

Strong Mutual Authentication in a User-Friendly Way in EAP-TLS

Carolyn Latze and Ulrich Ultes-Nitsche

University of Fribourg
Department of Informatics
Bd de Prolles 90
1700 Fribourg
Switzerland

Email: {carolin.latze|uun}@unifr.ch

Florian Baumgartner

Swisscom Innovations
Ostermundigenstrasse 93
3006 Bern
Switzerland

Email: florian.baumgartner@swisscom.com

Abstract: EAP-TLS is one of the best authentication schemes in wireless networks. To make it one of the most secure ones, a client has to authenticate itself using certificates, which allow to authenticate client and server mutually. But as certificates are widespread in a business environment but only less used by private users, mutual authentication in EAP-TLS for public hot-spots is not suitable. Therefore several solutions emerged, which on the one hand disclaim EAP completely, or on the other hand establish secure server authenticated EAP-TLS tunnels and use other EAP protocols inside this tunnel to authenticate the client. However, apart from reducing the level of security, these solutions usually do not provide automated login procedures and/or are not suitable for small devices. We propose a way to make EAP-TLS with mutual authentication more comfortable even for private users. To do so, we propose to use Trusted Platform Modules with their integrated certificate infrastructure. This leads to an authentication scheme, which can be used on full computers as well as on embedded devices. Furthermore, it will provide the possibility for automated login and real anonymity support.

1. INTRODUCTION

To authenticate users and machines in a wireless environment (or even in a wired scenario), EAP-TLS [1] is a widely used protocol, which provides a high level of security. This authentication protocol provides even mutual authentication to avoid man-in-the-middle attacks. But to implement this prevention, the client must have a certificate, which might be a valid assumption for employees of big companies, but not for private users. That is why there emerged several other solutions to authenticate a client in a more or less secure way. On the one hand, there are solutions like sticky pages, Wisher [2] and FON [3] and on the other hand, 802.1x based protocols like EAP-TTLS [4], PEAP [5], EAP-SIM [6] and EAP-AKA [7]. All these protocols are less secure than the original EAP-TLS protocol because of a missing mutual authentication and/or provide a less secure client authentication since simple passwords are usually less secure than certificates. Furthermore, some of the solutions do not provide an automated login, which would be needed, for instance, when using small WiFi phones.

We propose a method to use EAP-TLS with mutual authentication in a more comfortable way. As stated above, the

main problem of EAP-TLS is the need for client certificates, which excludes "normal" users. The Trusted Platform Module (TPM) as specified in [8] comes with an international authentication infrastructure in terms of X.509 certificates. The process of obtaining such a certificate for a client is much simpler than retrieving a certificate nowadays. Furthermore, as it is possible to assign different identities to one TPM, our approach will support real anonymity during authentication. As TPMs are shipped with many modern computers [9], it will become an omnipresent (authentication) device in the near future.

In the remainder of this paper, we will first introduce EAP and EAP-TLS, followed by some EAP based alternatives to EAP-TLS. Afterwards, we present commonly used non-802.1x-based authentication mechanisms. Finally, we introduce the Trusted Platform Module and its capabilities as well as our enhancements to EAP-TLS.

2. EAP IN GENERAL

The Extensible Authentication Protocol (EAP) [10] is an authentication framework to support different authentication methods in IEEE 802 networks. An EAP authentication setup consists of two or three components:

1. The *Peer*, also called *Supplicant* is the one, who wants to be authenticated.
2. The *Authenticator* is the one initiating the EAP authentication process.
3. The *Backend Authentication Server*, which may provide authentication services to the *Authenticator*. The *Backend Authentication Server* is meant to provide authentication methods the *Authenticator* is not able to provide.

Usually, every *Authenticator* will provide a fixed set of authentication methods. Furthermore, there might be a "pass-through" mode, which allows to pass authentication methods to an *Backend Authentication Server*. Such a setup easily allows to extend an EAP authentication with more methods without making changes on the *Authenticator*.

In an EAP authentication, it is the *Authenticator* that will send the first message. It has to send a request to the *Peer* usually asking for its identity. The *Peer* then has to send an appropriate reply. This initial identity request may be by-

passed, if the *Peer*'s identity is well known to the *Authenticator*. The authentication continues with sending requests and replies. The protocol is a "lock step" protocol [10], which means that it is not possible to send another request, if there was no valid reply to the previous one. In case, there is no reply, the *Authenticator* has to retransmit its request. After some retransmission without receiving a reply, the *Authenticator* will end the authentication process without informing the *Peer*. In case there is always a valid reply, the conversation continues until the *Authenticator* cannot authenticate the *Peer*, due to an unacceptable response, or until the *Authenticator* successfully authenticates the *Peer*. The first case will result in an EAP Failure message sent by Authenticator and the latter in sending EAP Success.

3. EAP-TLS

EAP-TLS [1] is an EAP integration of the TLS protocol [11]. The authentication between *Peer* and *Authenticator* starts as in every other EAP authentication with an identity request message sent by the *Authenticator*. As soon, as the *Peer* sends its identity the special EAP-TLS authentication takes place starting with EAP-Request/EAP-Type=EAP-TLS (TLS start) sent by the *Authenticator*. In the following exchange of EAP-Request/EAP-Type=EAP-TLS and EAP-Response/EAP-Type=EAP-TLS, the well-known TLS authentication takes place, where client and server authenticate each other either mutually or where the client authenticates only the server. In the end, the *Authenticator* will confirm the *Peer*'s authentication with EAP Success. Last, *Peer* and *Authenticator* have to negotiate the data encryption using ECP [12] and the compression methods using CCP [13].

The need for a certificate on the client side makes EAP-TLS very secure, since an attacker does not only have to steal the client's password, but also its certificate, which is more difficult. However, the way how certificates are issued today makes this very uncomfortable for the user. Usually, the certification process requires many user interactions with the certificate authority (e.g. Verisign [14] or Switch [15] in Switzerland), that then issues the certificate. To overcome this problem, there arose several solutions that aim at providing a similar degree of security without client certificates.

In the following sections, we will introduce two concepts aiming at achieving a similar level of security although only the server may be authenticated using certificates. Both solutions, named EAP-TTLS developed by Funk Software [16] and PEAP developed by Microsoft [17] and Cisco [18] rely on EAP-TLS tunnels with server authentication. The client will then be authenticated using other EAP protocols, e.g. simply using passwords. As on the one hand, simple passwords are rather insecure and on the other hand, secure certificates are too complicated to handle for private users, there arose solutions using personal hardware devices to identify users. This can be done for instance using special-purpose

smart cards, which is again uncomfortable, or using devices the user already possesses - like a mobile phone. The latter approach is used by EAP-SIM [6] and EAP-AKA [7]. We will also give a short introduction to these two protocols in order to discuss alternatives to Trusted Platform Modules that we are going to propose for secure EAP-based authentication.

4. 802.1x / EAP BASED ALTERNATIVES

4.1 EAP-TTLS

EAP-TTLS [4] is an EAP protocol which allows authenticating the client using other authentication mechanisms than certificates. Such other authentication mechanisms include the Password Authentication Protocol (PAP) and the Challenge-Handshake Authentication Protocol (CHAP) as described in [19] or further EAP methods. To secure these authentication methods against known attacks such as "man-in-the-middle", they are tunneled through EAP-TLS.

As EAP-TTLS allows using such simple authentication mechanisms like PAP and CHAP to authenticate the client, there is no need for using client certificates in the EAP-TLS tunnel, although they can be used anyway.

4.2 PEAP

PEAP [5] follows a similar idea like EAP-TTLS. As a first step, there has to be an EAP-TLS tunnel between *Peer* and *Authenticator*. While establishing this tunnel, the client may be authenticated using client certificates or not (there is also the possibility for the client to refuse this kind of authentication). Afterwards there will be a second authentication session using other EAP authentication mechanisms.

Again, similar to EAP-TTLS, PEAP eliminates the need for client certificates but provides security comparable to EAP-TLS.

4.3 EAP-SIM

EAP-TTLS and PEAP as introduced above often use simple password based algorithms to authenticate a client inside the formerly established EAP-TLS tunnel. As passwords are usually less secure than hardware tokens, there arose the idea of using smart cards. However, if users need to have special smart cards to authenticate themselves, this is as uncomfortable as forcing every user to have her own certificate. Therefore, solutions emerged, which propose to use already existing hardware tokens like mobile phones for user authentication. The first approach using these devices is EAP-SIM [6], which we will introduce briefly in this section.

EAP-SIM is an extension to the EAP [10] protocol, allowing authentication based on GSM's Subscriber Identity Module (SIM). Using the SIM replaces the need for client certificates. One may still use EAP-TLS with server certificates to establish a secure tunnel, but the client authentication will then

occur within the secure tunnel. We will now have a closer look at EAP-SIM.

The disadvantage of EAP-SIM is that the SIM was build for use in telecommunication networks, not in a WiFi setup. One of the problems is that SIMs do only support single sessions, which means that one cannot authenticate oneself using EAP-SIM and call a business partner at the same time. That is in particular a problem since EAP [10] requires fast re-authentication, implying that the SIM is needed during the entire connection. Furthermore, there are not many notebooks providing a slot for a SIM card. Hence PCMCIA SIM adapters would be required. Such adapters consume a huge amount of battery power, which is a problem in 802.11 networks. The reason why notebooks usually do not provide such a slot, is that such GSM chipsets are very expensive compared to WiFi chips. The authors of [20] proposed to use the mobile phones instead of using special notebook adapters. This solution surely solves the energy problem and makes the user's life easier because of using already existing hardware. However, it does not solve the other problems such as single sessions.

4.4 EAP-AKA

The Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA) [7] is an EAP extension that uses the Universal Mobile Telecommunications System (UMTS) network (or CDMA2000 in the USA). Again, this protocol uses methods developed for telecommunication networks, not for WiFi environments. As EAP-AKA uses also a SIM or a similar smart card, we have the same limitations as for EAP-SIM, and in addition we have the problem of poor geographical UMTS coverage. Furthermore, similar to EAP-SIM, EAP-AKA does not provide support for real anonymity, since the *Peer* must send its permanent identity when contacting the *Authenticator* for the first time.

5. COMMONLY USED NON 802.1x BASED AUTHENTICATION

802.11 networks do provide basic mechanisms like WPA and WPA2, which restrict access to a WiFi base station to those clients providing a correct passphrase. In a standard WiFi network a wireless base-station will connect to a central radius server storing these user credentials. If a passphrase has been proven correct, the access point establishes an association with the client.

Such a scheme works nicely as long as a central authentication instance is capable of providing a central Radius server, usually limiting such a kind of network authentication to centrally administered networks. Furthermore, since the passphrase is already required for basic network access, such a scheme can work only if all potential clients of the wireless network are known in advance, which usually is only true for enterprise networks.

While in cellular networks roaming agreements exist among the network operators, which allow for an authentication of a foreign client, such an infrastructure is completely missing in 802.11 networks. A customer must always be known to the local network provider.

For that purpose sticky portal pages usually are used, i.e. any client is allowed to associate with the access point, but any Internet access is redirected to a "sticky page" where users enter their credentials or buy a temporary accounts. Once customers have entered correct credentials, Internet access is granted.

However the approach using captive portals has several drawbacks:

- The process of intercepting network access and presenting a sticky page to a customer does not scale very well. The interception of packets requires packet filtering on the process plane which is slow and expensive.
- The process of entering username and password on a web-page is not as easy to automate as one would suspect. Usually the portal pages contain advertising and are changed rather frequently, which makes an automated login procedure rather complicated. However for the use of Voice over IP (VoIP), a reliable automated login procedure is an absolute must, and one of the great benefits of 2G networks is exactly the capability to establish a connection automatically. To leverage VoIP applications over WiFi a similar mechanism is an absolute must.
- While it is in general questionable to use a protocol like HTTP for authentication of a Layer 2 service, some portals even use pages which require a full blown web-browser to display and work correctly. What might work nicely on a notebook screen often is hardly readable on a personal mobile assistant or can even not be realized on embedded devices.
- Since any client must be able to connect at least to the captive portal, security problems might arise. Either by wrong configuration or by denial-of-service like attacks.

Even if there is quite a number of disadvantages, captive portals are for many public WLAN providers the de-facto standard or at least the standard fallback solution.

Other solutions exist, which are not based on captive portals, but use encrypted access points and key distribution schemes to authenticate users. A smart approach to overcome the problem of foreign users and key distribution is Whisher [2]. Whisher is a community service. A Whisher member ideally allows other members to use his access point to connect to the Internet. The key element of Whisher is a small local database on a client computer containing the encrypted keys of Whisher access points in a certain area. A client wanting to connect to the Internet looks up the key for an appropriate Whisher access point within range and establishes a connection. In contrast to other community approaches like FON [3], which depend on captive portals, Whisher can connect to WEP and WPA encrypted access points, therefore providing significantly more security.

However, even if Whisher supports an automated login and does not have to cope with the problems of captive portals, due to the local database, the solution is hardly suited for small devices like personal mobile assistants or WiFi gadgets.

6. EAP-TLS WITH TPM SUPPORT

EAP-TLS with client-side certificates is a very strong authentication method as it provides for instance mutual authentication, which prevents man-in-the-middle attacks [1]. EAP-TTLS [4], PEAP [5] or others do not provide this feature. However, the requirement of using client certificates is very hard to achieve since most private users will not have their own certificates. The TPM as introduced by the Trusted Computing Group (TCG) [21] provides its own certificate infrastructure, which can be used easily with EAP-TLS. We propose using the TPM in the entire certification process and not only for storing and binding client keys as proposed in [22]. Furthermore, we propose using TPM Attestation Keys for a real anonymity support within EAP-TLS.

In the remainder of this section, we will first introduce the Trusted Platform Module (TPM), then explain TPM's certificate infrastructure, and finally show how to integrate TPM's features in EAP-TLS.

6.1 Trusted Platform Module

The Trusted Platform Module (TPM) as specified by the Trusted Computing Group (TCG) [21] is a module, which provides cryptographic functions and secure storage of keys and signatures. The specification does not state that the TPM must be a hardware device attached to the motherboard, but this is the most secure way of doing it. The reason why the specification does not define any deployment details is that the TCG wants to make ubiquity easier. Furthermore the TPM should be a low cost device, otherwise nobody will buy it. This leads to the need for only a minimal subset of cryptographic methods provided by the module and minimal hardware requirements.

The module holds confidential information like cryptographic keys in a protected environment. Furthermore, the TPM provides methods to "seal" sensible data to a special state of the platform. This means, that these data may only be released if the platform is in a specified trustable state.

A very important feature of the TPM is that this module can be uniquely identified. It is equipped with a so called Endorsement Key Pair, which is unique. The private part of this key pair is never released from the TPM which ensures identification.

Furthermore, the module may provide so called "enhanced signatures", which include a proof that they come from a genuine TPM including a hash over the platform's current state. The TCG has registered an "international body" for X.509 certificates [23].

A TPM may manage several identities, which can be used for different purposes. The user might for instance use one identity for her e-banking account and the other one for an authentication as proposed in this paper. Using different identities for different purposes makes the user untrackable. Such a TPM identity must be signed by a certification authority (CA), which means that the CA is the only authority except the TPM's owner that is able to map the identities to a genuine TPM.

6.2 TPM Identity Retrieval

As stated above, the TPM may manage different identities, so called attestation credentials. In this section, we will describe how to retrieve such a TPM identity as specified in [8]

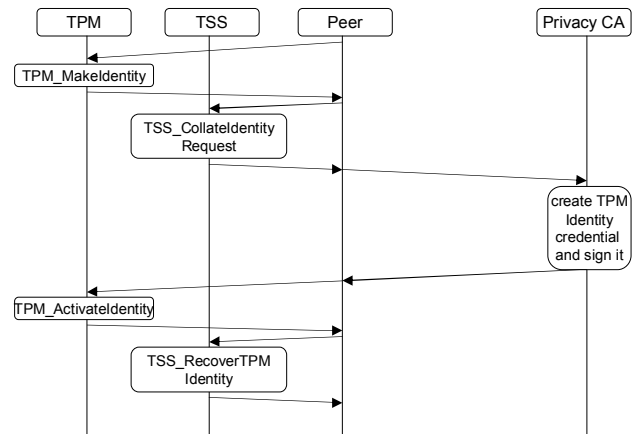


Figure 1: Obtaining a new TPM identity

Figure 1 shows the whole process. First of all, the TPM on the client side has to generate a new identity key using `TPM_MakeIdentity`. Afterwards, the trusted subsystem (TSS) has to collate all information needed to send an identity request to the CA (`TSS_CollateIdentityRequest`). This information includes for instance the endorsement credential, which identifies the TPM uniquely, the conformance credential, which states that the TPM is genuine and the platform credential, which states, that the platform is genuine. The whole request has to be signed by the TPM and then sent to the CA. The CA will then verify the request and the three given credentials, before creating the new identity credential. Afterwards, it opens a secure connection to the client and sends back the newly generated identity credentials. Withing `TPM_ActivateIdentity`, the TPM will first check that the new credential is valid for itself. Furthermore, the TPM has to verify the CA's signature. If all the checks pass successfully, the new identity will be stored securely inside the

TPM. Finally, the TSS has to retrieve a plain text copy of the new credential using `TSS_RecoverTPMIdentity`.

6.3 Integrating the TPM into EAP-TLS

To use the mutual authentication of EAP-TLS in a more comfortable way, we propose to use the TPM identities described above as client certificates. The process described in section 6.2 can be executed mostly without user intervention. The only thing, the user has to do, is to trigger the process and to give some passwords during identity retrieval. This makes it more comfortable for the user to retrieve certificates, which will lead to a better usage of certificates.

Furthermore, a user may request different identities for its TPM, which can only be mapped by the CA. This means, that integrating the TPM's certificate infrastructure does introduce real anonymity support in EAP-TLS.

7. CONCLUSION AND OUTLOOK

We proposed a solution allowing making use of all the features of EAP-TLS, re-establishing full *mutual* authentication. To make it easier for users to obtain a valid certificate, we automate the process using the TPM certificate infrastructure. Furthermore, we realize real anonymity support using TPM identities.

EAP-TLS can be easily implemented in Linux [24] or Windows Systems [25] and will therefore also run on embedded devices. These device will also include (optimized) TPMs [26], which allows to implement our proposed features also on these devices. Furthermore, as EAP-TLS may run completely without user interaction, we can provide an automated login as needed to enable VoIP over WiFi.

To increase the general security of public WLANs, we may also integrated integrity checks on whether or not EAP-TLS certificates are bound to certain platform states of the participating nodes.

REFERENCES

- [1] RFC 2716, B. Aboba and D. Simon, PPP EAP TLS Authentication Protocol, 1999
- [2] Whisher - WiFi Reloaded, <http://www.whisher.com>, April 2007
- [3] FON - Movimiento, <http://www.fon.com>, April 2007
- [4] Internet Draft, P. Funk and S. Blake-Wilson, EAP Tunneled TLS Authentication Protocol (EAP-TTLS), 2004
- [5] Internet Draft, A. Palekar, D. Simon, G. Zorn, J. Salowey, H. Zhou and S. Josefsson, Protected EAP Protocol (PEAP) Version 2, 2003
- [6] RFC 4186, H. Haverinen and J. Salowey, Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM), 2006
- [7] RFC 4187, J. Arkko and H. Haverinen, Extensible Authentication Protocol Method for 3rd Generation Authentication and Key Agreement (EAP-AKA), 2006
- [8] The Trusted Computing Group, Trusted Computing Platform Alliance (TCPA) Main Specification Version 1.1b, 2002
- [9] List of known TPM vendors and implementations, http://www.tonymcfadden.net/tpmvendors_arc.html, June 2007
- [10] RFC 3748, B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowitz, Extensible Authentication Protocol (EAP), 2004
- [11] RFC 2246, T. Dierks and C. Allen, The TLS Protocol, 1999
- [12] RFC 1996, G. Meyer, The PPP Encryption Control Protocol (ECP)
- [13] RFC 1962, D. Rand, The PPP Compression Control Protocol (CCP)
- [14] Verisign Security, <http://www.verisign.com>, June 2007
- [15] SWITCH, <http://www.switch.ch>, June 2007
- [16] Funk Software Inc, now with Juniper, <http://www.juniper.net>, June 2007
- [17] Microsoft, <http://www.microsoft.com>, May 2007
- [18] Cisco Systems, <http://www.cisco.com>, May 2007
- [19] RFC 1334, B. Lloyd and W. Simpson, PPP Authentication Protocols, 1992
- [20] C. Derenale and S. Martini, An EAP-SIM Based Authentication Mechanism to Open Access Networks, *Teletronikk*, 3/4.2006
- [21] Trusted Computing Group, <https://www.trustedcomputinggroup.org/home>
- [22] K.-H. Baek and S.W. Smith, Preventing Theft of Quality of Service on Open Platforms, May 2005
- [23] B. Balacheff, L. Chen, S. Pearson, D. Plaquin, and G. Proudler, trusted computing platforms - tcpa in context, Prentice Hall, 2003
- [24] WPA Supplicant, http://hostap.epitest.fi/wpa_supplicant/, June 2007
- [25] http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/auth_eap.mspx?mfr=true, June 2007
- [26] <http://linuxdevices.com/articles/AT3715099716.html>, June 2007