Gazi University Journal of Science
GU J Sci
26(4):563-570 (2013)

User Identity Protection in Wireless Local Area Networks

Ç. KURT[1, ♠], S. ÖZDEMİR[2]

[1] *Gazi University, Institute of Computer Science, Gazi University, Ankara, TURKEY*
[2] *Gazi University, Department of Computer Engineering, Gazi University, Ankara, TURKEY*

Received: 08.02.2013 Revised: 13.07.2013 Accepted:02.11.2013

ABSTRACT

Wireless networks are susceptible to security attacks due to their open transmission media. Therefore, wireless network security is somewhat more complex than that of wired network security. In wireless network security, authentication is the most essential procedure to ensure that the service is properly used by the intended users. Protected Extensible Authentication Protocol (PEAP) is widely being used in wireless networks. However, PEAP is shown to be a weak protocol in terms of protection of user identity. In this paper, we have designed and implemented a new and efficient wireless authentication protocol providing user identity secrecy. This method takes advantage of PEAP's management easiness and the robustness of dynamic key distribution to provide user identity protection.

Key words: wireless security; authentication; encryption; dynamic key.

1. INTRODUCTION

Wireless networks are becoming more and more popular. The use of wireless communication technologies and wireless networks has been growing. This is due to not only availability and mobility easiness of wireless networks but also the new applications introduced in laptops, smartphones and tablets. Most of these applications use unsecured public networks to transmit confidential information, like user name, password, private or sensible data that require high security levels. Wireless communication medium is, by its nature, vulnerable to variety of threats, including unauthorized access, eavesdropping of communication, modification and repetition of data, denial of service, and fabrication of data. In addition, wireless networks are vulnerable to the unauthorized (rogue) devices are relatively easier to connect to the network because they do not need any physical access. Therefore, the wireless security and user authentication are becoming more important issues as the wireless networks are becoming much popular.

IEEE introduced 802.1x [1] in June 2001 to reinforce the security process and to give more flexibility to wireless network users. This standard provides an intelligent authentication mechanism based on the

[♠]Corresponding author, e-mail: moteyaka@dpu.edu.tr authentication protocol Extensible Authentication Protocol (EAP) [2], which is defined by Internet Engineering Task Force (IETF) [3]. EAP supports multiple authentication methods and typically runs directly over data link layer protocols such as Point-to-Point Protocol (PPP) or IEEE 802, without requiring IP.

The success of EAP is the distinction between EAP itself and EAP based methods that are being used. The principal function of EAP is to encapsulate the confidential data (username, password, certificate, etc.) used for the authentication. EAP methods are responsible achieving the authentication process. As a result, authentication protocols using the EAP are not attached to a particular EAP method. However, when a security pitfall is discovered, we can simply change the faulty method without changing all the protocol or platform. EAP [2] supports many types of authentication methods.

The security requirements of a wireless communication system includes confidentiality on the air interface, anonymity of the user and, most importantly, authentication of the user to the network in order to prevent fraudulent use of the system. For enhancing the efficiency and the security in anonymous channel of wireless systems, many authentication protocols were proposed [4]. Currently, more than 40 EAP based authentication methods exist, but only six of them are standardized by the IETF. One of these methods is PEAP [5].

PEAP is a joint proposal by Cisco Systems, Microsoft and RSA Security as an open standard. It is already widely used authentication method with strong protection against the deployment of unauthorized access. PEAP is comprised of a two-part conversation. In Part 1, a Transport Layer Security (TLS)[6] session is negotiated, with server authenticating to the client and optionally the client to the server. The negotiated key is then used to encrypt the rest of the conversation. In Part 2, within the TLS tunnel, a complete EAP conversation is carried out, unless Part 1 provided client authentication.

Because PEAP is a tunneled method, it provides secrecy for users' identities from eavesdroppers by hiding the EAP Response Identity message in the Phase 2 of secure TLS tunnel. However, there is a weakness that depends on users' preferences for the Phase 1. The user identity is not actually protected in the Phase 1 because of using anonymous identity privacy is optional. So, it makes the wireless networks weak against the attacks targeting user identity.

In this paper, we propose a new authentication method for wireless networks to overcome the user identity problem in Phase 1 of PEAP. The proposed protocol is integrated with PEAP. The solution takes advantages of PEAP's management easiness, ease of deployment and the robustness of dynamic key distribution.

The rest of this paper is organized as follows: Related work is given in Section 2. We present the proposed authentication protocol E-PEAP and the integration with EAP in Section 3. The implementation details are explained in Section 4. Section 5 presents the security analysis and Section 6 concludes the paper.

2. RELATED WORK

In this section, we survey recent Wireless Local Area Network (WLAN) authentication protocols. We group the protocols into three categories, according to their authentication mechanism: shared-key methods, public-key methods and tunneled methods.

2.1. Shared-Key Methods

In shared-key authentication method which is defined by IEEE 802.11, a pre-shared key is a shared secret which was previously shared between the two parties using some secure channel before it needs to be used. Because the same key is shared between the authenticating parties, shared-key methods are also known as secret-key or symmetric-key methods To build a key from shared secret, the key derivation function should be used. Such systems almost always use symmetric key cryptographic algorithms.

Unlike in wired LANs, in WLANs it is easy to eavesdrop on the communications between the authentication server and the client. This is due to the fact that most shared-key authentication protocols derive the shared secret from the user's password and because most users choose weak passwords. This makes it easy for the attacker to gather enough encrypted messages and extract the secret key from them, using dictionary attacks [7]. Although some shared-key authentication methods, do protect the client's password from dictionary attacks, these methods require much greater computational power than other shared-key methods [8].

2.2. Public-Key Methods

Public key authentication method is the only method that each software (both client and server) is required to implement. Public key cryptography is based on very complex mathematical problems that require very specialized knowledge. Public key cryptography makes use of two keys, one private and the other public. The two keys are linked together by way of an extremely complex mathematical equation. The private key is used to decrypt and also to encrypt messages between the communicating machines. Both encryption and verification of signature is accomplished with the public key. The integrity of a public key is usually assured by completion of a certification process carried out by a Certification Authority (CA). Once the CA has certified that the credentials provided by the entity securing the public key are valid, the CA will digitally sign the key so that visitors accessing the material the key is protecting will know the entity has been certified. Clients are assumed to have, in advance, a copy of the CA's public key to use for validating certificates.

The requirement of well-implemented CAs makes most public key methods considerably more complicated to deploy than the secret-key methods, however [9]. In the absence of proper CAs, an imposter might be able to advertise his public key as the Authentication Servers' (AS) public key since there is no CA to verify that the key belongs to the authentication server.

2.3. Tunneled Methods

Protocols of this approach have been proposed in the Internet Engineering Task Force (IETF) for running EAP inside a authenticated tunnel. Examples of such protocols are PIC [10], PEAP [5], EAP-TTLS [11] and most recently SLA [12], which led to the mechanisms for supporting legacy authentication methods in IKEv2 [13]. Some of these protocols, like PEAP, are motivated by a wish to correct perceived weaknesses of EAP, such as the lack of user identity protection and lack of a standardized mechanism for key exchange. All of these new protocols are constructed in the same basic manner. First, a server authenticated tunnel is set up using a suitable protocol like TLS. Then, the client authentication protocol is run inside this tunnel. Other forms of tunneled authentication protocols, such as HTTP Digest authentication inside a TLS tunnel, also conform to this general model.

In this method, the user uses the resulting session key to establish an encrypted tunnel to encrypt their communication.. The tunnel has two purposes. First, as mentioned earlier, it allows use of a less secure legacy protocol for client authentication in Phase 2. Recall that for mutual authentication, EAP-TLS [14] requires the client to have a certificate issued by CAs that the authentication server trusts. Because the encrypted tunnel from the Phase 2 hides the content of the messages sent during the Phase 2, the client and the AS can be sure that the client authentication is as secure as EAP-TLS, but without requiring a CA that supports client with. Thus, PEAP and EAP-TTLS can provide mutual authentication that is as secure as EAP-TLS even when only legacy client-authentication methods are available. Second, using the tunnel hides the client's identity from an eavesdropper by hiding the EAP Response- Identity message in the encrypted tunnel. To do so, in Phase 1 of the authentication process, the client's EAP Response-Identity message contains a generic domain name instead of the username. But it is optional, not mandatory. This optional selection causes vulnerability for the network. The next section explains the other vulnerabilities and purposes our solution to eliminate these vulnerabilities.

2.3.1. PEAP

PEAP is a member of the family of EAP. PEAP uses TLS to create an encrypted channel between the client and the authentication server. PEAP provides additional security for the client-side EAP, such as EAP-MS-CHAPV2, that can operate through the TLS encrypted channel. Therefore, the EAP messages encapsulated inside the TLS tunnel are protected against various attacks. PEAP comprises two phases. In Phase 1, a TLS session is negotiated and established. The client also authenticates the server by using a certificate. In Phase 2, EAP messages are encrypted by using the key negotiated in Phase 1 and the client is then authenticated using a username and a password, which are protected by the TLS tunnel.
PEAP has many advantages. Not only does the tunnel provide identity privacy for Phase 2, but it can also provide delegation if the client authentication method requires delegation. Moreover, even when the client authentication protocol is vulnerable to dictionary attacks or replay attack, in the tunneled second phase it becomes no longer vulnerable to these attacks because the eavesdropper sniffing the tunneled session must break the secure EAP-TLS tunnel to mount these attacks on the client authentication. The RFC 4017 [15] and RFC 3748 [16] define some mandatory, recommended and optional requirements for the EAP methods used in IEEE 802.11 wireless and PEAP provides all of those recommendations.

PEAP uses a TLS channel to protect the user credentials. Using the TLS channel from the client to the authentication server, PEAP offers end-to-end protection, not just over the wireless data link, ratified. Within the TLS channel, PEAP hides the EAP type that is negotiated for mutual client and server authentication. Also, because each packet sent in the TLS channel is encrypted, the integrity of the authentication data can be trusted by the PEAP client and server.

PEAP is an open standard supported under the security framework of the IEEE 802.1x specification and supports any EAP compatible methods. PEAP is also defined as an extensible authentication method that can embrace new EAP authentication schemes as they become. It also provides support for EAP-TLS and EAP-MS-CHAPV2 that can perform computer authentication. Moreover, PEAP does not require the deployment of certificates to wireless clients. Only the authentication server needs to be assigned a certificate.

PEAP offers strong protection against the deployment of unauthorized access, because the client verifies the RADIUS server's identity before proceeding ahead with further authentication or connectivity. An attacker is unable to decrypt the authentication messages protected by PEAP. In addition, PEAP offers highly secure keys that are used to encrypt the data communications between the clients and Authentication Server. New encryption keys are derived for each connection and are shared with authorized Access Points accepting the connection.

3. PROPOSED PROTOCOL: E-PEAP

As with all EAP mechanisms, the initial authentication phase is unencrypted and not protected in PEAP. PEAP specifies an option of hiding a user's name known as identity privacy or anonymous identity in the Phase 1. But this feature is optional. Users may forget to use an anonymous identity or don't like using different identities as anonymous and real in the practice. This behavior makes the network vulnerable and potentially vulnerable in Phase 1. To protect privacy of user identity, in this paper, we propose a new protocol called Enhanced PEAP (E-PEAP). That relies on PEAP

Protocol to prevent finding user identities by reading unencrypted authentication exchanges which the client hasn't used anonymous identity feature of PEAP. We also use Merkle's Puzzle scheme [17] to provide dynamic key distribution. In what follows, we explain the details of our solution.

3.1. Frame Architecture

The RFC of PEAP is a rather short document and defines only four types of messages that can be sent: Request, Response, Success, and Failure. Both request and response messages have a type field, which is used to identify the request or response. The RFC does not specify how these messages should be passed around as EAP is not a LAN protocol at all. In order to get EAP messages passed around the network, they have to be encapsulated during their journey. IEEE 802.1x defines a protocol called EAP over LAN (EAPOL) which is used to get EAP messages from the supplicant to the authenticator directly at Layer 2 [1].

In our protocol, we created a new data field to carry random, numeric variable called "S Value" which is referred to the Merkle's Puzzle [17] from authentication server to the client. S value is a numeric data which refers an encryption key. This field has 2 bytes. So, the decimal number can be in the range of 1-65535.

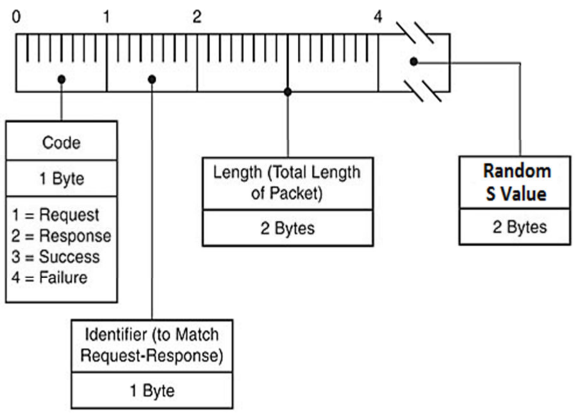


Figure 1. Optimized EAP Frame Format

3.2. Protocol E-PEAP

The proposed protocol E-PEAP encrypts Phase 1 identity with a security way of dynamic key distribution. To achieve this, we defined 10,000 S values which refer a unique encryption key for each values in both two sides of client and authentication server. When Client sends EAPOL Start message to the AS, AS replies with an EAP-Request and a 16 bits S value randomly chosen from the S database in it and waits for the EAP-Response in 3 seconds. If there is no answer in 3 seconds AS sends a new S value because an eavesdropper can be trying to calculate the encryption key. Client gets the EAP-Request and refreshes S value. It has the same S database which shows S values and their encryption keys. Client finds the right encryption key which matches the S value, uses it to encrypt identity with a specific XOR-5 Encryption Algorithm and sends encrypted identity to the AS. AS knows S value and its encryption key which it has sent in previous packet. It decrypts incoming encrypted identity with the same XOR-5 Encryption using the negotiated encryption key. If Phase 1 identity is corrected, the relation between S value and encryption key is broken and a new S value-encryption key chain is created. This provides strong semantic security. For example, if server sends S value of 50871, client uses relevant encryption key which is 10894 and Phase 1 authentication is passed successfully, the relation between 50871 and 10894 breaks and databases are updated with new key relations. For example; S value 10894 has 96498 as encryption key anymore. This development prevents the attacker from inferring the plaintext of encrypted messages if it knows plaintext.ciphertext pairs encrypted with the same key. It provides semantic security. Routine packet flow proceeds according to the PEAP.

PEAP is a tunneled method. The main benefit of a tunnel is that it provides identity privacy. Using tunnels, the tunnel methods can hide the user's identity from eavesdropping by hiding the EAP Response Identity message in the secure tunnel. But it works only for Phase 2. There is a weakness depends on users' manners for the Phase 1. In our designed Phase 1 identity protection method, user's identity is always getting encrypted even if the user uses his real identity or forgets to add anonymous identity field to the configuration file before authentication.

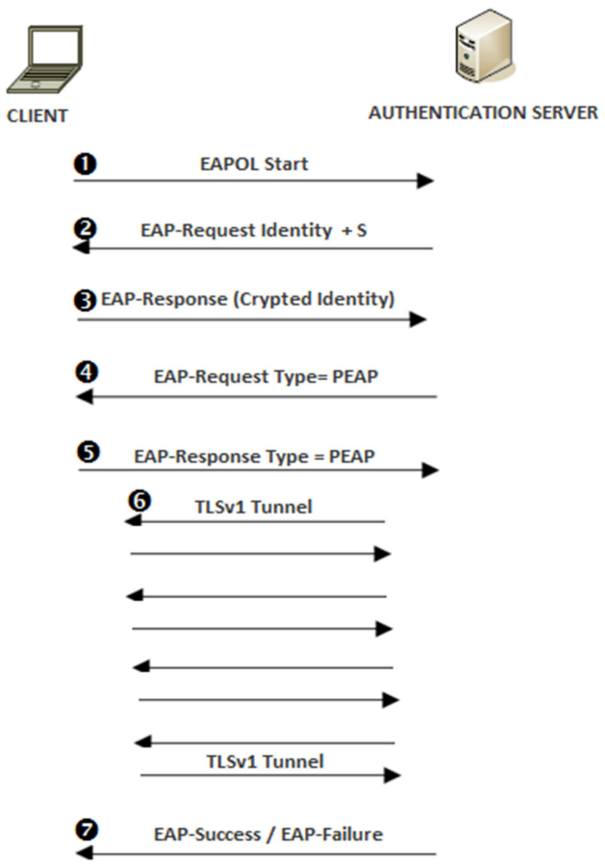

CLIENT
AUTHENTICATION SERVER
1 EAPOL Start
2 EAP-Request Identity + S
3 EAP-Response (Crypted Identity)
4 EAP-Request Type= PEAP
5 EAP-Response Type = PEAP
6 TLSv1 Tunnel
TLSv1 Tunnel
7 EAP-Success / EAP-Failure

Figure 2. EAP Integration

3.3. Implementation

This section details the implementation of our protocol, including the authentication server, authenticator (access point) and wireless client.

The authentication server is realized on Linux 2.6.20-21. Hostapd (version 0.7.3)[18] is running on the authenticator to support and manage the wireless connections with its wireless clients.

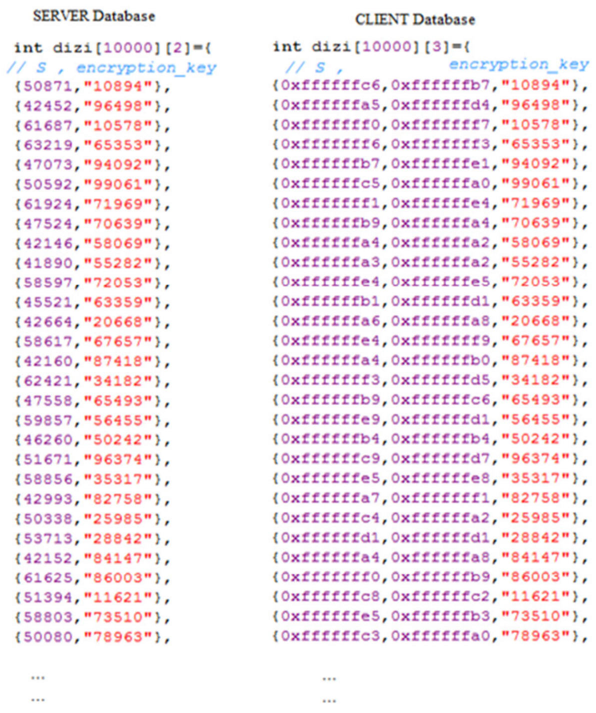
The hostapd program is configured as a pass-through authenticator (wireless AP) and the IP address of the authentication server can be specified in the configuration file, as shown in the following:

```
hostapd.conf :
interface=wlan1
driver=nl80211
ssid=EnhancedPEAP
ieee8021x=1
channel=6
hw_mode=g
wpa=3
wpa_key_mgmt=WPA-EAP
wpa_pairwise=TKIP CCMP
rsn_pairwise=CCMP
wpa_group_rekey=3600
wpa_gmk_rekey=3600
auth_algs=1
eap_server=1
auth_server_addr= 127.0.0.1
auth_server_port= 1812
auth_server_shared_secret= deneme
eap_user_file=/home/cagdas/hostapd-
0.7.3/hostapd/hostapd.eap_user
server_cert=/home/cagdas/server-cert0.pem
ca_cert=/home/cagdas/ca-cert0.pem
private_key=/home/cagdas/server-key0.pem
```

PEAP uses only server side certificate shown in the configuration file. Open source wpa_supplicant (version 1.0.0) [19] tool is used to the clients can connect the Authenticator via wireless media. Similar to other hostapd, we also configured wpa_supplicant as shown in the following:

```
wpa_supplicant.conf :
networkssid="EnhancedPEAP "
key_mgmt=WPA-EAP
eap=PEAP
pairwise=TKIP
group=TKIP
phase1="peapver=0"
phase2="MSCHAPV2"
identity="cagdas"    # real_identity, not anonymous
password="kurt"
```

We have created two databases which have 10,000 S values and encryption keys in two sides of wireless client and authentication server. Parts of the databases are shown below:



SERVER Database

```
 int dizi[10000][2]={
// S , encryption_key
 {50871,"10894"},
 {42452,"96498"},
 {61687,"10578"},
 {63219,"65353"},
 {47073,"94092"},
 {50592,"99061"},
 {61924,"71969"},
 {47524,"70639"},
 {42146,"58069"},
 {41890,"55282"},
 {58597,"72053"},
 {45521,"63359"},
 {42664,"20668"},
 {58617,"67657"},
 {42160,"87418"},
 {62421,"34182"},
 {47558,"65493"},
 {59857,"56455"},
 {46260,"50242"},
 {51671,"96374"},
 {58856,"35317"},
 {42993,"82758"},
 {50338,"25985"},
 {53713,"28842"},
 {42152,"84147"},
 {61625,"86003"},
 {51394,"11621"},
 {58803,"73510"},
 {50080,"78963"},

   ...
   ...
```

CLIENT Database

```
int dizi[10000][3]={
  // S ,                encryption_key
{0xffffffc6,0xffffffb7,"10894"},
{0xffffffa5,0xffffffd4,"96498"},
{0xfffffff0,0xfffffff7,"10578"},
{0xfffffff6,0xfffffff3,"65353"},
{0xffffffb7,0xffffffe1,"94092"},
{0xffffffc5,0xffffffa0,"99061"},
{0xfffffff1,0xffffffe4,"71969"},
{0xffffffb9,0xffffffa4,"70639"},
{0xffffffa4,0xffffffa2,"58069"},
{0xffffffa3,0xffffffa2,"55282"},
{0xffffffe4,0xffffffe5,"72053"},
{0xffffffb1,0xffffffd1,"63359"},
{0xffffffa6,0xffffffa8,"20668"},
{0xffffffe4,0xfffffff9,"67657"},
{0xffffffa4,0xffffffb0,"87418"},
{0xfffffff3,0xffffffd5,"34182"},
{0xffffffb9,0xffffffc6,"65493"},
{0xffffffe9,0xffffffd1,"56455"},
{0xffffffb4,0xffffffb4,"50242"},
{0xffffffc9,0xffffffd7,"96374"},
{0xffffffe5,0xffffffe8,"35317"},
{0xffffffa7,0xfffffff1,"82758"},
{0xffffffc4,0xffffffa2,"25985"},
{0xffffffd1,0xffffffd1,"28842"},
{0xffffffa4,0xffffffa8,"84147"},
{0xfffffff0,0xffffffb9,"86003"},
{0xffffffc8,0xffffffc2,"11621"},
{0xffffffe5,0xffffffb3,"73510"},
{0xffffffc3,0xffffffa0,"78963"},

        ...
        ...
```

Figure 3. Encyption Key Databases

For example, if server sends S value of 50871, client encrypts his real identity with 10894 and sends the encrypted identity.

4. EXPERIMENTAL RESULTS AND DISCUSSION

This section discusses how our protocol can protect against the various attacks. The possible attacks on the EAP method include finding user identities by reading unencrypted authentication exchanges, dictionary attacks or using a list of common passwords in order to attempt to gain access by simulating the authentication exchange offline, Man-In-The-Middle attacks in which an attacker mount a rouge access point between the client and the authentication into a trusted network or 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 [21,22].

By running a packet sniffer tool Wireshark [20], we capture the authentication messages exchanged between the service server and the wireless station. After analyzing the captured messages, we can see from the figure 4 and figure 5 that the EAP-Request/Identity and the EAP-Response/Identity messages.

4.1. Identity Privacy

The term identity privacy means hiding the client’s identity (e.g., his username or email address) from eavesdroppers of the authentication process. Recall from the previous section that the EAP message flow starts with the Request-Identity and Response-Identity messages. Because these EAP messages are sent in plaintext, an eavesdropper sniffing the communication in the beginning of the authentication process can easily discover the client’s identity. But in our purposed protocol, the identity is not in plain text format, it is encrypted. Eavesdropper will not be able to learn the user’s identity because it is being encrypted by a dynamic key in every 3 seconds. Even with a stolen identity, the eavesdropper still cannot login into the system without the correct encryption key. Because the authentication server will try to decrypt the encrypted identity with the encryption key negotiated before.

4.2. Man-in-the-Middle Attack

As discussed before, EAP authentication conversation starts with the Request-Identity and Response-Identity messages. Since these messages are sent in plaintext, attack can easily discover supplicant’s identity by eavesdrop the conversation at the beginning of the process or the attacker can modify the messages exchanged between entities. If an attacker eavesdrops the communication channel between Authentication Server and Client and replace the identity value that is in EAP-Response packet, the Man-in-the-Middle attack still fails because the attacker cannot generate the correct encryption key.

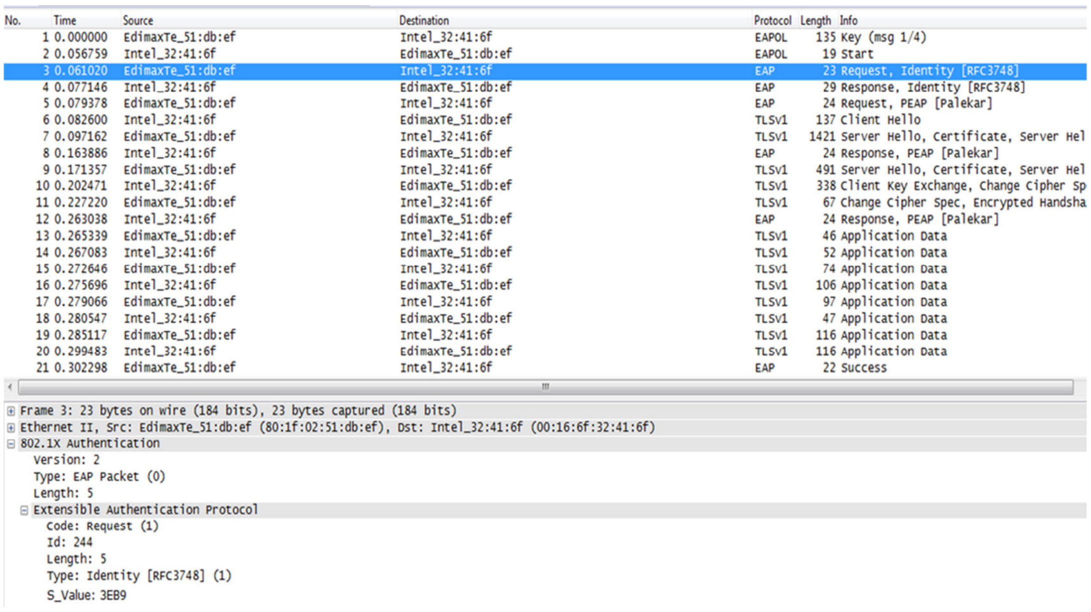
.

4.3. Dictionary Attack

In a dictionary attack, the victim must have some potentially guessable entity (usually a password or passphrase), and the attacker has access to some data derived from the entity in a known way, typically independent of the context. Thus, the attacker can verify guesses. Since the user’s identity is sent in plaintext, they may suffer from the brute-force attack. An attacker may guess real identity by dictionary guessing and comparing the hash values. We made a small experiment to obtain the time cost of compromising an identity. From the experimental results, it is not possible to find correct identity and correct encryption key in 3 seconds.

4.4.Derivation of Strong Keys

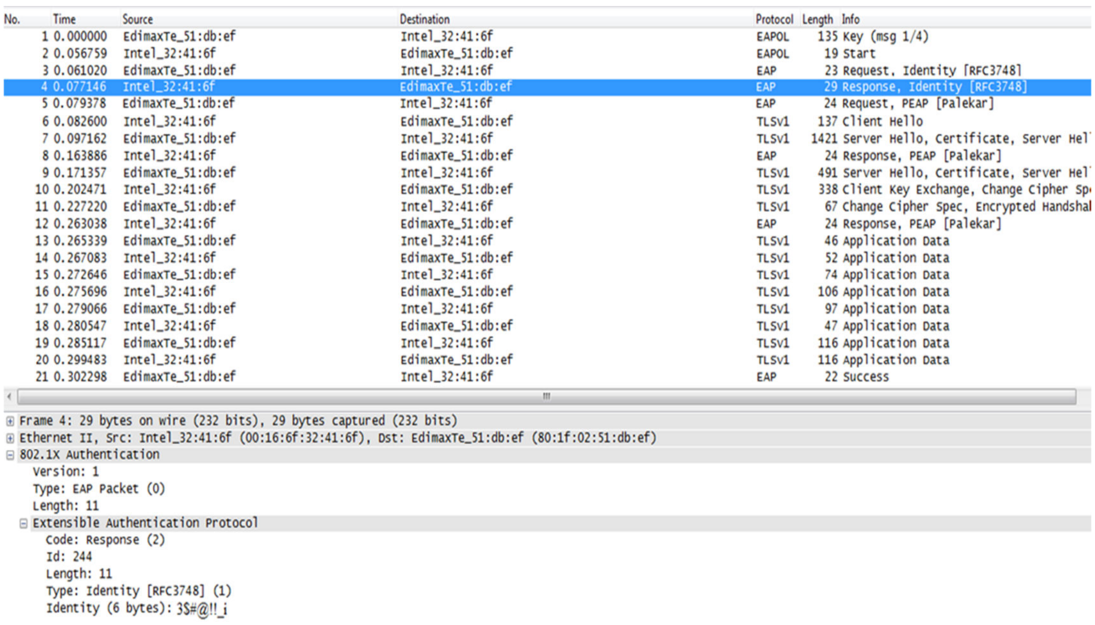
One major weakness of using a static key is that the secret key may eventually be derived from the eavesdropped messages. Any secret is not likely to remain secret forever. Once an attacker discovers the secret key via some attacks, for example dictionary attack, he can decrypt any message that is encrypted with the discovered key. But our purposed scheme generates dynamic encryption key in every authentication request



```
No.  Time      Source             Destination        Protocol Length Info
  1 0.000000   EdimaxTe_51:db:ef  Intel_32:41:6f     EAPOL     135 Key (msg 1/4)
  2 0.056759   Intel_32:41:6f     EdimaxTe_51:db:ef  EAPOL      19 Start
  3 0.061020   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        23 Request, Identity [RFC3748]
  4 0.077146   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        29 Response, Identity [RFC3748]
  5 0.079378   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        24 Request, PEAP [Palekar]
  6 0.082600   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     137 Client Hello
  7 0.097162   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1    1421 Server Hello, Certificate, Server Hel
  8 0.163886   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        24 Response, PEAP [Palekar]
  9 0.171357   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1     491 Server Hello, Certificate, Server Hel
 10 0.202471   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     338 Client Key Exchange, Change Cipher Sp
 11 0.227220   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      67 Change Cipher Spec, Encrypted Handsha
 12 0.263038   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        24 Response, PEAP [Palekar]
 13 0.265339   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      46 Application Data
 14 0.267083   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1      52 Application Data
 15 0.272646   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      74 Application Data
 16 0.275696   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     106 Application Data
 17 0.279066   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      97 Application Data
 18 0.280547   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1      47 Application Data
 19 0.285117   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1     116 Application Data
 20 0.299483   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     116 Application Data
 21 0.302298   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        22 Success

⊞ Frame 3: 23 bytes on wire (184 bits), 23 bytes captured (184 bits)
⊞ Ethernet II, Src: EdimaxTe_51:db:ef (80:1f:02:51:db:ef), Dst: Intel_32:41:6f (00:16:6f:32:41:6f)
⊟ 802.1X Authentication
    Version: 2
    Type: EAP Packet (0)
    Length: 5
  ⊟ Extensible Authentication Protocol
      Code: Request (1)
      Id: 244
      Length: 5
      Type: Identity [RFC3748] (1)
      S_Value: 3EB9
```

Figure 4. EAP-Request Packet along with S_Value



```
No.  Time      Source             Destination        Protocol Length Info
  1 0.000000   EdimaxTe_51:db:ef  Intel_32:41:6f     EAPOL     135 Key (msg 1/4)
  2 0.056759   Intel_32:41:6f     EdimaxTe_51:db:ef  EAPOL      19 Start
  3 0.061020   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        23 Request, Identity [RFC3748]
  4 0.077146   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        29 Response, Identity [RFC3748]
  5 0.079378   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        24 Request, PEAP [Palekar]
  6 0.082600   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     137 Client Hello
  7 0.097162   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1    1421 Server Hello, Certificate, Server Hel
  8 0.163886   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        24 Response, PEAP [Palekar]
  9 0.171357   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1     491 Server Hello, Certificate, Server Hel
 10 0.202471   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     338 Client Key Exchange, Change Cipher Sp
 11 0.227220   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      67 Change Cipher Spec, Encrypted Handshal
 12 0.263038   Intel_32:41:6f     EdimaxTe_51:db:ef  EAP        24 Response, PEAP [Palekar]
 13 0.265339   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      46 Application Data
 14 0.267083   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1      52 Application Data
 15 0.272646   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      74 Application Data
 16 0.275696   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     106 Application Data
 17 0.279066   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1      97 Application Data
 18 0.280547   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1      47 Application Data
 19 0.285117   EdimaxTe_51:db:ef  Intel_32:41:6f     TLSv1     116 Application Data
 20 0.299483   Intel_32:41:6f     EdimaxTe_51:db:ef  TLSv1     116 Application Data
 21 0.302298   EdimaxTe_51:db:ef  Intel_32:41:6f     EAP        22 Success

⊞ Frame 4: 29 bytes on wire (232 bits), 29 bytes captured (232 bits)
⊞ Ethernet II, Src: Intel_32:41:6f (00:16:6f:32:41:6f), Dst: EdimaxTe_51:db:ef (80:1f:02:51:db:ef)
⊟ 802.1X Authentication
    Version: 1
    Type: EAP Packet (0)
    Length: 11
  ⊟ Extensible Authentication Protocol
      Code: Response (2)
      Id: 244
      Length: 11
      Type: Identity [RFC3748] (1)
      Identity (6 bytes): 3$#@!!_i
```

Figure 5. EAP-Response Packet along with Encrypted Identity

5. CONCLUSION

In recent years, differences to IEEE standards for wireless networks added support for authentication algorithms. In this paper, we presented E-PEAP, a new authentication method proposal designed and implemented for wireless networks that support EAP. The proposed method mainly relies on the PEAP protocol. Our new designed protocol mainly provides user identity privacy with the dynamic encryption key scheme which efficiently supports identity privacy and robustness. The proposed protocol also resilient against Man-in-the-Middle and Dictionary attacks. Our protocol requires a new data field in EAP-Request packet messages for initial authentication. Authentication Server puts a random S value to send it to the client. Client uses the relevant encryption key to encrypt his identity. The real world implementation and security analysis show that the proposed protocol E-PEAP is feasible and achieves better security that original PEAP.

REFERENCES

[1] IEEE. (IEEE 802.1X), "Wireless LAN media access local and metropolitan area networks: port-based network access control". Tech. rep., IEEE; June (2001).

[2] Aboba B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H., "Extensible Authentication Protocol (EAP)" , RFC 3748, (2004).

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

[4] Baek, K., Sean W.S., Kotz, D., "A Survey of WPA and 802.11i RSN Authentication Protocols", (2004).

[5] Andersson, H., Josefsson, S., Zorn, G., Simon, D., Palekar, A., "Protected EAP Protocol (PEAP)", IETF personal draft draft-josefsson-pppext-eap-tls-eap-05.txt, (2002).

[6] Dierks, T., Rescorla, E., "The Transport Layer Security(TLS) Protocol", http://tools.ieft.org/html/rfc5246

[7] Wu, T., "A Real-World Analysis of Kerberos Password Security", In Proceedings of the 1999 Internet Society Network and Distributed System Security Symposium, (1999).

[8] Cisco, "Dictionary Attack on Cisco LEAP Tech Note", http://www.cisco.com/warp/public/707/cisco-sn-20030802-leap.shtml, (2003).

[9] Kaufman, C., Perlman, R., Speciner, M., "Network Security, Private Communication in a Public World", Prentice Hall PTR, 2nd Edition, (2002).

[10] M.G.Rahman, H.Imai. "Security in wireless communication." Wireless Personal Communications. Vol.22, No.2, pp.213-228, (2007).

[11] Sheffer, Y., Krawczyk, H., Aboba, B., PIC, "A Pre-IKE Credential Provisioning Protocol",. IETF draft draft-ietf-ipsra-pic-06.txt, (2002).

[12] Funk, P., Blake, S., "EAP Tunneled TLS Authentication Protocol (EAP-TTLS)", RFC 5281, (2002).

[13] Harkins, D., Piper, D., Hoffman, P., "Secure Legacy Authentication (SLA) for IKEv2", IETF personal draft draft-hoffman-sla-00.txt, (2002).

[14] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", IETF ipsec working group draft draft-ietf-ipsec-ikev2-05.txt, (2003).

[15] D.Simon, B. Aboba, R. Hurst, "The EAP-TLS Authentication Protocol, EAP-TLS", http://www.ietf.org/rfc/rfc5216.txt.

[16] Stanley, D., Walker, J., Aboba, B., "Extensible Authentication Protocol Method Requirements for Wireless LANs", RFC 4017, (2005).

[17] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, "Extensible Authentication Protocol" , RFC 3748, (2004).

[18] Merkle, R. C., Secure communications over insecure channels. Communications of the ACM, 21(4):294–299, (1978).

[19] Internet : OPENSSL Tool : http://www.openssl.org/.

[20] Internet : Linux WPA/WPA2/IEEE 802.1X Hostapd http://hostap.epitest.fi/hostapd/.

[21] Internet : Linux WPA/WPA2/IEEE 802.1X Supplicant. http://hostap.epitest.fi/wpa_supplicant/

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

[23] Hwang, H., Jung, G., Sohn, K., Park, S., "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, pp.164-170, (2008).

[24] Internet : Wireshark Packet Sniffer Tool : http://www.wireshark.org/

[25] Asokan, N., Niemi, V., and K. Nyberg, "Man-in-the-Middle in Tunneled Authentication,", Nokia Research Center, Finland, October 24, (2002).

[26] 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, pp.164-170, (2008).