack: The EAP-PK method is robust to the replay attack because the challenge and

RAND are generated for each new authentication and are used one time.

The proposed method is evaluated by using the formal security verification platform AVISPA. In the next section of this paper, we will present the AVISPA tools and discuss the validation results of the EAP-PK

8. AVISPA Description and architecture

Network security protocols, such as key-exchange and key-management protocols, are difficult to design and to debug. The formal verification is a logic for proving security properties of network.

In the last decade the formal verification of security protocols has been booming and was the subject of intense research. This gave birth to a number of verification tools, like Murphi [10], CSP [11], FDR [12], NRL protocol analyzer [13], Isabelle [14] and AVISPA [15].

The main goal of this section is to briefly describe the Automated Validation of Internet Security Protocol and Applications tool (AVISPA).

AVISPA is an automatic push-button formal validation tool for internet security protocols. It has been developed in a project funded by the European Commission under the Information Society Technologies IST programme. AVISPA is based on sending and receiving messages, and performing decryption and digital signature verification actions.

AVISPA takes as input a High Level Protocol Specification Language (HLPSL) for describing security protocols and specifying their intended security properties. HLPSL is an explicit and intuitive language to model a protocol, its semantics is based on Lamport's Temporal Logic of Actions (TLA) [16][17]. The HLPSL is based on roles; each protocol is divided into a set of Basic Roles representing the actions of one single agent in a run of the protocol, and Composition Roles which represent the entire protocol and instantiate the Basic Roles. Each role is modelled as a 'state'. Each state has variables which are responsible for the state transitions, retrieves its initial information by parameters, and communicates synchronously with other roles by channel [17]. The security goal is the most important feature of this tool. It allows the model checkers to find the possible attacks. In general, authentication is modelled by these words: witness, request, wrequest and secret. The figure 8 shows the structure of the AVISPA Tool [2].

Once the protocol is modelled in HLPSL, AVISPA translates them into a lower-level language Intermediate Format (IF) by a translator called hlpsl2if. IF is executed directly by the back-ends tools (OFMC, CL-AtSe, SATMC and TA4SP) to verify if the security goals are satisfied or violated. For more information about the back-end tools refer to AVISPA user manual [22].

The AVISPA tools and HLPSL language are a very popular formal verification pack. However, the differences between the specification language and the notation User and Server, particularly the definitions role by role and not message by message, make this pack difficult to use. For this reason, a new tool "Security Protocol Animator" (SPAN) was created to facilitate the specification phase by allowing the animation of the language HLPSL [23].

SPAN can be used to design and to verify the rightness of the formally modeled protocol; it helps to simulate the designed protocol using HLPSL specifications and to build Message Sequence Charts (MSC) of the protocol [24]. SPAN also allows checking the generation of nonce values and message texts. Since SPAN implements an active intruder, it can also be used to interactively find and build attacks on protocols.

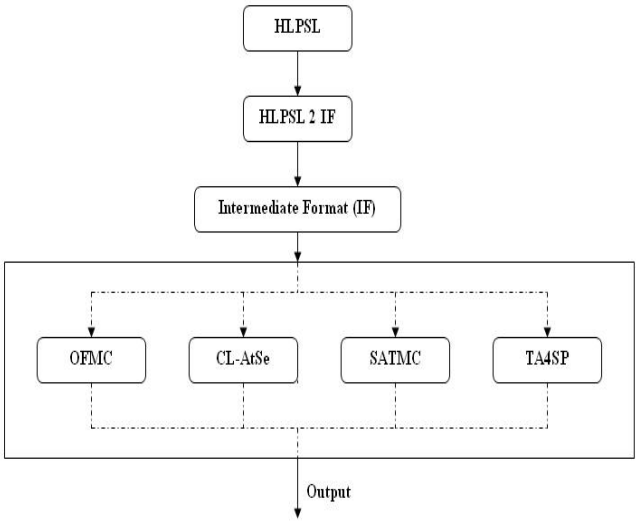

Fig. 8. Architecture of the AVISPA tool

9. Specification and validation

This section presents the validation results of the EAP-PK, obtained by using the tools AVISPA and SPAN. Since the authenticator only passes through the authentication messages between the peer and authentication sever, the authenticator can be omitted in the formal verification.

EAP-PK protocol is defined in User (user station) and Server (authentication server) Model and then is coded in the formal language HLPSL used in AVISPA. The correctness of the written HLPSL code is checked using

the protocol animation tool SPAN. Then, the protocol is analyzed by executing the AVISPA tools. The figure 9 presents an extract of the EAP-PK specification in the HLPSL language.

We assume that the user station and the authentication server had a pre-shared pair of key (Ke, Kd) in advance. The server then generates a nonce value (challenge) while the client generates RAND. After protocol verification with SPAN, the intruder simulation was done to check the robustness and whether it makes any abnormal flaws in the protocol run. The HPSL code and the follow goals were verified:

Mutual Authentication: The supplicant authenticates the server by comparing the calculated MAC value with the received MAC value, which proves that the server knows the pre-shared key Ke. The verification is done by:

MAC'= {Challange'.server_id.client_id}_Ke

In the other side the server authenticates the supplicant by checking with the received value RES:

RES' = {Challange'.RandA'.server_id}_Ke

Key secrecy: The instruction secret (Ke,sec_SK,{A,S}) asserts that the Ke should be kept secret between the A (client) and the S (server).

```
role server(S,A : agent,
            Ke      : symmetric_key,
            SND_S, RCV_S: channel (dy))
played_by S
      def=
  local
            State                          : nat,
            Challange, RAND                : text,
            MAC,RES                        : message

const
     response_id,client_id,start_eap,server_id  : text
  init
      State := 1

  transition
      1.    State     = 1 /\ RCV_S(start) =|>
            State'    := 3/\ SND_S(request_id)

      2.    State     = 3 /\ RCV_S(response_id.A) =|>
            State'    := 5 /\ Challange' := new()
                              /\ MAC' := {Challange'.S.A}_Ke
                              /\ SND_S({Challange'}_Ke.S.MAC')
                              /\ witness(S,A,auth1,Challange')
                              /\ secret(Challange',sec_challange,{A,S})

      3.    State     = 5 /\ RCV_S(RAND'.RES)
                              /\ RES' = {Challange'.RAND'.S}_Ke  =|>
            State'    := 7 /\ SND_S(success)
                              /\ request(S,A,auth,RAND')
                              /\ secret(Ke,sec_SK,{A,S})
end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role client( A,S: agent,
             Ke      : symmetric_key,
                       SND_A, RCV_A: channel (dy))
played_by A
      def=
  local
      State          : nat,
      Challange,RAND  : text,
      MAC, RES        : message
      const
      sec_SK          : protocol_id,
      sec_challange   : protocol_id,
      auth            : protocol_id,
      auth1           : protocol_id,
      request_id      : text,
      response_id,client_id,start_eap,server_id,success  : text

  init  State := 0

  transition

      %1.   State   = 0 /\ RCV_A(start) =|>
      %     State' := 1 /\ SND_A(start_eap)

      1.    State   = 0 /\ RCV_A(request_id) =|>
            State' := 2 /\ SND_A(response_id.A)

      3.    State   = 2 /\ RCV_A({Challange'}_Ke.S.MAC')
                           /\ MAC'= {Challange'.S.A}_Ke  =|>
            State' := 4 /\ RAND':= new()
                              /\ RES' := {Challange'.RAND'.S}_Ke
                              /\ SND_A(RAND'.RES')
                              /\ witness(A,S,auth,RAND')
                              /\ request(A,S,auth1,Challange')

end role
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Fig. 9 Extract of the EAP-PK specification in the HLPSL language.

Attack robustness: The witness and request events' goal is to authenticate the source of the message. Witness (S, A, auth1, Challange') signifies "agent S asserts that he wants to be the peer of agent A, agreeing on the challenge value". Request (A, S, auth1, Challange') means "agent A accepts the challenge value and now relies on the guarantee that agent S exists and agrees on this value". This means that the supplicant and the authentication server have the correct and same encryption and decryption keys (Ke, Kd). The supplicant is able to authenticate the server on the challenge value and the server is able to authenticate the supplicant on the RAND value. These roles permit to detect several types of attacks such as: Man in the Middle and dictionary attack.

Replay attacks protection: One time use of challenge and RAND values allows the EAP-PK method to be robust to the replay attack in which the intruder replays old

message from a previous protocol run or by specifying multiple parallel sessions between the same agents.

The figure 10 presents a simulation part of intruder attacks simulation obtained by SPAN with 2 parallel sessions. And figure 11 shows the result of the OFMC protocol verification. As we can see, no attacks were detected by the OFMC and all the stated security goals were satisfied.

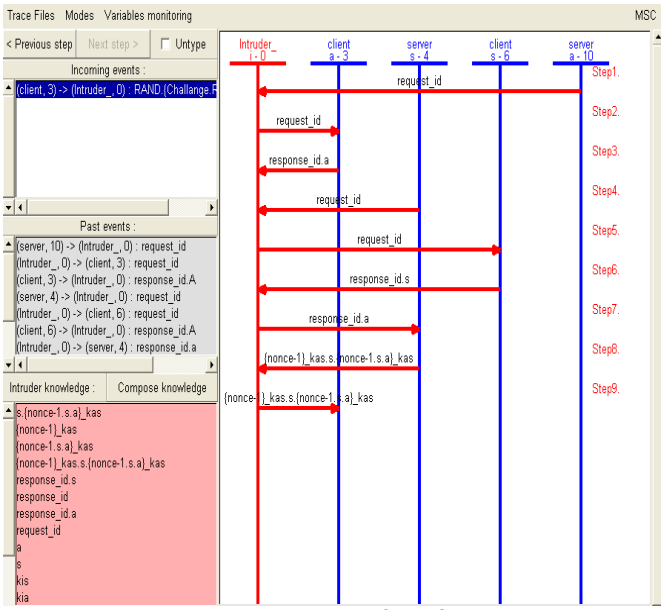

Fig. 10. EAP-PK SPAN attack simulation

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  C:\progra~1\SPAN\testsuite\results\EAP-XX.if
GOAL
  as_specified
BACKEND
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 0.09s
  visitedNodes: 55 nodes
  depth: 8 plies
```

Fig 11: EAP-PK OFMC protocol Verification

10. Conclusion

The EAP protocol gives dynamicity and flexibility to the IP networks. However the existed EAP methods do not offer the expected properties for a secure authentication and easy implementation. In this paper we proposed a new EAP method called EAP-PK which offers interesting properties of fast and mutual authentication, simplicity of use and robustness to man in the middle, DOS and offline attacks. The proposed method can be deployed inside wireless networks without using a PKI infrastructure or changing the existed network hardware. To simplify the use of this method, the pre-shared key can be generated from a password shared between the client and the authentication server.

References

[1] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Extensible Authentication Protocol , RFC 3748, June 2004.

[2] D. Stanley, J. Walker, B. Aboba, Extensible Authentication Protocol Method Requirements for Wireless LANs, RFC 4017, March 2005.

[3]L. Blunk, J. Vollbrecht, PPP Extensible Authentication Protocol, Merit Network, RFC 2284 March 1998

[4] W. Simpson, PPP Challenge Handshake Authentication Protocol, RFC 1994, August 1996.

[5] T. Dierks, C. Allen, The TLS Protocol Version 1.0, IETF RFC 2246, January 1999.

[6] M Falk, Fast and secure roaming in WLAN, Final thesis, Department of Computer and Information Science, Linkoping university, December 2004.

[7] H. Hwang, G. Jung, K. Sohn, S. Park, A Study on Man in the Middle Vulnerability in Wireless Network Using 802.1X and EAP, International Conference on Information Science and Security , Seoul, Korea, 2008, pp 164-170.

[8] R. Dantu, G. Clothier, A. Atri, EAP methods for wireless networks, Computer Standards Interfaces 29 (3) (2007) 289–301.

[9] B. Aboba, D. Simon, RFC 2716, PPP EAP TLS Authentication Protocol, the Internet Society, Oct. 1999.

[10] J.C Mitchell, M. Mitchell, U. Stern, Automated Analysis of Cryptographic Protocols Using Murphi, IEEE Symposium on Security and Privacy, IEEE Computer Society Press (1997) 141-151.

[11]S. Schneider, Verifying Authentication Protocols in CSP, IEEE Transactions on software engineering, 24 (1998) 741–758.

[12] A. W Roscoe, Modelling and verifying key exchange protocols using CSP and FDR, In Proceedings of 8th IEEE Computer Security Foundations Workshop, IEEE Computer Society Press, County Kerry, 1995.

[13] C. Meadows, The NRL Protocol Analyzer: an overview, Journal of Logic Programming, 26(1996) 113–131.

[14]University Of Cambridge, http://www.cl.cam.ac.uk/research/hvg/Isabelle/overview.html, updated 12-07 2006.

[15] A. Armando, D. Basin, J. Cuellar, M. Rusinowitch, L. Viganò, The AVISPA Tool for the Automated Validation of Internet Security Protocols and Applications, CAV 2005, LNCS 3576, 2005, pp 281–285.

[16] L. Lamport, The temporal logic of actions. ACM Transactions on Programming Languages and Systems, 16(3), May 1994 872–923.

[17] Y. Chevalier, L. Compagna, J. Cuellar, P. Hankes Drieslma, J. Mantovani, S. Modersheim, L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS 2004), 2004.

[18] Shannon, Claude, Communication Theory of Secrecy Systems., Bell System Technical Journal 28 (4) 656–715.

[19] K. Arjen, Lenstra, R. Eric, Selecting Cryptographic Key Sizes, Verheul, PKC2000, 446-465.

[20] H. Haverinen, J. Salowey, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM). RFC 4186, Jan. 2006.

[21] J. Arkko, H. Haverinen, Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA), RFC 4187, Jan 2006.

[22] Y. Glouche, T. Genet, SPAN – a Security Protocol Animator for AVISPA – User Manual. IRISA / Universit´e de Rennes 1, http ://www.irisa.fr/lande/genet/span, 2006.

[23] Y. Boichut, T. Genet, Y. Glouche, O. Heen, Using animation to improve formal specifications of security protocols, In Proceedings of SAR-SSI'07, 2007.

[24] D. Harel, P. S. Thiagarajan. Message sequence charts. UML for Real : Design of Embedded Real-time Systems, 1 ed, 2003.

[25] S.Bellovin, M. Merritt, Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks, Proceedings of IEEE Symposium on Research in Security and Privacy, Oakland, May 1992.

[26] The Internet Engineering Task Force (IETF). http://www.ietf.org.

Younes El Hajjaji El Iidrissi, was born in Rabat, Morocoo in 1980. He graduated from Mohammed V University in Rabat, Morocco, and received the M.S degree in Info-Telecom science in 2006. Currently, he is preparing his Doctorat National degree in computer science. His research interests are: wireless network, security protocol and access control.

Noureddine Zahid, holds « Doctorat de Troisième Cycle » and « Doctorat d'Etat » degrees in Electronics Engineering and Informatics; all form the Mohammed V University in Rabat, Morocco. He was an Assistant Professor in computer science at the Department of Physics between 1988 and 1999, and Professor "Habilité" between 1999 and 2003. In 2003, he was promoted to the position of Professor in the same department. He is a member of Conception & Systems Laboratory. He is the co-founder and the current Chair of the Informatics Telecommunications and Imagery master (ITI) in the Faculty of Science in Rabat. He has presented and published many articles in scientific journals and conferences. He has reviewed numerous papers for international journals in the field of fuzzy systems and pattern recognition. His research interests lie in the areas of fuzzy logic, pattern recognition, machine learning and biometric.

Mohamed Jedra, received the « Doctorat de Troisième Cycle » and « Doctorat d'Etat » degrees in Electronics Engineering and Informatics; respectively in 1990 and 1999, from Mohammed V University in Rabat, Morocco. From 1990 to 1999, he was the Network and Internet Center Manager in the Faculty of Sciences and Assistant Professor at the Department of Physics. In 1999, he became a Professor Habilité of Informatics in the same Department and in 2003 he was promoted to the position of Professor. From 1987 to the present, he was a member of the Conception & Systems Laboratory. He is the co-founder and the current Chair of the Architecture of Informatic Systems UFR / Formation and Research Unit in the Faculty of Sciences in Rabat since 2003. His main research interests include computational intelligence, pattern recognition and biometric. He was a reviewer of several conferences dealing with topics related to computational intelligence and pattern recognition. He has published numerous refereed papers in specialised journals and conferences. Mohamed Jedra is a member of IEEE since 1995. He is also a member of IEEE Computer Society, IEEE Computational Intelligence Society and IEEE Signal Processing Society.