

Towards a Zero Configuration Authentication Scheme for 802.11 Based Networks

Carolin Latze and Ulrich Ultes-Nitsche
Department for Informatics
University of Fribourg
Bd. de Pérolles 90
1700 Fribourg
Switzerland
Email: {carolin.latze|uun}@unifr.ch

Florian Baumgartner
Swisscom Schweiz AG
Ostermundigenstrasse 93
3006 Bern
Switzerland
Email: florian.baumgartner@swisscom.com

Abstract—Compared to many 802.11 based networks, GSM has an significant advantage. In contrast to 802.11, GSM provides a standardized authentication scheme, which requires no configuration on the end user’s side, but still allows international roaming. GSM does this by using a trusted module within each client: a subscriber identification module.

In contrast to the comparable heavy GSM standard, the early 802.11 standards focused on data transmission within small local area networks, therefore omitting a secure and simple to use authentication mechanism. This caused several different and partly incompatible authentication schemes to evolve, ranging from simple password based login pages to certificate based mutual authentication protocols.

While these protocols can provide state of the art secure authentication they are, from a user’s point of view, almost unacceptable complex, especially if used in an ad-hoc manner outside an corporate environment.

Trusted platform modules, as part of any modern computer, can reduce the user’s overhead to establish a secure 802.11 based connection dramatically by providing secure, potentially anonymous identities. As shown in this paper this approach can be further extended by using an modified TLS handshake, allowing an automated, on-the-fly retrieval of required credentials. Together with the trusted platform modules, this extension can provide a full fledged zero configuration authentication for 802.11 networks.

Keywords: EAP, TPM, 802.1X, TLS, GSM

I. INTRODUCTION

The emergence of 802.11 enabled mobile phones lead to the desire of using Voice over IP (VOIP) over WLAN services instead of the Global System for Mobile Communications (GSM), but everyone who ever tried to configure his 802.11 enabled mobile phone to use it with an public wireless hotspot, knows, that there is still a big difference between GSM and 802.11 based networks. A GSM based mobile phone works without any configuration by the user. He just buys it and uses it, whereas 802.11 based mobile phones have to be configured manually for every public wireless LAN, they should connect to. The reason for the difference lies in the early standards for GSM and 802.11. GSM came with an identity on every device - the subscriber identity module (SIM) - whereas 802.11 did not specify any authentication in its first specification. Meanwhile several 802.1X authentication protocols have been

specified, all having their specific drawbacks: either they do not scale well, are not as secure as promised, cannot be automated, or are too complicated to configure for naïve users. To put it short, common 802.1X authentication methods are far away from being as comfortable as GSM. In order to overcome these problems, in 2007 the authors of this paper proposed to integrate a Trusted Platform Module (TPM) into EAP-TLS [1]. EAP-TLS - the TLS integration into the Extensible Authentication Protocol (EAP) - is one of the most secure 802.1X authentication protocols if used in mutual mode, but the need for X.509 certificates on the client side makes it uncomfortable for the user.

The TPM as specified by the Trusted Computing Group (TCG) [2] comes with a certificate infrastructure. The module may request so called identities, which are X.509 certificates attesting that a TPM is genuine and correct but give no hint about which TPM it is. Using such identities, a TPM may use a different identity for different purposes and the only thing the counterpart knows, is that it speaks with a genuine and correctly implemented TPM. The identity retrieval process does not require user interaction like the “normal” X.509 certification process does. Therefore it is more comfortable for the naïve user to use TPM identities with EAP-TLS than using “normal” X.509 certificates.

This paper presents another extension to this so called EAP-TPM authentication protocol. In the past approach, the user had to request its identity before connecting to an EAP-TPM secured network. The extension presented in this paper allows to connect in an ad-hoc manner and retrieve the needed identity on-the-fly using a modified TLS handshake in order to provide a full fledged zero configuration authentication protocol for 802.11 based networks.

The paper starts with an introduction into GSM in order to outline its advantages for the user, followed by an overview over commonly used 802.1X authentication protocols. It goes on with a section over EAP-TPM and present its proposed extension in detail and ends with a conclusion.

II. GSM NETWORKS

Global System for Mobile Communications (GSM) [3] is one of the world's standards for mobile phones. While offering rather limited transmission speeds only, GSM allows for seamless hand-overs between base stations and international roaming. GSM enforces cryptographic authentication of the mobile end device and encrypted sessions.

Even if GSM is a 20 year old standard and the cryptographic methods used within cannot provide state of the art security, the set of feature's GSM provides are quite convincing, especially since all this is possible with no configuration required on the user's side.

This zero-configuration capability of GSM is achieved by adding a cryptographic device, a subscriber identification module (SIM) to the mobile phone. The SIM card stores a set of algorithms and credentials, which allow to authenticate the mobile end device and to generate keys for the later session encryption. Besides the International Mobile Subscriber Identity (IMSI), the SIM card contains a shared secret K_i and a set of secret algorithms A_n ¹.

A device is only allowed to authenticate to the network if a signed response, calculated with A_3 out of K_i and a random number $RAND$, which has been transmitted by the network, matches the result of a similar calculation in authentication center. $RAND$ is further used with an algorithm A_8 to generate cipher keys for the algorithm A_5 , which encrypts the data transmission during a phone call.

From a user's view the exchange of keys does not require any action, but all which is required is plugging in a SIM card into a phone to activate the device. However, the rich feature set of GSM comes at a certain cost. GSM is definitively not a system, which can be used and operated like 802.11 based networks, by any advanced computer user. GSM has been defined to allow nationwide operators to build up centrally managed wireless networks for mobile voice and data communication, without any respect regarding locally operated wireless networks on a semi-professional level.

III. 802.11 NETWORKS

When 802.11 legacy was first introduced in 1997 [4], it only specified protocols for the data transmission, which lead to the emergence of several different, incompatible authentication protocols. The first popular authentication protocol called Wired Equivalent Privacy (WEP) only came in 1999 [4]. Because of several severe security flaws, WEP became depreciated in 2001. Beside the security problems, in order to get the key, the users have to be known in advance. Last, as the WEP key is the same for every user in the network, it is not useful to authenticate single users.

Another popular 802.11 authentication mechanism are captive portals. Such portals filter the network traffic and display a login page for every new user. Usually, this login page asks the user for its username and password and displays a lot

¹COMP128 an implementation of the A_3 and A_8 algorithm became public in 1998

of advertising. Because of the latter fact, it is not possible to automate this login process as it was possible with WEP. In general the captive portals have the same disadvantage like WEP: all the users have to be known before they connect to the network, but they have the advantage of really authenticating one user. However authenticating a user using username and password requires - depending on the network size - a rather large database on the server side, leading to a rather bad scalability.

WiFi Protected Access (WPA) [5] was the first protocol that tried to overcome the security problems of WEP. In 2004, the new 802.11i standard has been released [6], which included a better version of WPA, called WPA2. As with WEP, using WPA or WPA2 does not allow to authenticate one single user, the algorithm uses simply one key for all the participants of a network. In order to authenticate one single user, 802.11i proposes to use the Extensible Authentication Protocol (EAP) using WPA2 only to keep track of the associations. EAP is as the name suggest an extensible authentication framework that allows to integrate almost every known authentication protocol, ranging from simple password based algorithms to certificate based mutual authentication protocols. EAP protocols based on passwords like EAP-MD5 [7] again have a bad scalability and do not allow a user to connect in an ad-hoc manner. Certificate based mutual authentication protocols like EAP-TLS [8] do not have those limitations. As certificates are transmitted in the beginning of every TLS handshake, there is no need to maintain a user database on the server side, which leads to a good scalability. But in order to provide an ad-hoc authentication, the user has to know the acceptable Certificate Authorities (CAs) in advance and has to have already a X.509 certificate issued by one of those CAs. That means, the ad-hoc authentication is still a problem. Furthermore, when dealing with normal X.509 certificates, for instance issued by VeriSign [9], the user has to go through a rather complicated retrieval process. Usually he has to prove his identity showing a copy of his ID card, which is uncomfortable and usually too complicated for a naïve user

Summarizing, one can say, that 802.1X authentication methods are mostly not really usable for a VOIP setup similar to GSM.

IV. EAP-TPM

In 2007, the authors of this work proposed to use Trusted Platform Modules (TPMs) to retrieve certificates automatically, which will then be used in EAP-TLS [1]. In 2008, the authors present a proof-of-concept implementation and discuss the need for a new type of certificates [10]. The reason for the need for new certificates lies in the certificate retrieval process of the TPMs explained below.

A. The Trusted Platform Module (TPM)

In 2002, the Trusted Computing Group (TCG) introduced a new security module called Trusted Platform Module (TPM) [2]. A TPM is a module that is usually attached to the motherboard, providing cryptographic functions and secure

key storage. Furthermore, the module may be used to collect and report integrity measurements, for instance at boot time to enable a so called trusted boot. Another application for the TPM are trusted network connections. A client that connects to a certain server may request the server's integrity measurements from the server's TPM in order to decide whether to trust this server or not. The server may do the same with the client.

The TPM has also some other nice features for instance the ability to be identified uniquely all over the world. For this purpose the TPM includes a so called endorsement key pair whose private key is never released. Additionally there is the possibility to retrieve new identities for a TPM. That means that a TPM can use different identities for different purposes. The user may use one identity for e-Banking and another one for online-shopping. Those identities are X.509 certificates issued by a special Certificate Authority (CA) called Privacy CA (PCA).

When a TPM is delivered, it comes with some certificates indicating that it is a module operating and deployed according to the standard. In order to request a new identity, the TPM sends those certificates in conjunction with the public part of a newly generated RSA key to the PCA. The PCA then checks all those certificates and issues the new identity. Additionally there is the possibility to include platform integrity measurements into this new certificate in order to be able to verify the trusted state of this platform.

B. Integrating the TPM into TLS

As explained above in order to request a new identity - that is a new X.509 certificate - amongst others the TPM sends the certificates that verify its correctness to the Privacy CA (PCA). Checking those certificates by a PCA is similar to checking the copy of the ID card by a normal CA, but the TPM procedure has the advantage of not relying on human interaction, which means, that the process can be automated. The certificate the PCA issues in this process is a valid X.509 certificate, but the key, that belongs to that certificate is restricted to simple SHA-1 signing. Therefore this so called identity key cannot be used to sign new X.509 certificate or in a TLS handshake. So as to use this certificate anyway in order to reduce the user's overhead to retrieve the TLS prerequisites, the authors of [10] propose to generate new keys and certify that those new keys belong to a TPM with a certain identity. The TCG's software stack (TSS) [11] provides a function called `Tspi_Key_CertifyKey`. This methods returns the key's signature and the `TCPA_CERTIFY_INFO` structure that carries the new key's properties:

```
typedef struct tdTPM_CERTIFY_INFO {
    TPM_STRUCT_VER version;
    TPM_KEY_USAGE keyUsage;
    TPM_KEY_FLAGS keyFlags;
    TPM_AUTH_DATA_USAGE authDataUsage;
    TPM_KEY_PARMS algorithmParms;
    TPM_DIGEST pubkeyDigest;
```

```
    TPM_NONCE data;
    BOOL parentPCRStatus;
    UINT32 PCRInfoSize;
    [size_is(PCRInfoSize)] BYTE* PCRInfo;
} TPM_CERTIFY_INFO;
```

The key's properties include key features like the signature scheme, but it is also possible to include information about the platform's state. If this so called `PCRInfo` is present, the key can only be used if the platform is in the state, the Platform Configuration Register (PCR) specifies. The `TCPA_CERTIFY_INFO` structure will be hashed using SHA-1 and signed by the identity key. The signature is also returned by `Tspi_Key_CertifyKey`. The formal certificate definition as described in [10] looks as follows:

```
Certificate ::= SEQUENCE {
    parentSerialNumber CertificateSerialNr,
    pubKey OCTET STRING,
    tpmCertificate TPMCertificate,
    signatureValue BIT STRING
}
```

with TPMCertificate:

```
TPMCertificate ::= SEQUENCE {
    versionMajor OCTET,
    versionMinor OCTET,
    versionRevMajor OCTET,
    versionRevMinor OCTET,
    keyUsage OCTET STRING,
    keyFlags OCTET STRING,
    authDataUsage OCTET,
    algorithmID OCTET STRING,
    encScheme OCTET STRING,
    sigScheme OCTET STRING,
    parmSize INTEGER,
    parms [0] OCTET STRING OPTIONAL,
    --If not present,parmSize MUST be 0--
    pubkeyDigest OCTET STRING,
    nonce OCTET STRING,
    parentPCRStatus BOOLEAN,
    PCRInfoSize [1] INTEGER OPTIONAL,
    --If not present,parentPCRStatus
    MUST be FALSE --
    PCRInfo [2] OCTET STRING OPTIONAL,
    --If not present,parentPCRStatus
    MUST be FALSE --
}
```

This TPM certificate states that a key belongs to a certain TPM as it is signed with the TPM's identity key. Furthermore, the `TPMCertificate` structure carries all the information included in the `TCPA_CERTIFY_INFO` structure. Together with the identity certificate, this certificate forms a trusted chain, which may then be used in the TLS handshake, but as those certificates are no X.509 certificates and TLS requires X.509 certificates [12], an EAP-TLS based authentication

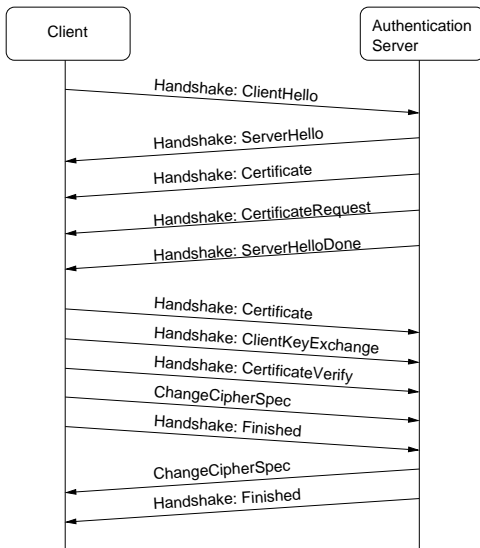


Fig. 1. Timeline Unmodified SSL Handshake

protocol based on those certificates is not really TLS anymore. Therefore, the authors of [10] decided to call this protocol EAP-TPM.

For the sake of completeness, it has to be mentioned that there is another solution to this problem called Subject Key Attestation Evidence (SKAE) [13]. SKAE solves the problem by defining a new X.509 extension, that carries the `TCPA_CERTIFY_INFO` structure returned by `Tspi_Key_CertifyKey`. The whole process is more complicated than the certification process proposed in [10]. It starts with the identity retrieval as usual. After having such an identity, a new key will be created and certified using the identity key. The client generates a certificate request with the certified key and sends it to a SKAE extension aware CA that signs it. The X.509 certificate with the SKAE Extension may now be used for authentication purposes.

The SKAE approach needs modifications on the client and server side and on the CA to handle the new X.509 extension, whereas EAP-TPM only needs a modified client and server. Modifying CAs is a problem as a network operator has no influence on them (at least, he should not have any influence). Therefore only modifying client and server will have a shorter time-to-market.

V. A ZERO CONFIGURATION SCHEME FOR 802.11 NETWORKS

EAP-TPM still requires certificates before starting an authentication sequence. In order to extend EAP-TPM to a real zero configuration authentication scheme, there has to be some kind of automatic certificate retrieval during the TLS handshake.

Figure 1 shows the timeline of a standard SSL/TLS handshake used in the first version of EAP-TPM. The handshake starts with the `ClientHello` message containing the version of the SSL protocol on the client's side, a random number that will be used to generate session keys, a session id in order to

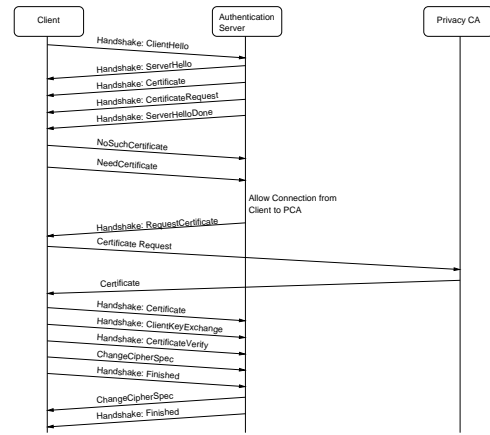


Fig. 2. Timeline Modified SSL Handshake

indicate a session to resume (has to be omitted for new sessions), the supported cipher suites and the supported compression algorithms. The server responds with a `ServerHello` message containing the protocol's version number on the server's side, a random number, the session id, that may be used by the client to resume that session at a later time, the supported cipher suites and the supported compression algorithms. Afterwards, the server sends its own X.509 certificate in the `Certificate` message. As EAP-TPM uses mutual authentication, the next step is the `CertificateRequest` message in order to request the client's certificate. Besides including the type of the requested certificate - the server may choose between RSA and DSS signatures - the server may also specify the acceptable CAs. Usually, this feature is not used. The most popular Linux SSL implementation OpenSSL [14] does not even implement the sending of acceptable CAs till now. The authors of this work propose to use this feature in order to deploy a zero configuration authentication protocol. For this purpose, the `CertificateRequest` message will not only contain acceptable CAs but also acceptable Privacy CAs (PCAs). After having received this message, the client has to check whether he has an identity issued by one of those PCAs or not. In case he does not have such an identity, he may request a new identity on-the-fly and go on with the handshake afterwards.

Figure 2 shows the details of the modified handshake described above. As in a unmodified TLS handshake, it starts with a `ClientHello` and a `ServerHello`. Afterwards, the server sends the `Certificate` message followed by a `CertificateRequest` containing the acceptable PCAs. The structure of `CertificateRequest` does not change:

```

struct {
    ClientCertificateType
        certificate_types<1..2^8-1>;
    DistinguishedName
        certificate_authorities<3..2^16-1>;
}CertificateRequest
  
```

The PCAs have to be specified in the

certificate_authorities section of this message. The ServerHelloDone indicates the end of the handshake. Now, the client has to check whether he has an appropriate identity or not. In case he has one, the handshake goes on as shown in Figure 1, but in case he does not own such an identity, it sends an alert message NoSuchCertificate. The general structure of an alert message in SSL looks as follows:

```
struct {
    AlertLevel level;
    AlertDescription description;
}Alert

where AlertLevel is defined as
enum{warning(1),fatal(2),(255)}AlertLevel;
```

For the NoSuchCertificate alert, an AlertLevel of warning would be appropriate because according to the TLS standard, an alert message with the level fatal causes an immediate shutdown of the session [12].

For the alert description, TLS defines already 24 fixed possibilities:

```
enum {
    close_notify(0),
    unexpected_message(10),
    bad_record_mac(20),
    decryption_failed(21),
    record_overflow(22),
    decompression_failure(30),
    handshake_failure(40),
    no_certificate(41),
    bad_certificate(42),
    unsupported_certificate(43),
    certificate_revoked(44),
    certificate_expired(45),
    certificate_unknown(46),
    illegal_parameter(46),
    unknown_ca(48),
    access_denied(49),
    decode_error(50),
    decrypt_error(51),
    export_restriction(60),
    protocol_version(70),
    insufficient_security(71),
    internal_error(80),
    user_cancelled(90),
    no_renegotiation(100),
    (255)
} AlertDescription;
```

The no_certificate option was only supported in SSLv3 but indicates exactly the problem the authors want to indicate. Therefore, they propose to implement that option also in TLS and use it to indicate that the client misses an appropriate certificate. Afterwards, the client has to send NeedCertificate message in order to determine which

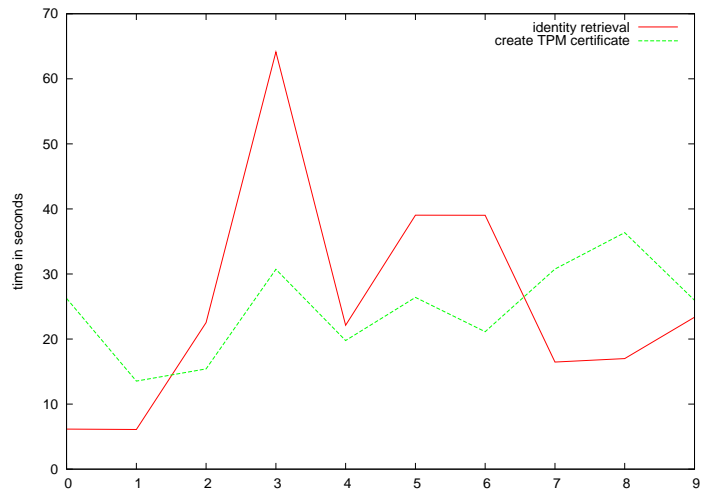


Fig. 3. Time needed to generate new identity certificate

PCA he wants to use:

```
struct {
    opaque privacy_ca<1..2^16-1>;
}NeedCertificate
```

Now, the server sets a rule to allow an unauthenticated and unsecured connection to this certain PCA and sends a RequestCertificate message to the client to tell him, he is allowed to contact the PCA:

```
struct {
    ConnectionAllowed allowed;
    opaque privacy_ca<1..2^16-1>;
}RequestCertificateq
```

with ConnectionAllowed:

```
enum{true(1),false(2)}ConnectionAllowed;
```

The reply from the server also includes the desired Privacy CA (PCA) in order to have some kind of acknowledgment. In case the replied PCA value does not match the requested value, the client knows that something went wrong during the handshake and has to shutdown the authentication session. In case, the values match, the client may request its new identity by sending an identity request to the desired Privacy CA (PCA), which then issues a new identity. Afterwards, the client has to generate a new RSA key pair and certify it using the new identity as described in [10]. Now, the handshake goes on where it stopped. In the original SSL/TLS handshake shown in Figure 1, the final Finished message includes digests over all the handshake messages in order to prove integrity. Of course, all those new handshake messages have to be included into the digest too.

The server may analyze the connection between the client and the PCA. Hence, he knows when the client received its new identity and may close its ports again. After having closed that connection he has to start a timer as the client has to generate a new RSA key pair and a new TPM certificate before going on with the handshake. If this timer exceeds a certain

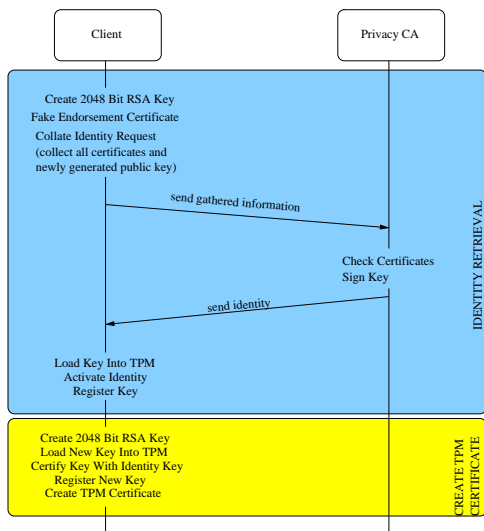


Fig. 4. Certificate Generation Process

threshold, the server has to shutdown the handshake.

Figure 3 shows the time needed to request a new identity certificate from www.privacyca.com and the time needed to create and certify a new key using the identity key.

As shown in Figure 4, the time to request a new identity includes the time needed to collect all the information necessary to send to the Privacy CA (PCA), the time needed to create a new identity key, that will be certified and the communication with the PCA. Furthermore, the new identity has to be activated and registered with the TCG software stack in order to use it. It has to be mentioned, that at the moment no TPM manufacturer delivers already all the certificates needed to request an identity. Therefore, the identity retrieval used in the measurements included generating some fake certificates, which causes higher delays than there will be in the future. That means that the current measurements can be seen as an upper limit.

The second graph of Figure 3 includes the time needed to create a new RSA key pair and the time needed to certify this new key using the identity key as shown in Figure 4. The graph shows that the average time needed to generate a TPM certificate is around 30 seconds. A timer on the server has to respect this time; however, there may be certain reasons why a client is not able to send its certificate within this timeout. First of all, there may be simple technical problems on the client side causing the key generating to fail. Second, it may be a very slow client. In the latter case, the client may start a new TLS handshake after having finished its key generation. As he now has already an appropriate certificate, this handshake will succeed.

VI. CONCLUSION

This paper presents a zero configuration authentication scheme for 802.11 based networks. The authors propose to extend the formerly presented EAP-TPM protocol. This protocol allows to use TPM certificates in a TLS authentication,

which makes it easier for naïve users to use TLS with mutual authentication. While the first version of EAP-TPM still required the user to trigger the certification process before connecting to the secured network, the actual paper proposes to extend EAP-TPM with an extended TLS handshake that allows to request certificates on-the-fly during authentication. Implementing this extension allows the users to connect to EAP-TPM secured hotspots without any prerequisites. Like in GSM, buying a TPM equipped device is sufficient to make use of the network. The TPM in 802.11 based networks will take the role of the SIM in GSM networks.

Even if the authors describe the TPM as being the equivalent to SIM cards in 802.11 networks, using the TPM identities allows to have real anonymity in VOIP, which was never possible in GSM with only one SIM as one SIM can only carry one identity. If somebody wanted to have several identities in a GSM network, he had to use several SIM cards. Using a TPM allows to have several identities on one TPM. The authentication server will not be able to map several identities to one TPM. The only instance that is able to map them is the Privacy CA, which is a trusted third party. That means, 802.11 networks using the proposed EAP-TPM authentication protocol including the extended TLS handshake do provide more comfort than GSM networks.

REFERENCES

- [1] C. Latze, U. Ultes-Nitsche, and F. Baumgartner, *Strong Mutual Authentication in a User-Friendly Way in EAP-TLS*, 15th International Conference on Software, Telecommunications and Computer Networks (SoftCOM 2007), Split - Dubrovnik, Croatia, September 2007
- [2] The Trusted Computing Group, *Trusted Computing Platform Alliance (TCPA) Main Specification Version 1.1b*, 2002
- [3] The European Telecommunications Standardisation Institute *Global System for Mobile Communications - Phase 1*, <http://www.etsi.org>
- [4] *IEEE 802.11 - The Working Group Setting the Standards for Wireless LANs*, <http://grouper.ieee.org/groups/802/11>
- [5] IEEE Computer Society, *IEEE Standard for Information technology - Telecommunication and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, 2007
- [6] IEEE Computer Society, *IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control*, 2004
- [7] L. Blunk and J. Vollbrecht, *PPP Extensible Authentication Protocol (EAP)*, RFC 2284, 1998
- [8] B. Aboba and D. Simon, *PPP EAP TLS Authentication Protocol*
- [9] *VeriSign - Security (SSL Certificates), Intelligent Communications, and Identity Protection*, <http://www.verisign.com>
- [10] C. Latze and U. Ultes-Nitsche, *A Proof-of-Concept Implementation of EAP-TLS with TPM support*, 7th Annual ISSA Conference (ISSA 2008), Johannesburg, South-Africa, July 2008, to be published
- [11] The Trusted Computing Group, *TCG Software Stack (TSS) Specification Version 1.2 Level 1*, 2006
- [12] T. Dierks and C. Allen, *The TLS Protocol Version 1.0*, 1999
- [13] TCG Infrastructure Workgroup *Subject Key Attestation Evidence Extension*, Specification Version 1.0, 16 June 2005, <https://www.trustedcomputinggroup.org/specs/TWG/>
- [14] *Openssl: The Open Source Toolkit for SSL/TLS*, <http://www.openssl.org>
- [15] *Privacy CA*, <http://www.privacyca.com>