Fast Certificate-based Authentication Scheme in Multi-operator maintained Wireless Mesh Networks

Levente Buttyán[a], László Dóra[*,a], Fabio Martinelli[b], Marinella Petrocchi[b]

[a] *Laboratory of Cryptography and Systems Security (CrySyS), Budapest University of Technology and Economics, Hungary*
[b] *Istituto di Informatica e Telematica (IIT), National Research Council (CNR), Pisa, Italy*

Abstract

In this paper, we consider QoS aware mesh networks that are maintained by multiple operators and they cooperate in the provision of networking services to the mesh clients. In order to support mobile users and seamless handover between the access points, the authentication delay has to be reduced. Many proposed fast authentication schemes rely on trust models that are not appropriate in a multi-operator environment. In this paper, we propose two certificate-based authentication schemes such that the authentication is performed locally between the access point and the mesh client. We assume that the access point is always a constrained device, and we propose different mechanisms for mesh clients with different computational performance. For constrained devices, we propose a mechanism where weak keys are used for digital signatures to decrease the latency of the authentication. The authenticity of the weak keys are provided by short-term certificates issued by the owner of the key. The short-term certificate has the digital signature generated by the owner's long-term key. We prove formally that the use of our weak key mechanism on the mesh client side is as secure as the use of some stronger keys. We perform a detailed performance evaluation on our proof-of-concept implementation, and we also compare our solution to the current standard methods.

Key words: Mesh networks, Authentication, EAP, Provable security

1. Introduction

1.1. EU-MESH networks

In the EU-MESH project (www.eu-mesh.eu), we study multi-operator maintained QoS-aware wireless mesh networks for high speed Internet access. In this

*Corresponding author.
Email addresses: buttyan@crysys.hu (Levente Buttyán), dora@crysys.hu (László Dóra), fabio.martinelli@iit.cnr.it (Fabio Martinelli), marinella.petrocchi@iit.cnr.it (Marinella Petrocchi)

paper, we refer to such networks shortly as the EU-MESH network.

The EU-MESH network consists of mesh routers that form a static wireless ad hoc network. Some of the mesh routers function as gateways to the wired Internet, and some of them function as wireless access points (AP) where mobile mesh clients can connect to the network. The sets of gateways and APs can overlap and they do not necessarily cover the entire set of mesh routers.

We envision that the mesh routers are potentially operated by multiple operators, and they cooperate in the provision of networking services to the mesh clients. This cooperation is based on business agreements (similar to roaming agreements in the case of cellular networks). Mesh clients (MC) are mobile computing devices (laptops, PDAs, *etc.*) operated by customers. Customers may be associated with one or more operators by contractual means and have the ability to roam to the rest of the cooperating operators, if necessary.

We assume that MCs connect to APs directly (*i.e.*, MCs are one hop away from the mesh network). MCs use the services provided by the mesh network in order to run various applications. Typically, MCs use the mesh network to access the Internet.

The mesh network supports QoS-based applications and mobility of the MCs. QoS services may have requirements on the length of the interruptions in the communication that they can tolerate. When a MC moves from one AP to another, it has to re-authenticate itself as part of the handover process. Before a successful authentication process, the MC should not be allowed to access the network (otherwise, it can exploit the free short-term access by changing the APs and gaining access without authentication). Thus, the re-authentication delay must be minimized in order to ensure that the interruption caused by the handover remains tolerable for the applications.

In this paper, we are focusing on the MC re-authentication process in EU-MESH networks. Furthermore, we consider the problem of setting up a connection key between the MC and the AP that is used for the continuous enforcement of some access control policy in the network. Although the problem of fast authentication in [1] has been studied before, the proposed schemes rely on trust models that are not appropriate in a multi-operator environment. Our main contribution is that we propose a fast authentication scheme applicable in case of a multi-operator environement.

1.2. Requirements

The main requirements for authentication and access control enforcement in a QoS aware multi-operator maintained mesh network can be categorizes into two groups: One concerning the authentication method and another one which is related to the establishment of the connection keys for the access control enforcement.

Requirements on the authentication method between mesh client and access point:

- *Fast authentication method to support user mobility:* As a main requirement, the authentication method has to support mobility of mesh clients

which may use QoS aware services (e.g. VoIP). Such services may have requirements on the length of the interruptions in the communication that they can tolerate. Thus, the re-authentication delay must be minimized in order to ensure that the interruption caused by the handoff remains tolerable for the applications.

- *Mutual authentication:* During the authentication method, the access point authenticates the mesh client, but the access point also has to provide its authenticity to the mesh client. Furthermore, if it is not the access point that authenticates the mesh client, the mesh client also has to authenticate itself using a third party (typically an authentication server).

- *DoS resistance:* The authentication method should not have any vulnerabilities to DoS attacks. Note, that a successful attack against a central unit (e.g. central authentication server) may lead to a state where no handoff can be completed.

- *Compatibility with standards:* In a multi-operator environment, it is fundamental that the protocols used in the authentication mechanism are standardized or built from standardized elements. Otherwise, a mesh client will not be able to authenticate itself at an access point belonging to another mesh operator.

- *Scalability:* One of the main advantages of mesh networks is the increased coverage. This, however, usually means an increased number of mesh routers, access points, and mesh clients. Therefore, the authentication method must be scalable in terms of the number of access points and mesh clients.

- *No single trusted entity:* In a multi-operator environment, no single trusted entity may exist. Hence, each operator should run its own authentication server(s), but those could cooperate with the servers of other operators based on business agreements.

Requirements on the establishment of connection keys:

- *Connection keys should not reveal long term keys:* The connection keys that the access points obtain during the authentication of the mesh clients should not reveal any long-term authentication keys. This requirement must hold because in the multi-operator environment, the mesh clients may associate to access points operated by foreign operators.

- *Independence of connection keys:* As the neighboring access points may not fully trust each other due to the multi-operator environment, the authentication and the key generation mechanism have to prevent an access point from deriving connection keys that are used at another access point.

- *Freshness:* It must be ensured for both participants that the connection key derived during the authentication process is fresh.

1.3. Contributions

The contributions of this paper are the following: 1) we define the problem of fast authentication in a multi-operator environment, 2) we consider a nonce and a timestamp based protocol, 3) we propose, as we call, the weak key mechanism, which reduces the authentication delay when the mesh client is constrained, 4) we embed and implement our scheme into EAP framework, and 5) we analyze the authentication delay by measuring the delay of our implementation by comparing to current standard authentication methods. Besides that, 6) we offer the formal model, and 7) we apply a formal methodology for security analysis to a specification focussing on the use of the weak key by the mesh client, in order to prove that its use is as secure as the use of some stronger keys (*i.e.*, longer keys that can be revealed with a very low probability). Note that the work behind this paper includes protocol design, formal analysis, real implementation and performance evaluation of the proposed mechanisms. All these contributions result in a complex and realistic evaluation of the considered problem.

This paper can be viewed as an extension of [2]. Here, we redesigned the nonce based authentication protocol in order to provide better DoS resistance. We extended the description of the weak key mechanism, and we also gave a more detailed performance evaluation. The formal methodology is a new contribution, too.

1.4. Overview

The rest of the paper is organized as follows. In Section 2, we give an overview about the state-of-the-art. In Section 3, we propose two certificate based authentication methods and two different certificate sets taking into consideration both powerful and less powerful mesh devices. We evaluate authentication delay of our proof-of-concept implementation by comparing it to standard solutions in different scenarios in Section 4. We investigate the security of a proposed mechanism in Section 5. In Section 6 we evaluate how our scheme satisfy the security and QoS requirements. Finally, we conclude our paper in Section 7.

2. State-of-the-art

In the literature, many authentication and access control enforcement methods have been proposed. We investigate them through a taxonomy presented in [1], where a more detailed description of the state-of-the-art can be found, too. We investigate the proposed solutions categorized by the place of the access control enforcement and by the place of the authentication.

The access control can be enforced by 1) a central entity, 2) the gateways or 3) the APs. Central or gateway level access control enforcement can be assured by the CAPWAP (Control And Provisioning of Wireless Access Points, [3]) standard. The binding to IEEE 802.11 standard is presented in [4]. Herein, the physical and link level functionality of the APs are separated and the link level

functionality is implemented in a central entity. This central entity communicates with a MC through a tunnel established between the central entity and the AP which the MC is associated with. During a handoff, the MC associates with the next AP and runs the 4-way handshake [5] with the central entity. The main problem with this approach is that the whole wireless mesh network is unprotected from unauthorized access, because the AP and the intermediate nodes simply forward the messages to the central entity and the central entity is the first place where the access control can be enforced. Therefore, an attacker is able to decrease the QoS level provided by the network by flooding the mesh network with rogue traffic. Thus, in what follows, we consider only those cases when the APs are responsible for the access control enforcement.

When the access control is enforced by the APs, the authentication can be performed 1) at a remote, central authentication server, 2) at local entities (e.g. mesh routers) playing the role of authentication server, or 3) at APs.

The main benefit of the central authentication server is the easy administration of the subscribers, however, it is a single point of failure. In some proposals (e.g. EAP Extensions for EAP Re-authentication Protocol, also known as HOKEY [6]), the central authentication is only a fallback solution for the case when the responsible mesh router has no data for the MC authentication (typically for the first authentication). The main benefit of this solution that the round-trip time of the authentication messages can be reduced and the scalability can be enhanced, however, in many cases, the physical protection cannot be assured for the authenticator mesh points which usually need to store sensitive data. The authentication can be delegated to or performed by the APs themselves. This is the most scalable solution, therefore, we try to find the solution in that direction.

In IEEE 802.11r standard [7], when a MC first connects to the network, it performs a full IEEE 802.1X authentication with a remote authentication server. The access point AP_0 through which this full authentication is performed will play a special role during the upcoming handoff processes. Before leaving the AP currently associated with, the MC indicates the handoff and the identity of AP_0 to the next AP (through the current AP or directly). The next AP obtains an authentication key K from AP_0. The MC is able to generate K using some public information and the initial authentication key shared with AP_0. The handoff is completed by running a 4-way handshake like protocol with the next AP and deriving connection keys from K. IEEE 802.11r handles only intra-domain handovers, thus, it can support the multi-operator environment only through the EAP framework.

A solution based on ID-based cryptography is proposed in [8], where the authors exploit that the public key pairs can be used both for authentication and for key agreement with an off-line central authority. In this solution, fast handover can not be guaranteed when the handover is performed between two APs belonging to different operators.

Another approach is presented in [9], in which the authors suggest to change the port-based network access control operation of IEEE 802.1X. Instead of restricting the dataflow of MC to authentication messages through the uncon-

trolled port, the current AP allows MCs access to normal data traffic via a dynamically established tunnel between the current and the previous AP. The tunnel remains alive until the authentication is completed. This solution requires to change the current standard.

In [10] and [11], the authors propose a solution where the current AP issues a credential which can be used to certify the MC's authenticity for the next AP. The authors propose to perform a full authentication after the lightweight credential based authorization. This mechanism requires the MCs to trust APs belonging to other operators when issuing credentials.

As we showed here partially and in [1] in more detail, none of the proposed mechanisms can fulfill all the requirements of the authentication process in EU-MESH networks. Therefore, we propose and investigate a new mechanism in the upcoming sections.

3. Our proposal

In this section, we propose two certificate based authentication protocols for EU-MESH networks. First, we describe the architecture of the certificate based authentication protocols. Then, we investigate the speed characteristic of some classical crypto-primitives. After introducing a nonce-based and a timestamp-based authentication method, we define what public key algorithms and key sizes to use during the authentication in order to fulfill the general security requirements while still ensuring a short authentication delay during the handover.

3.1. Architecture

In our certificate based authentication and access control scheme, each operator operates its own certificate authority (CA). Each CA is responsible for issuing certificates for the access points belonging to the operator and issuing certificates to their subscribers. The CA also maintains the certificate revocation list (CRL).

The operators which decide to cooperate (O_1 and O_2) issue cross-certificates of their CAs which means that operator O_1 issues a certificate on the public key of O_2's CA and O_2 issues a certificate on public key of O_1's CA. With the cross-certificates, entities (subscribers or access points) can perform certificate based authentication and key exchange mechanisms even if they belong to different operators.

Each certificate must contain the following items:

- Identity of the issuer
- Time of issuance
- Lifetime (or time of expiration)
- Identity of the owner

- Key usage (encryption or digital signature)
- Public key algorithm
- Owner's public key
- Certificate signature value

The X.509 format [12], as it is a standard format, makes the communication between foreign entities smoother and it is prepared to be extended with additional items. The disadvantage of this format is that it may waste a lot of space.

The certificate signature algorithm consists of two parts: 1) definition of the hash algorithm and 2) definition of the digital signature algorithm. We have no special requirements on the hash algorithm beyond the fundamental security related ones (*e.g.*, collision resistance) because it is usually a very fast crpyto primitive. In contrast to this, the digital signature is a more time consuming one. Herein, the RSA algorithm is a perfect solution, because even if the signing operation needs considerable time, it is not performed in a time critical period. On the other hand, the verification, which is performed during the time critical handover, is very fast.

We suggest to handle the revocation in different ways depending on whether a certificate is issued to a mesh client or an access point. Maintaining CRL suits very well to access points because they have permanent connection to the CA. In contrast to this, the mesh clients, who can be off-line while the private key of an access point becomes compromised, are not able to download the CRL before it connects to the mesh network. Therefore, the CA maintains the CRL and distributes it among the access points and CAs belonging to other operators regularly or when the list changes. The CRL contains the revoked public key pairs of the mesh clients whose certificates are issued for longer time period (months or years). In contrast, the access points' certificates are short-term, valid only for some days. The access points are able to renew their keys and certificates at any time, because they are part of the infrastructure and they are always on-line. The size of the vulnerability window is small due to the limited lifetime, and the damage is also smaller in case of a compromise.

3.2. Design rationale

Here, we investigate the properties of the public key based cryptographic algorithms based on published benchmarks [13] and own measurements. We considered the following key exchange, digital signature and encryption algorithms: Diffie-Hellman (DH), Elliptic Curve DH (ECDH), RSA, DSA, EC-DSA, EC-ElGamal.

Benchmarks showed that the elliptic curve based solutions (ECDH, EC-DSA, and EC-ElGamal) are not beneficial because these algorithms are slower than the classical ones at similar security levels. In the case of DH key exchange algorithm, the computational complexity is as large as the private key operation of RSA, but on both sides. Furthermore, DH does not provide authenticity, and

the key exchange and providing authenticity all together would cause too long delay. Therefore, in what follows, we consider only the RSA and the DSA algorithms.

In the case of RSA, the public key operations (encryption and digital signature verification) are quick operations when the exponent is relatively small (typically 65537), while the private key operations (decryption and digital signature generation) are three orders of magnitude slower. In contrast to this, the digital signature generation with DSA with some precalculation can be performed very quickly, while the verification is three orders of magnitude slower. In what follows, we assume that the DSA precalculations are performed and the generation of digital signature is fast.

The latency of a public-key cryptographic operation on one block mainly depends on the key size of the algorithm and on the performance of the device which performs the algorithm. There is always a trade-off between the speed of the algorithms and the level of the security. Nowadays, e.g. RSA with 512 bit key size is secure for 1 hour, and with 1024 bit for 1 year [14]. In our proposals, we consider only these two key-sizes because the operations with 256 bit long or shorter keys are insecure and with 2048 bit long or longer keys cause intolerable delays in the authentication process.

3.3. Certification based authentication and key transport protocols

3.3.1. Nonce based solution

In [2], we chose the Blake-Wilson and Menezes Provably Secure Key Transport Protocol [15] (BWM), because of two reasons. Firstly, among the considered protocols [16] this protocol has the minimal number of public key based computations as one signature per each participants, that the protocol requires, is a minimum to prove that each one is online and a public key based crypto-primitive to provide a secure key for the upcoming communication. Secondly, this protocol was proven to be secure [15].

However, the BWM protocol as we could adapt it to the mesh environment has a DoS vulnerability. Namely, the AP has to prove its presence first which requires public key cryptographical computation on MC side and therefore, a malicious MC can perform a DoS attack against the AP, easily. Since the key has to be transported by the AP (the motivation is explained in Section 3.4), the roles cannot be changed easily. Therefore, we changed only the order of the verification of online presence.

The procedure of our nonce based authentication mechanism is shown in Figure 1. AP first sends its ID, and a fresh nonce (N_{AP}). MC also generates a nonce and concatenates it to the ID's of the participants (ID_{AP} and ID_{MC}) and the nonce generated by AP. The signature $S_{P_{MC}}(M_1)$ is calculated on these data using MC's private key. One certificate issued by the operator of the MC (OP_{MC}) for the digital signature ($Cert_{OP_{MC}}(S_{MC})$) and one for the encryption ($Cert_{OP_{MC}}(Q_{MC})$) is included in the message, too. On the other side, AP verifies the signature and the certificates and checks whether N_{AP} and ID_{AP} is the nonce and the identity, respectively that it sent in the first message.

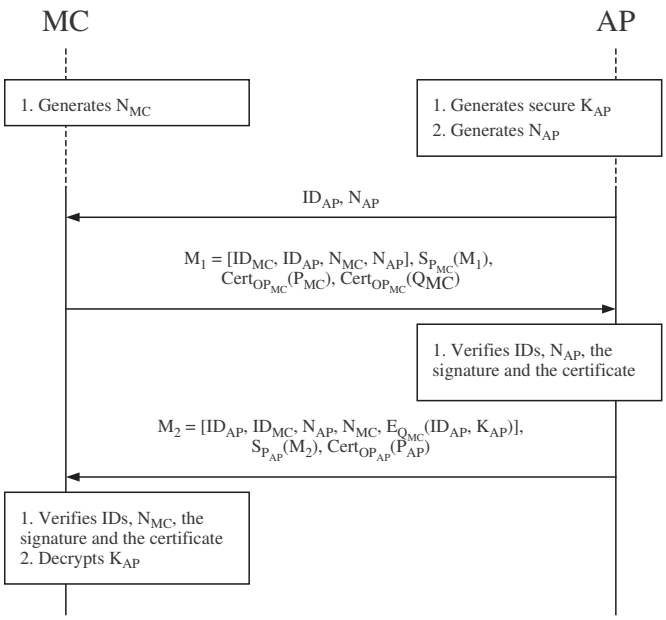


Figure 1: Nonce based authentication

Then, AP encrypts the previously generated key K_{AP} concatenated with its ID using Q_{MC}. AP concatenates the IDs, the nonces and the encrypted key and calculates its digital signature $S_{P_{AP}}(M_2)$. The third message consists of the concatenated data, the digital signature and the certificate issued by the AP's operator (OP_{AP}) for generating digital signatures. Finally, MC has to verify the IDs and nonces match with the ones previously sent and received, and it verifies the certificate, too. After decrypting K_{AP}, MC has to check whether the ID sent in encrypted text matches ID_{AP} and also matches the identity sent in the certificate.

The connection key K_{conn} is calculated with the following method:

$$K_{conn} = Hash(K_{AP}, N_{MC}) \tag{1}$$

where $Hash()$ is a one-way function.

The protocol assures for both participants that they are online. Each participant generates a digital signature over a nonce sent by the other party. The nonce has to be fresh and unpredictable.

Implicit key authenticity is assured, because the key K_{AP} is known only by the AP, who calculated the random bits, and MC, who is the only who can decode the message sent by AP and encrypted with the public part of (Q_{MC}. The Q_{MC} is is used only for encrypting K_{AP}s, and only the MC is able to decrypt with the private key. As no else than the AP and the MC knows the K_{AP}, only they can calculate the K_{conn}.

Key freshness is assured, because K_{conn} is calculated from two elements provided by both participants using one-way function. One-way function assures that AP is not able to choose K_{AP} and MC is not able to choose N_{MC} such that K_{conn} takes a desired value.

The protocol itself does not provide key confirmation, but our implementation will rely on standard IEEE 802.11i [5] which provides key confirmation through the 4-way handshake.

3.3.2. Timestamp based solution

No new requirement has to be met when a timestamp based solution is used, because the verification of the certificates requires loosely synchronized clocks, anyway. Furthermore, it needs fewer random bits and the signed timestamps can be used as a basis of accounting (however this was not mentioned as a requirement before).

The timestamp based scheme, which uses two digital signatures and one encryption, can be seen in Figure 2.

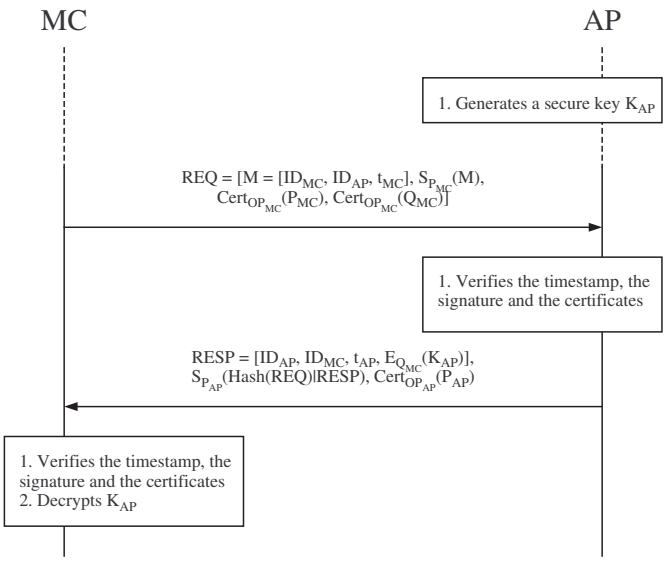


Figure 2: Timestamp based authentication

First, MC sends its timestamp t_{MC} signed with its private key. The first message contains the IDs of the participants and the relevant certificates ($Certs_{MC}$). After AP has checked if the difference between t_{MC} and t_{AP} (i.e. AP's currently generated timestamp) is within the acceptance time window and the IDs are correct, it verifies the signature and the certificates. The acceptance time window is the maximum difference between the timestamps sent in messages (t_{AP} and t_{Mc}) and current time that a participant can accept. AP creates a message containing the IDs, t_{AP} and an encryption of a securely generated key

K using MC's public key. The message sent back to MC contains a signature over these data and the hash value of the message received from MC. The relevant certificates ($Certs_{AP}$) are also included. MC verifies the signature and the certificates, and checks the difference between the clocks. If the IDs agree with the value sent in the first message, MC decrypts key K.

AP's DoS resistance is enhanced by the fact that the MC sends the first authenticated message and the AP has to generate and encrypt K_{AP} and generate digital signature only after MC proves its authenticity and the run of the protocol will end successfully with high probability. Although an attacker can replay eavesdropped messages, it is limited to those messages which are sent to the current AP within the acceptance time window.

This scheme, as it has been presented so far, provides key authenticity and key freshness both for MC and AP, but no key confirmation. The key is controlled by both parties as it is calculated in the following way:

$$K_{conn} = Hash(K_{AP}, t_{MC}) \tag{2}$$

The protocol assures for both participants that they are online. Each participant generates a digital signature over the current time. An attacker is not able to get a valid digital signature over a future timestamp, only if the time synchronization does not work securely.

The key confirmation is provided by the 4-way handshake of standard IEEE 802.11i [5]. Implicit key authenticity is assured, because of the same reasons described at description of the nonce based solution.

Key freshness is also assured, because K_{conn} is calculated from two elements provided by both participants using one-way function. One-way function assures that AP is not able to choose K_{AP} and MC is not able to choose t_{MC} such that K_{conn} takes a desired value. Note that t_{MC} is predictable, in contrast to the nonces, but the calculation of K_{AP} to get a specific K_{conn} is not feasible if the entropy of K_{AP} and K_{conn} high enough. We suggest to use 128 bit long keys and SHA-1 hash function as a one-way function.

3.4. Public key algorithms and key parameters

So far, we did not investigate the parameters of the public key algorithms and the certificates. In both protocols, a MC needs a public key pair for the encryption (Q_{MC}) and another one for the digital signature (P_{MC}). APs only require a public key pair for digital signature (P_{AP}).

The RSA algorithm suits very well to the digital signature of certificates because, as we have already described, even though the signing operation needs considerable time, it is not performed in a time critical period. While the verification is very fast and it is performed during the time critical handover.

Note that a MC has two different public key pairs: 1) one for encryption and 2) one for digital signature. It is insecure to use the same key pair for the two function, because an attacker can exploit one function against the other. Note that it requires two different certificates when the certificates are issued

according to X.509 standard as well as we suggested because of compatibility reasons.

In order to decrease the latency of the verification of two certificates owned by an entity, we define an extension for the X.509 certificates as the standard is flexible enough to add new entries. Considering certificates A and B, when a CA issues the certificates, it generates certificate B in the regular way, but it calculates its hash value h, too. h is added to certificate A as an X.509v3 extension and when the digital signature is calculated it includes the h, too. If a verifier can handle this extension, the verifier calculates the hash value of certificate B and compares it with the appropriate extension of certificate A. If it matches, the verifier verifies certificate A, if not, the certificates are rejected. With this mechanism, the two certificate verification can be reduced to one signature verification and one hash value computation. If a verifier does not support this extension, the verifier can simply ignore it and verify the two certificates separately.

Regarding the digital signatures and encryption, it is beneficial to shift as many computationally intensive operation to the MC as many possible, because of the following reasons:

- Usually MCs which benefit from the seamless handover are more powerful than the APs. It is because one of an important design principle in the case of MCs are to handle media streams which are, therefore, usually equipped with powerful elements. On the other hand, an important design principle is the price in the case of APs, therefore, these can be viewed as constrained devices.

- When the authentication of MCs at an AP are overlapping, the longer lasts the authentication at the AP side, the longer the other MCs have to wait. Furthermore, the more the AP has to calculate, the bigger the chance is that more authentications are overlapping.

- Finally, if the MC has to compute more, it increases the DoS resistance, as an attacker needs more investment to perform a successful DoS attack.

Considering the encryption, currently RSA is a widely known and accepted algorithm which is asymmetric from the time consumption point of view. As we already described, the public key operation of RSA (encryption) is quicker and the private key operation (decryption) is slower operation. This is the reason that we suggest the AP to generate and encrypt the secret key K_{AP} used for connection key.

In order to ensure the confidentiality of the key K_{AP}, we propose to use minimum 1024 bit long keys.

Regarding the parameters of AP's and MC's public key used for digital signature, we differentiate two cases: 1) when the MC is significantly more powerful than the AP and 2) when the difference is less significant. We describe these two cases in the following subsections.

3.4.1. Powerful mesh client

When the MC has more power than the AP (which is the typical case if we consider laptop computers as MCs), the MC can use RSA for digital signature, while the AP generates digital signatures with DSA. In that case all the computationally intensive operations (private key operations with RSA and digital signature verification with DSA) are shifted to the powerful MC, whereas, the lightweight operations are performed by the AP.

The public keys of the MCs, as we defined earlier, are long-term keys. Therefore, we chose 1024 bit long public-private keys. The APs' public key are mid-term as they may change them frequently (e.g. daily). We also chose 1024 bit long keys for mid-term keys.

3.4.2. Constrained mesh client

Note that a less powerful MC is not able to perform all the computing intensive operations. Therefore, we propose another technique to reduce the delay of the whole protocol at the cost of some pre-computation by both participants.

The idea is based on speeding up the digital signature operations by using weak keys. These weak keys have a very short lifetime, such that they surely expire by the time they will be broken.

The weak keys are generated by the participants before the handover happens. In fact, MCs and APs issue certificates themselves. We have to emphasize that these certificates are not self-signed certificates but new elements of certificate chains generated by a MC or an AP. Let us assume that MC wants to issue a short-term certificate. First, it generates a weak public key pair (T_{MC}). Then, it uses its identity as the name of the certificate and determines the expiration date which must be defined carefully, as the weak key can broken quickly. Finally, it supplies the certificate with digital signature using its private key C_{MC}, which is certified by the MC's operator for issuing certificates for weak keys. Therefore, any other entity who knows the CA's public key can validate the authenticity of the weak public key. The same mechanism can be performed at the AP side.

The validity of the certificates are short-term, therefore, maintaining of CRL is not required for implementing this mechanism. Furthermore, in this mechanism, the target AP and the MC which will perform the handover do not need to communicate with each other or to obtain some information about each other, because the certificates are issuer specific. The certificates of the weak keys are signed with RSA so they can be verified very quickly.

We suggest to use 512 bit long keys as short-term keys which seems to be the best tradeoff today between the validity time and the computational overhead. Similarly to the case of a powerful mesh client, the MC uses RSA and AP uses DSA to generate digital signatures. As we described above, in this case all the time consuming operations have to be performed by the MC, but these operations are less time consuming than those with long-term keys.

The time synchronization needs to be performed in a secure way, otherwise an attacker can make a MC or AP to accept an already expired certificate of an

already broken public key pair. However, the investigation of the secure time synchronization is out of scope of this paper.

Even if a secure time synchronization is provided by the system, it cannot be performed before the first association to the network. Note that in that case no QoS aware services run by the MC, therefore, any authentication method is suitable which does not require synchronized clocks.

In Figures 3 and 4, respectively the nonce based and timestamp based authentication scheme is described using weak keys. Here, we emphasize the differences compared to the basic protocols shown in Figures 1 and 2.

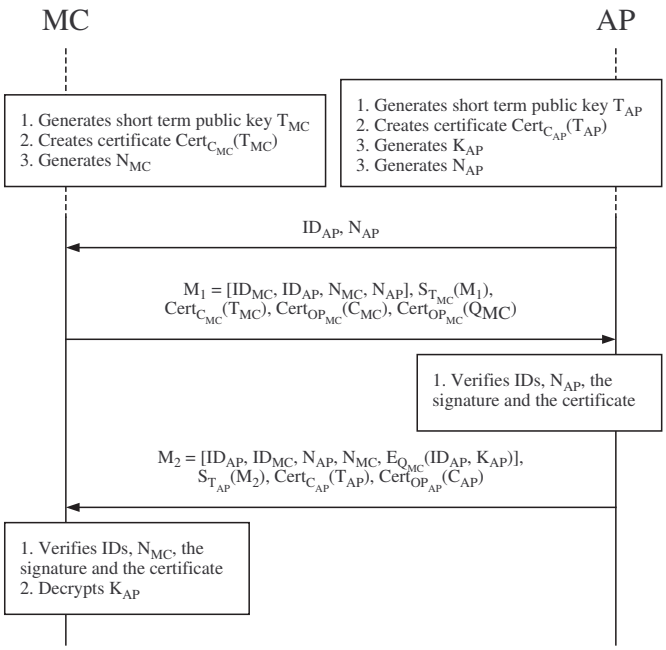


Figure 3: Nonce based authentication

As Figure 3 shows, MC and AP generate weak keys and certificates. MC must check before the handover whether it has a valid certificate. If not, it generates a new one. AP must have a valid temporary key at any time, because the AP does not know when the next MC wants to authenticate. Therefore, it always generates a new certificate before the previous one expires.

The implementation should be designed in such a way that it does not occur that the MC or the AP does not have weak key and belonging certificates available during the handover process. Nevertheless, the participants can use their public-private keys that is used for issuing certificates for weak keys or dedicated public-private keys and belonging certificates should be maintained to handle this case. Obviously, the fast handover can not be assured in this case.

In the authentication phase, the digital signatures are generated using the temporary private keys. Instead of the certificate of the long-term public key used for digital signature (e.g. $Cert_{OP_{MC}}(P_{MC})$), each participant includes two certificates: 1) The short-term certificate for the temporary key used for digital signature (e.g. $Cert_{C_{MC}}(T_{MC})$), and 2) the certificate of the long-term key which is used for issuing short-term certificates (e.g. $Cert_{OP_{MC}}(C_MC)$).

At both sides, the participants have to verify the whole certificate chain, which requires one more certificate verification compared to the case when the mesh client is powerful and no weak key is used.

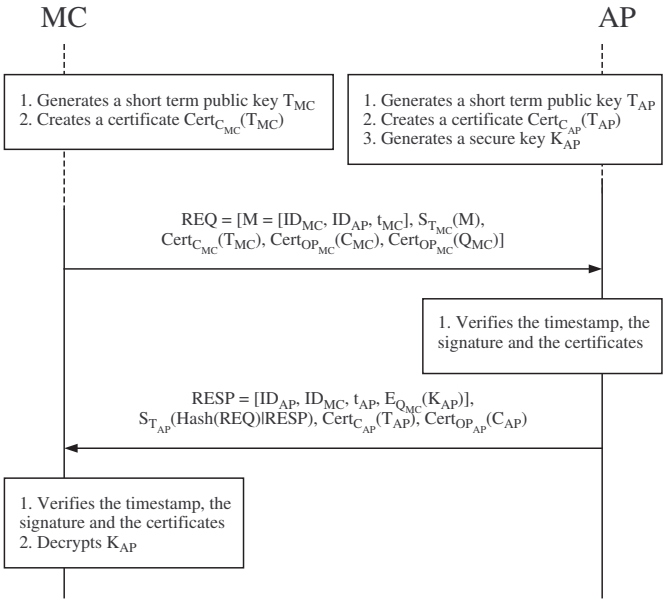


Figure 4: Timestamp based authentication

Note that the usage of the weak key mechanism is optional for each MC. The AP uses the weak key mechanism only if the MC used weak key mechanism. Otherwise the powerful case is supported. It is important because, as one can learn from the performance analysis, the weak key mechanism may increase the authentication delay when the MC is powerful. Consequently, the APs must have public-private keys and belonging certificates for both cases.

In Section 5, we model the timestamp based protocol in a timed process algebra suitable for timing issues ($tCryptoSPA$). We show a possible methodology to carry out an analysis with respect to the use of the weak keys. In particular, we present the analysis with respect to the use of the weak secret key of MC. The method can be opportunely used to analyze also the correct use of the weak key of AP. Note that we do not prove formally that the timestamp based protocol is secure in general. We did it informally, and the formal proof

is assigned as a possible future work. We prove that if MC uses weak keys generated by itself with short-term certificates for digital signature, it is as secure as the timestamp based protocol with long-term keys where the long-term keys can be revealed with a very low probability.

Note that the usage of the weak key mechanism is not limited to the considered authentication scenario. It can be beneficial where the usage of the public key cryptography is advantageous, but the devices are constrained. If weak keys are used among stationary devices, the performance improvement can be more significant, because the certificate verification is performed only once in its lifetime.

3.5. Cross-certificates

As we have already mentioned, our solution is designed for multi-operator environment. In such an environment, the operator of the AP and the MC may not be the same. Therefore, their root CA is different. To handle these situations, the root CAs issue so called cross certificates. In that case, the cross certificates are sent with the other certificates. These enlarge the size of the sent messages and also requires one further certificate verification. The cross certificates are not shown in Figure 1, 2, 3, and 4.

Note that in a regular case, the cross certificate does not change frequently and the participants can learn it, in particular the public key of the other operator's CA can be learnt. Thus, the authentication delay after some bootstrap time become the same as if the MC would be authenticated at an AP at the same operator. However, in the following, we will investigate a cross certificate scenario as a worst case scenario, too.

The verification of the certificate chains must be limited to the operators' CA that has directly signed the certificate of the operator which issued the certificate of the MC. Otherwise, the MC could connect to APs of an operator that has no agreement with the operator of the MC, but both operators have an agreement with a third operator.

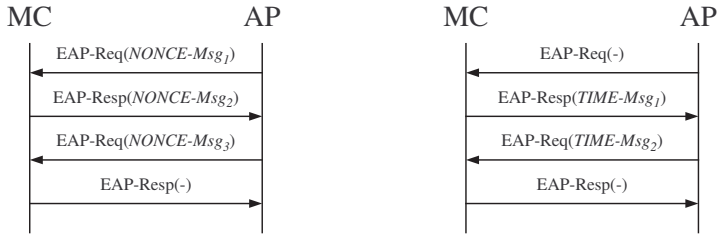
4. Performance analysis

4.1. Implementation

We created a proof-of-concept implementation. We embedded the authentication messages into EAP (Extensible Authentication Protocol) frames [17]. EAP messages are embedded into EAPOL messages in IEEE 802.1X [18] which is referred by IEEE 802.11i and IEEE 802.11r, the current standard solutions for Wi-Fi authentication. K_{conn} defined in Eqs. 1 and 2 is used as a Pairwise Master Key defined in IEEE 802.11i.

The EAP authentication consists of authentication message pairs: EAP Request and EAP Response. In IEEE 802.11i, the EAP Request ($EAP - Req$) always comes from the AP or an authentication server and EAP Response ($EAP - Resp$) comes from the MC as Figure 5 shows. To embed our proposed protocols into the EAP framework, we had to extend our protocols with

the desired number of dummy messages (denoted by '-' in Figure 5). The EAP embedded nonce based authentication protocol can be seen in Figure 5(a), where the fourth EAP message is a dummy message. In the case of the timestamp based protocol, the first and the fourth EAP messages are dummy messages as it is shown in Figure 5(b). Even if the timestamp based protocol consists of two messages, it is initiated by the MC and not by the AP in accordance with EAP framework. This is the reason that we had to add two additional dummy messages to the original protocol.



(a) Nonce based (b) Timestamp based

Figure 5: EAP embedded protocols

The hostapd [19] on the AP side and wpa_supplicant on the MC side gave an extensible framework for our proof-of-concept EAP implementation. We utilized the OpenSSL [20] library for the implementation of crypto-primitives. The source code of the implementation is available from the authors upon request.

To measure the total delay of an authentication run, we caught the events sent by wpa_supplicant when authentication starts and successfully ends. We measured the elapsed time between these two events getting the whole authentication delay. We also measure the time consumption of processing an incoming message and generating the response. This measuring process is coded directly into the hostapd and wpa_supplicant application.

Note that we did not consider the delay of 4-way handshake, because it is independent of the authentication method and its delay has been already investigated in other papers (e.g. [21]).

4.2. Testbed

We investigated the authentication delay in different scenarios. In each case, the AP was a MikroTik Routerboard 133 (175 MHz MIPS32 CPU, 32 MB memory) with OpenWRT (r11349, kernel v2.6.28.6) installed on it. In order to analyze how the MC's performance affects the authentication delay, we used three different MCs: 1) high performance (Dell Inspiron 6000 laptop with 1.86 GHz 32 bit CPU), 2) moderate performance (same laptop with the CPU running at 800 MHz), and 3) low performance (another MikroTik router with same parameters as the AP has).

We compared our proposal to classical, widely used solutions (e.g. EAP-TLS, EAP-TTLS) with authentication servers (AS). For these cases, we installed

hostapd as a stand alone RADIUS [22] server on a PC (with Core2Duo 6400 2.13 GHz CPU, 1 Gb RAM, 32 bit Linux distribution, and kernel v2.6.28). In these scenarios, we connect the AS to the AP with direct link, thus, the roundtrip time between the AS and the MC is minimized.

The type of the wireless card was Atheros AR5414 and Intel 2915 in the case of MikroTik Routerboard and Dell laptop, respectively. The AP and MC communicated through 11g link.

4.3. Authentication delay

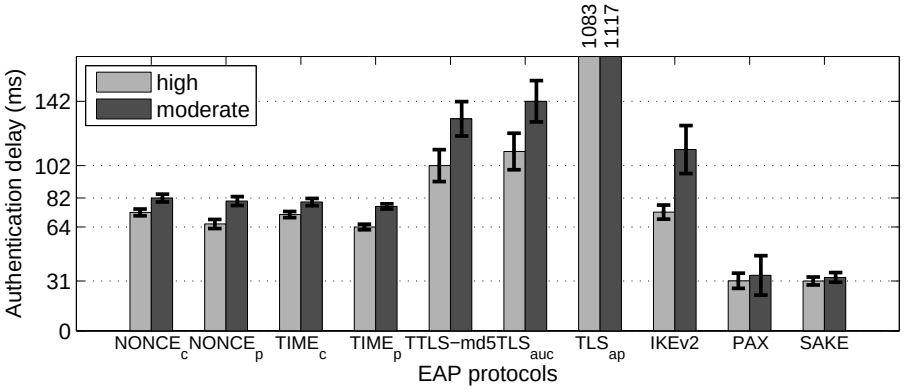
In this paper, we proposed a nonce based (NONCE) and a timestamp (TIME) based authentication scheme with two different certificate sets: one for powerful MCs and another one for constrained MCs (respectively denoted by p and c in the index of the protocol name). We compared these four authentication proposals to 1) EAP-TTLS [23] with EAP-MD5 [17] inside (TTLS-md5), 2) centralized EAP-TLS [24] (TLS_{as}), 3) distributed EAP-TLS (TLS_{ap}), 4) EAP-IKEv2 [25] (IKEv2), 5) EAP-PAX [26] (PAX), and 6) EAP-SAKE [27] (SAKE).

Note that EAP-TLS does not require central subscriber management, because it uses only certificates for the authentication and key exchange. Therefore, the TLS connection establishment can be performed at the APs themselves. This is why we differentiated between the centralized and distributed EAP-TLS. In these methods, we used the same certificates and RSA public-private keys as we did in our proposed methods, with pre-generated 1024 bit Diffie-Hellman key parameters.

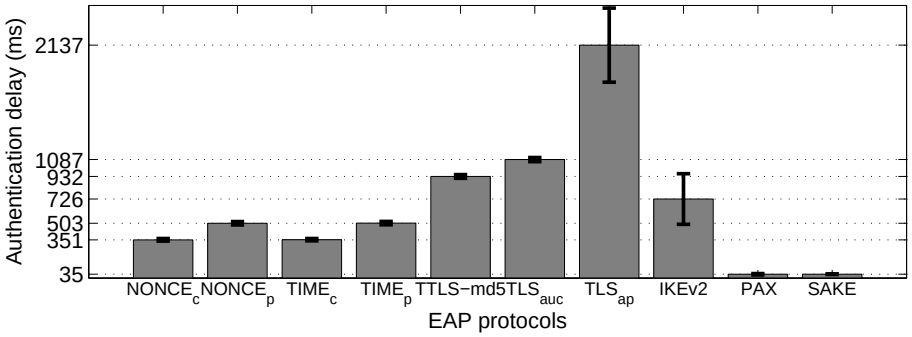
EAP-PAX and EAP-SAKE are shared-secret based solutions relying on symmetric crypto-primitives, only. In these mechanisms, public key cryptography is used if the MC wants to hide its identity. But this is optional, and we consider scenarios where only symmetric cryptography is used.

We compared the ten authentication scenarios with three different MC devices. We measured each case 100 times and calculated the average and the empirical standard deviation of the authentication delays. The results can be seen in Figure 6. We present the results in two subfigures, because the authentication delay in the case of the constrained MC device (shown in Figure 6(b)) is of different order of magnitude compared to the delay using the high and moderate performance MC devices which are shown in Figure 6(a). On the horizontal axes, different protocols in different scenarios can be seen, while on the vertical axes, the authentication delay is shown. In each scenario, the different bars correspond to the measurements made with the different MC devices. The whiskers on the top of the bars refers to the empirical standard deviation of the authentication delays. Note that the authentication delay of EAP-TLS_{ap} was so long compared to the other measurements in Figure 6(a) that we do not show it with complete bar, instead we write explicitly the average value on the top of the reduced bar.

As one can see, each of our mechanisms significantly reduced the authentication delay compared to the centralized public key based authentication methods



(a) High and moderate MC



(b) Constrained MC

Figure 6: Average authentication delay with empirical standard deviation

(TTLS-md5, TLS_{as} and IKEv2) where the AS is a powerful entity in contrast to our mechanism where the AP has limited performance. Furthermore, in the case of the considered centralized methods, the roundtrip time is minimized which, in a real application, may increase with the latency caused by some wireless hops in the mesh network and with the latency caused by the wired network. The authentication delay in the case of TLS_{ap} is even larger, because TLS was not designed for fast connection establishment on constrained devices.

The considered symmetric cryptography based solutions (PAX and SAKE) can complete in around 30-40 ms not taking into consideration the realistic value of the round trip time between the AP and a central authentication server. Note that in the case of high and moderate performance devices, the difference between the symmetric cryptography based solutions and our public key solution is 30-40 ms, but in our certificate based solution there is no further transversal delay. In the case of a constrained device, the delay is considerably higher than in the symmetric cryptography based solutions, but, as we have already described, the centralized solutions have higher vulnerability against DoS attacks, if the

central authentication server can be reached from other places than the mesh network (e.g. Internet).

4.4. Weak key mechanism

As one can see, in Figure 6(b), the weak key mechanism has significant benefit when the MC has low performance. The overall reduction of the authentication delay is 30% on average in the considered scenario. However, as Figure 6(a) shows, the weak key mechanism increases the authentication delay when the MC has high or moderate performance.

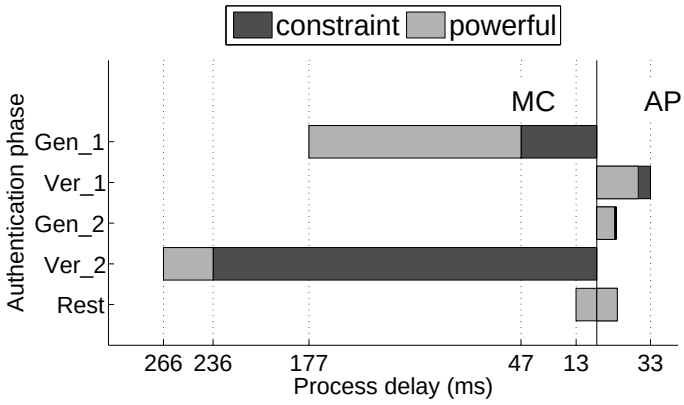
To explain this phenomenon, we measured the delay of the processing time of incoming and outgoing messages separately. These are shown in Figure 7, where we also compare the authentication delay with and without the weak key mechanism. On the right side of the figures, the bars refer to the AP side, while on the left side, the bars refer to the MC side. In these measurements the transportation delay is not counted, but we show the overall transportation delay at the bottom of each figure. These are calculated by getting the difference between the total authentication delay and the sum of all the message processing time. For the sake of simplicity, we shared the transportation delay equally between the AP and the MC.

Note that in Figure 7, only those messages are indicated which are related to the original proposal. Thus, processing time of dummy messages sent, because the EAP framework requires it, are counted in the transportation delay. However, the delay of processing dummy messages is negligible.

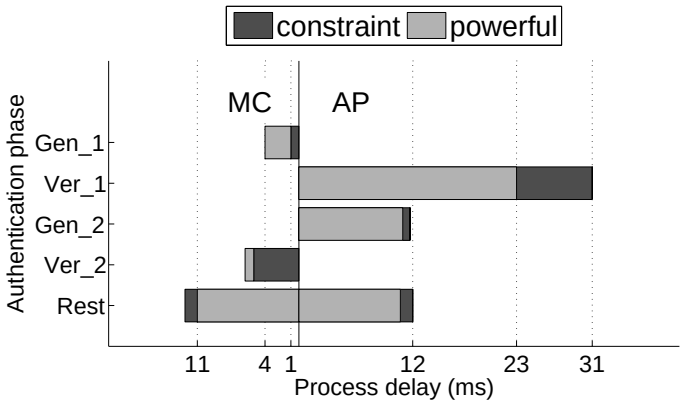
We considered the timestamp based authentication protocol running by constrained and high performance devices in Figure 7(a) and 7(b), respectively. The darker color refers to the case when weak keys are used and the lighter color refers to the case when no weak key is used.

In Figure 7(a), one can see that the weak key mechanism is very beneficial for the MC, but causes some additional delays at the AP side. The weak key mechanism reduced considerably the delay of the generation of the second message. This process includes the generation of MC's digital signature with RSA private key. Regarding the verification of this message on AP side, on the one hand, the verification of the digital signature shortens the delay, but, on the other hand, the AP must verify the certificate issued for the weak key and it causes some additional delay. In the generation of the third message, the AP cannot benefit from the usage of the weak key, because the digital signature generation with DSA can be enhanced by precomputation such that the reduction of the key size does not provide additional significant benefit. Again on the MC side, the usage of the weak key is advantageous, because the verification of the digital signature is reduced. However, the verification of the additional weak key certificate issued by the AP mitigates the positive effect. In the transportation time, there is no significant difference.

In Figure 7(b), one can see the same effects, however, there the MC is powerful, thus, the benefit on the MC side is less significant, and the usage of the weak key mechanism is disadvantageous.



(a) Constrained device



(b) High performance device

Figure 7: Comparison of authentication delay message by message in different scenarios with or without weak key mechanism

Both Figures 7(a) and 7(b) show that the weak key mechanism is not beneficial for the AP at all. The reason is that basically all the computationally intensive cryptographical operation was performed by the MC, thus the main improvements can be achieved on that side, but the additional delays caused by the weak key mechanism burden the AP, too.

To sum up the effect of the weak key mechanism, we can state that it reduces the time consumption of the crypto-primitives, but also causes additional delays: verification (t_{cert}) and transportation of the certificate (t_{trav}). From the reduction of the digital signature generation time (Δt_{gen}) and verification time (Δt_{verif}) both parties benefit, while the certificate verification delay arise at one party, and the transportation delay depends on the link between the two parties.

Taking these into consideration, in general, the usage of the weak key at one

party is beneficial in our proposed authentication scheme if the Eq. 3 holds.

$$t_{cert}^{(B)} + t_{trav} < \Delta t_{gen}^{(A)} + \Delta t_{verif}^{(B)} \tag{3}$$

A in upper index refers to the node that generates the certificate and B refers to the other party. Δt_{op} is the difference between the time consumptions of any operation *op* with a long term key ($t_{op}(S)$) and with the weak key ($t_{op}(w)$) as Eq. 4 shows.

$$\Delta t_{op} = t_{op}(S) - t_{op}(w) \tag{4}$$

4.5. Using cross-certificates

We also investigated the effect of the cross-certificates. In order to show what is the time consumption of cross-certificates, we considered the timestamp based protocol for powerful mesh clients (*i.e.*, no weak keys are used) with moderate performance and compared the case with and the case without cross-certificates. The result can be seen in Figure 8.

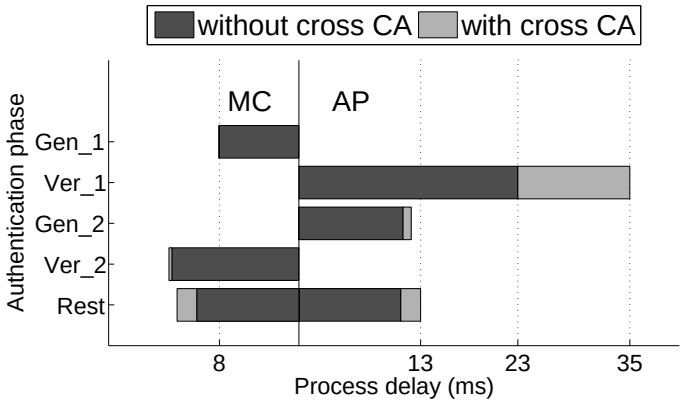


Figure 8: Comparison of authentication delay message by message using moderate performance mesh client with or without cross certificates

To generate the first message needs the same amount of time in both cases, because here, it only needs to add the additional certificate to the first message. The AP needs 12 ms on average to verify the cross-certificate. Also, the generation of the second message has some additional delay, because the constrained AP adds its cross-certificate to the message. On the MC side, the verification is very fast as it is a powerful device. The transportation delay increased by 4 ms on average when a cross-certificate is sent by each party.

5. Security of the weak key mechanism

Formal methods are a popular mean for the analysis of security aspects of computer network protocols. First, the protocol under scrutiny is specified

in a formal language, which often results in a more precise definition of its functioning. Subsequently, the security aspects to be analyzed are specified in a logic. Finally, to decide whether or not certain security properties are fulfilled by the protocol, either automatic tools or manual proof techniques are used to analyze the protocol.

Here, we are going to apply a methodology suitable for modeling and analysing security properties in cryptographic protocols whose correct deployment depends on temporal issue. In particular, we focus on the use of the weak keys in the timestamp based protocol (TIME_c). We aim at formally proving the intuition that, within a certain interval of time, the use of the weak keys is as secure as the use of some stronger keys (*i.e.*, longer keys that are assumed to be disclosed with a very low probability, or, at least, only after that they are no longer significant for the good outcome of the execution of the protocol).

The formal language chosen for the specification of the TIME_c protocol is the $tCryptoSPA$ language (see Section 5.1), belonging to the family of process algebras, that are executable languages for the description of distributed systems.

The foundation of our analysis methodology is the seminal idea of noninterference [28], for investigating the unauthorized information flow in multilevel systems, *e.g.*, from a high level to a lower one. By starting from there, general schemata [29, 30] for the definition of security properties has been formulated, in order to encompass in a uniform way a variety of properties. In Section 5.2, we recall one of those schemata, namely the timed Generalized Non Deducibility on Compositions, tGNDC for short, that basically compares, within a timed framework, what it is expected to be the correct behaviour of a system with a modified behaviour due to the fact that the system is not running in isolation, but it is running together with a malicious process, the so called intruder, trying to interfere with the normal execution of the system. If the two behaviors appear to be the same, then it means that the intruder has not sufficient means to significantly interfere with the honest system and that the investigated property is guaranteed.

Sections 5.3 and 5.4 show 1) how we specify the TIME_c protocol in $tCryptoSPA$ and 2) how we analyse the security of the weak keys, by comparing that specification (plus the specification of the malicious process) with the model representing the correct, expected behavior of the ideal system. The way through which the comparison is carried out is by inspecting all the executions of our specification, and showing that those executions are included in, or at least are equal to, the executions of the ideal specification (except for internal, silent actions). Technically, we show that our specification and the correct one form a so called *weak simulation* (see Section 5.1).

5.1. The tCryptoSPA language

We adopt $tCryptoSPA$ [30, 31] as the modeling language for the TIME_c protocol. $tCryptoSPA$ allows to describe cryptographic protocols where information about the concrete timing of events is necessary. We briefly remind the reader of the syntax, the informal semantics, and some auxiliary notions.

Syntax and informal semantics. The syntax of *tCryptoSPA* is based on a set $\mathcal{C}$ of channels (ranged over by c), a set $\mathcal{M}$ of messages, $Const$ of constants and Var of variables, ranged over by x. The set of *tCryptoSPA* processes is defined as:

$$P ::= \mathbf{0} \mid c(x).P \mid c!m.P \mid \tau.P \mid tick.P \mid P_1 + P_2 \mid P_1||P_2 \mid P\backslash L \mid$$
$$A(m_1, \ldots, m_n) \mid [\langle m_1, \ldots, m_r \rangle \vdash_{rule} x]P_1; P_2 \mid \iota(P)$$

where $m, m_1, \ldots, m_r, m_n$ are messages or variables, x is a variable, and L is a set of channels. Both the operators $c(x).P$ and $[\langle e_1 \ldots e_r \rangle \vdash_{rule} x]P_1; P_2$ bind the variable x in P, P_1. Messages without variables are called *closed* messages.

The informal semantics of the *tCryptoSPA* processes is the following:

- $\mathbf{0}$ is the process that does nothing;
- $c(x).P$ is the process that can receive a message m on channel c and then behaves like P. The received message replaces the variable x;
- $c!m.P$ is the process that can send m on channel c, then behaving like P;
- $\tau.P$ is the process that executes the invisible action τ and then behaves like P;
- $tick.P$ is a process willing to let one time unit pass and then behaving as P;
- $P_1 + P_2$ (*choice*) represents the non deterministic choice between the two processes P_1 and P_2; with respect to *tick* actions, time passes when both P_1 and P_2 are able to perform a *tick* action – and in such a case by performing *tick* a configuration where both the derivatives of the summands can still be chosen is reached – or when only one of the two can perform *tick* – and in such a case the other summand is discarded; moreover, τ prefixed summands have priority over *tick* prefixed summands.
- $P_1||P_2$ (*parallel*) is the parallel composition of processes that can proceed in an asynchronous way but they must synchronize on complementary actions to make a communication, represented by a τ. Both components must agree on performing a *tick* action, and this can be done even if a communication is possible;
- $P\backslash L$ allows only visible actions whose channels are not in L;
- $A(m_1, \ldots, m_n)$. behaves like the respective defining term P where all the variables $x_1, \ldots, x_n$ are replaced by the messages $m_1, \ldots, m_n$;
- $[\langle m_1, \ldots, m_r \rangle \vdash_{rule} x]P_1; P_2$ is the process used to model message handling and cryptography. The process $[\langle m_1, \ldots, m_r \rangle \vdash_{rule} x]P_1; P_2$ tries to deduce an information z from the tuple of messages $\langle m_1, \ldots, m_r \rangle$ through the application of rule $\vdash_{rule}$; if it succeeds then it behaves like $P_1[z/x]$, otherwise like P_2. The set of rules that can be applied is defined through an inference system (*e.g.*, see Figure 9).

- $\iota(P)$ is the idling operator. It allows P to wait indefinitely. At every instant of time, if process P performs an action l, then the whole system proceeds in this state, while dropping the idling operator.

For a detailed description of the operational semantics of a *tCryptoSPA* term the interested reader is referred to [31].

Auxiliary notions. The time model adopted in the language is known as the *fictitious clock* approach of, *e.g.*, [32]. A global clock is supposed to be updated whenever all the processes agree on this, by globally synchronizing on the special action *tick*, representing the passing of a time unit. All the other actions are assumed to take no time.

In order to model message handling and cryptography we use a set of inference rules. Note that *tCryptoSPA* syntax, its semantics and the results obtained are completely parametric with respect to the inference system used.

In Fig. 9, we show a suitable inference system for modeling the TIME_c protocol. Rule (*tuple*) builds a tuple of messages x, y, ...; rule i returns the i-th component of a tuple; rule (*sign*) allows message x to be digitally signed by applying the secret key $sk(y)$ of agent y; rule (*ver*) allows a digital signature $\{x\}_{sk(y)}$ to be verified by applying the public key of signer y, $pk(y)$; rule (*hash*) allows an agent to apply a one-way hash function to message x and obtain digest $h(x)$.

Given an inference system, we can define a *deduction function* $\mathcal{D}$ s.t. if ϕ is a finite set of closed messages, then $\mathcal{D}(\phi)$ is the set of closed messages that can be deduced starting from ϕ by applying instances of the rules in the system.

The agents' activities are described by the actions they can perform. The set *Act* of actions which may be performed by a system is defined as: $Act = \{c(m), c!m, \tau, tick, \mid c \in C, m \in \mathcal{M}, m \text{ closed}\}$. We let l range over $Act\backslash\{tick\}$.

We define *sort(P)* to be the set of all the channels syntactically occurring in the term P.

The expression $P \stackrel{a}{\Rightarrow} P'$ is a shorthand for $P(\stackrel{\tau}{\longrightarrow})^* P_1 \stackrel{a}{\longrightarrow} P_2(\stackrel{\tau}{\longrightarrow})^* P'$ where $(\stackrel{\tau}{\longrightarrow})^*$ denotes a (possibly empty) sequence of transitions labeled τ. The expression $P \Rightarrow P'$ is a shorthand for $P(\stackrel{\tau}{\longrightarrow})^* P'$. Let $\gamma = a_1, \ldots, a_n \in (Act\backslash\{\tau\})^*$ be a sequence of actions; then $P \stackrel{\gamma}{\Rightarrow} P'$ iff there exist $P_1, \ldots, P_{n-1} \in \mathcal{P}$ such that $P \stackrel{a_1}{\Rightarrow} P_1 \stackrel{a_2}{\Rightarrow}, \ldots, P_{n-1} \stackrel{a_n}{\Rightarrow} P'$. Let $\mathbf{0}' = tick.\mathbf{0}'$.

For timed behavioral relations among *tCryptoSPA* processes, in the following we will be mainly interested in *timed trace* inclusions. Let $T(P) = \{\gamma \in (Act\backslash\{\tau\})^* \mid \exists P'.P \stackrel{\gamma}{\Rightarrow} P'\}$ be the set of timed traces associated with process P. The timed trace pre-order $\leq_{ttrace}$ is defined as: $P \leq_{ttrace} Q$ iff $T(P) \subseteq T(Q)$. P and Q are *timed trace equivalent*, denoted by $P =_{ttrace} Q$, if $T(P) = T(Q)$.

We define the concept of weak simulation as usual.

Definition 1. We say that a relation R among processes is a weak simulation, if for every $(P, Q) \in \mathcal{R}$ we have:

- If $P \stackrel{a}{\longrightarrow} P', a \neq \tau$, then there exists Q' s.t. $Q \stackrel{a}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.
- If $P \stackrel{\tau}{\longrightarrow} P'$ then there exists Q' s.t. $Q \Longrightarrow Q'$ and $(P', Q') \in \mathcal{R}$.

Let $\prec$ the union of all weak simulations among processes. Then, we have $\prec \subseteq \leq_{ttrace}$.

5.2. Timed security properties and tGNDC

Timed security protocols are cryptographic procedures where information about the concrete timing of events is necessary, *e.g.*, for the presence of timeouts and timestamps. In a timed setting, the most common security properties, like authentication, secrecy, and integrity can be re-formulated. For example, we may think to

- A timed notion of authentication, called *timed agreement* (see also [33]), according to which an agreement must be reached within a certain deadline, otherwise authentication does not hold.
- A timed notion of secrecy, called *timed secrecy*, according to which a message is secret only within a time interval and after the deadline it can become a public piece of information.
- A timed notion of integrity, called *timed integrity*, which simply requires a correct delivery of messages within a certain amount of time.

Here, we recall a general schema for the definition of such properties [30, 34, 31]. The schema is called *Timed Generalized NDC* (tGNDC for short), and it is a real-time generalization of *Generalized NDC* (GNDC for short) [29].

Informally, the tGNDC schema states that a system specification P satisfies the timed property $tGNDC^{\alpha,C}_{\triangleleft}$ if the behaviour of P, despite the presence of a hostile environment $\mathcal{E}_C$ that can interact with P only through a fixed set of channels C, *appears* to be the same (w.r.t. a timed behavioral relation $\triangleleft$ of observational equivalence) as the behaviour of a modified version $\alpha(P)$ of P that represents the *expected* (correct) behaviour of P. The tGNDC schema thus has the form

$$P \in tGNDC^{\alpha,C}_{\triangleleft} \text{ iff } \forall X \in \mathcal{E}_C : (P \parallel X)\backslash C \triangleleft \alpha(P),$$

where $(P \parallel X)\backslash C$ denotes the parallel composition of processes P and X restricted to communication over channels other than C. X is an arbitrary process in the environment $t\mathcal{E}_C$, the set of all processes that must eventually let time pass, and whose communicating actions are in C. By varying the parameters $\triangleleft$, α and C, the tGNDC schema can be used to define and verify a class of timed security properties—among which timed secrecy, timed integrity, and timed agreement [35, 34, 31, 30].

In the specific context of analyzing (cryptographic) communication protocols, the *static* (initial) knowledge of the hostile environment must be bound to a specific set of messages. This is needed to avoid a hostile intruder that would be too strong and able to corrupt any secret as it would know all (cryptographic) keys, *etc.*. This brings us to the definition of a new environment $t\mathcal{E}^{\phi}_{C}$, based on $t\mathcal{E}_C$, of all processes communicating through channels C and having an initial knowledge of, at most, bound to ϕ. For the analysis of timed safety properties

(such as timed secrecy) it suffices to consider the trace inclusion relation $\leq$ as behavioral relation between the terms of the algebra [29]. Hence we consider the tGNDC instance

$$P \in tGNDC^{\alpha,C}_{\leq_{ttrace}} \text{ iff } \forall X \in t\mathcal{E}^{\phi}_{C} : (P \parallel X)\backslash C \leq_{ttrace} \alpha(P), \tag{5}$$

which was, *e.g.*, used in [34] to analyze timed integrity in wireless authentication protocols for digital streams. Informally, (5) requires the traces of process $(P \parallel X)\backslash C$ to be included in the traces of process $\alpha(P)$, representing the expected behaviour of P when no adversary is present.

In the following, we will model an instance of the TIME_c protocol with $tCryptoSPA$, and we will analyse, under the tGNDC schema, a timed security property representing a combination of timed secrecy and timed agreement.

5.3. Specification of the protocol in $tCryptoSPA$

In this section, and in the following section, we are going to apply a methodology suitable for modeling and analysing security properties in cryptographic protocols whose correct deployment depends on temporal issues. We focus on the use of the weak keys in the TIME_c protocol. The intuition is that the use of the weak keys is as secure as the use of some stronger keys (*i.e.*, longer keys that can be revealed with a very low probability) until the time at which a digital signature is received is within a certain interval from the sending of that signature.

To formally justify that intuition, we focus on the use of MC's weak key sk^w_{MC}. To carry out the analysis, we will present a decorated formal specification of the protocol, by adding special *ad hoc* control actions. Note that a security model and analysis of the protocol with respect to the correct use of sk^w_{AP} is not shown, but it may be carried out in an analogous way.

We first use an intuitive notation usually reported in literature. We consider a set of agents able to send and receive messages. With the following notation,

$$c_j \quad A \rightarrow B \quad : \quad msg$$

we represent the transmission of message msg from a sender A to a receiver B. c_j is the j-th communication channel.

We denote the digital signature of message m by A's secret key sk_A as $\{m\}_{sk_A}$. We use sk^w_A, pk^w_A to represent the pair of *weak* public/private keys of A.

For the sake of readability, we omit to specify both the message to be signed, and the signature on that message. Thus, we use $\{m\}_{sk_A}$ is a shortcut for $m, \{m\}_{sk_A}$.

We represent a digital certificate as the digital signature over a tuple A, pk_A, exp_{pk_A}, where A is the owner of the certificate, pk_A is the public key that is being certified, and exp_{pk_A} is the expiration date of the certificate.

Thus, a regular certificate signed by a certification authority CA is denoted as: $C_A = \{A, pk_A, exp_{pk_A}\}_{sk_{CA}}$, while $C^w_A = \{A, pk^w_A, exp_{pk^w_A}\}_{sk_A}$ is the certificate certifying A's weak public key.

$$
\begin{array}{lllr}
c_1 & MC \to AP & : & \{MC, AP, t_{MC}\}_{sk^w_{MC}}, C_{MC}, C^w_{MC} \\
c_2 & AP \to MC & : & \{MC, AP, t_{AP}, \{K_{AP}\}_{pk_{MC}}, hash(Msg1)\}_{sk^w_{AP}}, C^w_{AP}
\end{array}
$$

We define d_{AP} to be the maximum delay allowed from the moment at which MC sends its message on channel c_1 to the one at which AP receives a message on the same channel. Analogously, d_{MC} is the maximum delay allowed from the moment at which AP sends its reply on c_2 to that at which MC receives a reply on the same channel.

A standard practice in security modeling and analysis is to use some particular *control* actions, that are not in the original specification of a protocol, but they can be used by the honest participants for verification purposes, in order to reveal some information to the outside concerning, *e.g.*, a state reached during a run of the protocol.

Being interested in the secure use of sk^w_{MC}, we will decorate the protocol with a pair of $RUN/COMMIT$ control actions, indicating, respectively, the fact that MC has successfully started the protocol apparently with AP, and that AP has successfully received a valid signed message from (apparently) MC. Also, we insert a special action $PUBLIC$, representing the publication of the secret value sk^w_{MC}. This publication, following the approach of [36], is performed by a particular process called *Timer* that reads from a public channel $c_t \neq \{c_1, c_2\}$ and, after a prefixed timeout value, performs the output of the secret short key on the channel $PUBLIC$, if what it has read on c_t was signed by MC.

We remind the reader that control actions, as well as control processes such as *Timer*, are inserted in the specification only for modeling and analysis purposes, and they do not interfere with the normal execution of the TIME_c protocol. This is the reason why, in modeling and analysing the TIME_c protocol, we will refer to a specification Q, that is the protocol specification P enriched with control actions, and control process $Timer$.

Before introducing the formalization, we list and discuss our modeling assumptions.

We use a process *Clock*, counting the time elapsing, which other processes can ask for the current time value. Thus, *Clock* is the process that either let time pass, or sends the current time on channel *time*.

In modeling the time elapsing, we assume that operating on inference constructs (*e.g.*, decrypting, encrypting, signing, extracting elements from tuples, *etc.*) take no time. Instead, whenever a receiving or sending action is performed on c_1, c_2, at least a *tick* action is performed, to resemble the intuitive notion that time is indeed passing. Special sending and reception over *RUN, COMMIT*, and *PUBLIC* take no time, since they are inserted only for verification purposes, as well as the signaling over the communication channel c_t with process *Timer*.

We assume that, if some process fails in performing a deduction construct, then it becomes idle, *i.e.*, the second derivative of a deduction construct is always $\iota(0)$. This is remarkable, since it resembles the idea that time is indeed

passing, even if a process in Q gets stuck, *e.g.*, for a failure in applying a rule of the inference system. However, for the sake of readability, we omit to explicitly put $\iota(0)$ in the specification.

Finally, we assume that the protocol initiator MC is an urgent process, *i.e.*, it is not *idle*. When MC receives the current time from $Clock$, it simultaneously sends the first message on c_1, and issues the special action RUN. In practice, we consider $t_{MC} = 0$. On the contrary, the responder AP and the process $Timer$ are initially idling.

The $tCryptoSPA$ specification of the protocol is as follows. The bold types highlight the channels over which the control messages are sent/received. They decorate the specification for carrying out the subsequent analysis.

$$Clock(n) \doteq tick.Clock(n+1) + time!n.Clock(n)$$

$MC(sk^w_{MC}) \doteq$	
$time?t_{MC}.$	*Receive current time*
$\mathbf{RUN}!(MC, sk^w_{MC}).$	*control message on RUN*
$[\{MC, AP, t_{MC}\}_{sk^w_{MC}} \quad C_{MC} \quad C^w_{MC} \vdash_{tuple} Msg1]$	*prepare Msg1*
$c_1!Msg1.$	*send Msg1 on c1*
$\mathbf{c_t}!\{MC, AP, t_{MC}\}_{sk^w_{MC}}.$	*send signature to Timer*
$\iota(c_2?y.tick.$	*let time pass*
	until MC receives y on c_2
$time?t'_{MC}.$	*receive current time*
$[y \vdash_1 Sig_{AP}]$	*extract the signature*
$[y \vdash_2 C^w_{AP}]$	*extract C^w_{AP}*
$[C^w_{AP} \quad pk_{AP} \vdash_{ver} z]$	*verify C^w_{AP}*
$[z \vdash_{snd} pk^w_{AP}]$	*extract the weak key pk^w_{AP}*
$[Sig_{AP} \quad pk^w_{AP} \vdash_{ver} sig]$	*verify the signature*
$[sig \vdash_3 t_{AP}]$	*extract timestamp t_{AP}*
$[t'_{MC} - t_{AP} \leq d_{MC}]$	*check the clocks*
$[sig \vdash_5 h]$	*extract digest from Msg2*
$[Msg1 \vdash_{hash} h']$	*compute digest of Msg1*
$[h = h']$	*test equality of digests*
$[omissis]$	*verify IDs*
$[omissis]$	*when all checks succed,*
	go on with key decryption....
$.\iota(0)$	*let time pass*
)	*(closed bracket for first ι....)*

$AP(sk^w_{AP}) \doteq$
$\iota($ *Let time pass until:*
$c_1?y.$ *Receive y over channel c_1*
$time?t_{AP}.$ *Receive current time*
$[y \vdash_1 Sig_{MC}]$ *extract the signature*
$[y \vdash_2 C_{MC}]$ *extract MC's regular certificate*
$[y \vdash_3 C^w_{MC}]$ *extract C^w_{MC}*
$[C_{MC} \quad pk_{CA} \vdash_{ver} z]$ *verify the regular certificate*
$[z \vdash_2 k]$ *extract encryption key pk_{MC}*
$[C^w_{MC} \quad pk_{AP} \vdash_{ver} w]$ *verify C^w_{MC}*
$[w \vdash_2 pk^w_{MC}]$ *extract the weak key pk^w_{MC}*
$[w \vdash_2 exp_{pk^w_{MC}}]$ *extract expiration time $exp_{pk^w_{MC}}$*
$[Sig_{MC} \quad pk^w_{MC} \vdash_{ver} sig]$ *verify the signature*
$[sig \vdash_3 t_{MC}]$ *extract timestamp t_{MC}*
$[t_{AP} - t_{MC} \leq d_{AP}]$ *check the clocks*
$[t_{MC} \geq t_{AP} - exp_{pk^w_{MC}}]$ *check the validity of pk^w_{MC}*
COMMIT$!(AP, pk^w_{MC}).$ *control message on COMMIT*
$[y \vdash_{hash} y']$ *compute digest*
$[K \quad k \vdash_{enc} K_k]$ *encrypt K*
$[MC \quad AP \quad t_{AP} \quad K_k \quad y' \vdash_{tuple} t]$ *prepare first part of message*
$[t \vdash_{sign} t']$ *sign first part of message*
$[t' \quad C^w_{AP} \vdash_{tuple} msg2]$ *prepare message to be sent on c_2*
$c_2!msg2.\iota(0)$ *send message on c_2,*
$)$ *and let time pass*

$Timer \doteq$
$\iota(\mathbf{c_t}?x.$ *Receive on channel c_t*
$[x \quad pk^w_{MC} \vdash_{ver} x']$ *if x is signed with sk^w_{MC}*
$tick^{exp_{pk^w_{MC}}}.$ *let $exp_{pk^w_{MC}}$ time units pass after that:*
PUBLIC!$sk^w_{MC}.\iota(0)$ *publication of MC's short secret key*
$)$

We thus consider:

$$Q(sk^w_{AP}, sk^w_{MC}) =$$
$$(Clock(0)||AP(sk^w_{AP})||MC(sk^w_{MC})||Timer)\backslash\{c_1, c_2, time, c_t\}$$

as the specification representing the TIME_c protocol decorated, for verification purposes, by *control* actions *RUN*, *COMMIT*, and *PUBLIC*, plus *control* processes *Timer*, and *Clock*. Q allows only visible actions whose channels are different from $c_1, c_2, time, c_t$.

Note that we omit to explicitly put some steps in the MC's specification, since we are now interested in the correct reception of the first message by AP.

5.4. An analysis of the TIME$_c$ protocol (with respect to sk^w_{MC})

The TIME$_c$ protocol assumes the use of weak keys for digital signatures. These keys have a very short lifetime, and they should expire by the time they can be revealed. The expiration times $exp_{pk^w_{MC}}$ and $exp_{pk^w_{AP}}$ certified by C^w_{MC} and C^w_{AP} are indeed the flags denoting the maximum period of time after that these weak keys will be revealed.

As announced, we refer to the secure use of sk^w_{MC}. The security analysis of the protocol with respect to the use of sk^w_{AP} may be carried out in an analogous way, provided that a symmetric specification is considered, with a *RUN* signal on AP and the corresponding *COMMIT* signal on MC.

We consider a combination of the timed secrecy property on sk^w_{MC}, plus a timed agreement on a AP's commitment to the received signature from MC.

We assume that C^w_{MC} is issued when MC starts the protocol. In this case, AP should accept a signed message if and only if $t_{MC} \geq t_{AP} - exp_{pk^w_{MC}}$.

In terms of timed secrecy, we want that, whenever sk^w_{MC} becomes public, no more than $t_{AP} - exp_{pk^w_{MC}}$ units of time passed since MC has started the protocol. This condition is specified in the AP's formalization by the inference construct $[t_{MC} \geq t_{AP} - exp_{pk^w_{MC}}]$.

Suppose that AP signals the acceptance of a digital signature by MC as a valid signature by issuing a control action on the special channel *COMMIT*. Thus, the correct, expected behaviour of Q, with respect to the secrecy of sk^w_{MC}, is defined as:

$$\begin{aligned}\alpha^{sk^w_{MC}} = \quad & RUN!(MC, sk^w_{MC}).\\ & (\iota(COMMIT!(AP, pk^w_{MC})).\iota(PUBLIC!sk^w_{MC}).\iota(0)\\ & +\iota(PUBLIC!sk^w_{MC}).\iota(0))\end{aligned}$$

This specification represents a process in which, whenever $RUN!(MC, sk^w_{MC})$ is issued,

- either: the sending on *PUBLIC* is always preceded by the sending on *COMMIT*, *i.e.*, the AP's acceptance of the digital signature with sk^w_{MC} is always before than the publication of sk^w_{MC};
- or: the sending on *PUBLIC* is performed, but the sending on *COMMIT* will be never performed. This describes the cases in which 1) the first message never reaches AP, *e.g.*, because the intruder has dropped that message, and the timeout value passed; or 2) the first message reaches AP, but some deduction constructs fails, *e.g.*, because the intruder has delayed that reception, so that AP becomes idle. On the other side, the timeout is passed, and *Timer* issues on *PUBLIC*.

In terms of $tGNDC$, we ask that:

Definition 2. $Q(sk^w_{MC})$ is $tGNDC^{\alpha^{sk^w_{MC}}}_{\leq_{ttrace}}$, *i.e.*,

$$\forall\ X \in t\mathcal{E}^{\phi}_{C}\ :\ (Q||_C X) \leq_{ttrace} \alpha^{sk^w_{MC}}$$

We remind the reader that the intruder X can communicate with the honest participants only through channels $\in C$. In our settings, $C = \{c_1, c_2\}$. Also, special channels *PUBLIC, RUN, COMMIT* are not observable by the intruder, meaning that the intruder neither can write on nor listen to those channels. In fact, they are inserted only for signaling to the *external* world some points reached by the protocol. Finally, channels *time* and c_t are observable by the intruder, but not writable. This is because 1) we assume a reliable communication between the honest participants and the global clock, *i.e.*, is not possible for the intruder to pretend to be *Clock*, or to modify the messages communicated on channel *time*; and 2) *Timer* is just an artificial invention to simulate the short life time of the weak key.

For the sake of readability, we let:

$$\begin{array}{ll} G = & \{c_1, c_2, time, c_t\} \\ \alpha' = & \iota(COMMIT!(AP, pk^w_{MC})).\iota(PUBLIC!sk^w_{MC}).\iota(0)+ \\ & \iota(PUBLIC!sk^w_{MC}).\iota(0) \\ \alpha_2 = & \iota(PUBLIC!sk^w_{MC}).\iota(0) \end{array}$$

We also use $Q, Q', \ldots$ as a shortcut for $Q(sk^w_{MC}), Q'(sk^w_{MC}), \ldots$, and α as a shortcut for $\alpha^{sk^w_{MC}}$.

Definition 2 can be proved by finding a suitable weak simulation relation $\mathcal{R}$ between $(Q||_C X)$ and $\alpha^{sk^w_{MC}}$. The following relation $\mathcal{R}$ satisfies our need.

$$
\begin{aligned}
\mathcal{R} \;=\;& (((Q||X_{\phi_0})\backslash C, \alpha) \mid X_{\phi_0} \in t\mathcal{E}_C^{\phi_0}) \\
& \cup (((Q'||X_{\phi_0})\backslash C, \alpha) \mid X_{\phi_0} \in t\mathcal{E}_C^{\phi_0}, \\
& Q' = (Clock(0)||AP||RUN![\ldots]||Timer)\backslash G) \\
& \cup (((Q^2||X_{\phi_0})\backslash C, \alpha') \mid X_{\phi_0} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^2 = (Clock(0)||AP||c_1![\ldots]||Timer)\backslash G) \\
& \cup (((Q^3||X_{\phi_1})\backslash C, \alpha') \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^3 = (Clock(0)||AP||c_t![\ldots]||Timer)\backslash G) \\
& \cup (((Q^{3*}||X_{\phi_1})\backslash C, \alpha') \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1} \\
& Q^{3*} = (Clock(0)||AP||\iota([\ldots])||tick^{exp_{pk_{MC}^w}}.PUBLIC!sk_{MC}^w)\backslash G) \\
& \cup (((Q^{3**}||X_{\phi_1})\backslash C, \alpha') \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1} \\
& Q^{3**} = (Clock(m)||time?[...]||\iota([\ldots])||tick^m.PUBLIC!sk_{MC}^w)\backslash G, \\
& \textit{for some } m) \\
& \cup (((Q^{3'}||X_{\phi_0})\backslash C, \alpha') \mid X_{\phi_0} \in t\mathcal{E}_C^{\phi_0}, \\
& Q^{3'} = \textit{set with interleaved synchronizations on ct and time,} \text{ i.e.,} \\
& Q^{3'} = (Clock(0)||time?(x).[\ldots]||ct!([\ldots])||Timer)\backslash G) \\
& \cup (((Q^4||X_{\phi_1})\backslash C, \iota(0)) \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^4 = (Clock(exp_{pk_{MC}^w})||\iota([\ldots])||\iota([\ldots])||\iota(0))\backslash G) \\
& \cup (((Q^{4*}||X_{\phi_1})\backslash C, \alpha_2) \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^{4*} = (Clock(n)||[y \vdash_{hash} y'][...]||\iota([\ldots])||tick^n.PUBLIC!sk_{MC}^w[\ldots])\backslash G, \\
& n \leq exp_{pk_{MC}^w}) \\
& \cup (((Q^{4**}||X_{\phi_1})\backslash C, \iota(0)) \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^{4**} = (Clock(exp_{pk_{MC}^w})||\iota(0)||\iota([\ldots])||\iota(0))\backslash G) \\
& \cup (((Q^5||X_{\phi_1})\backslash C, \iota(0)) \mid X_{\phi_1} \in t\mathcal{E}_C^{\phi_1}, \\
& Q^5 = (Clock(exp_{pk_{MC}^w})||[AP \textit{ performed } COMMIT]||\iota([\ldots])||\iota(0))\backslash G)
\end{aligned}
$$

where X_ϕ represents the adversary X whose knowledge is ϕ. In particular, the initial knowledge ϕ_0 is a set of messages that do not include sensitive information, like, *e.g.*, the secret keys of the other participants. The initial knowledge increases when X receives some message over the channels over which it can communicate (*i.e.*, the set C). Thus, $\phi_1 = \phi_0 \cup Msg1$. As the honest participants, X can apply to the set of messages in its knowledge all the rules of the inference system in Fig. 9.

We omitted to explicitly put in $\mathcal{R}$ the pairs in which the first process performs deduction constructs.

Now, we show that $\mathcal{R}$ is a weak simulation. This is done by inspecting the specification of the single processes that form Q.

- $((Q||X_{\phi_0})\backslash C, \alpha)$. In Q, MC and $Clock$ can synchronize on $time$ and the result is the performance of a τ action. Globally, the intruder cannot interact, and its knowledge does not increase. Thus, $(Q||X_{\phi_0}) \xrightarrow{\tau} (Q'||X_{\phi_0})$, and α is able to simulate it. $(Q'||X_{\phi_0}, \alpha) \in \mathcal{R}$.

- $((Q'||X_{\phi_0})\backslash C, \alpha)$. In Q', MC can perform a sending action on channel RUN. Thus, $(Q'||X_{\phi_0}) \xrightarrow{RUN!} (Q^2||X_{\phi_0})$. Also α performs $RUN!$ and then behaves as α'. $(Q^2||X_{\phi_0}, \alpha') \in \mathcal{R}$.

- $((Q^2||X_{\phi_0})\backslash C, \alpha')$.
  - Q^2 may perform $c_1!$ and X_{ϕ_0} can synchronize on c_1. Thus, its knowledge becomes $\phi_1 = \phi_0 \cup Msg1$. Thus, $(Q^2||X_{\phi_0})\backslash C \xrightarrow{\tau} (Q^3||X_{\phi_1})\backslash C$. α' is able to simulate it, and $((Q^3||X_{\phi_1})\backslash C, \alpha') \in \mathcal{R}$.
  - In Q^2, MC can perform $c_1!$ and AP the corresponding receiving action. Thus, the intruder's knowledge remains ϕ_0, the whole left process performs only τ actions, *i.e.*, $(Q^2||X_{\phi_0})\backslash C \xrightarrow{\tau} (Q^{3'}||X_{\phi_0})\backslash C$. α' is able to simulate it, and $((Q^{3'}||X_{\phi_0})\backslash C, \alpha') \in \mathcal{R}$.
- $((Q^3||X_{\phi_1})\backslash C, \alpha')$. In Q^3, processes MC and $Timer$ can synchronize on ct. The intruder's knowledge does not increase. Thus, $(Q^3||X_{\phi_1})\backslash C \xrightarrow{\tau} (Q^{3*}||X_{\phi_1})\backslash C$. The τ transition is simulated by α', and the derivatives are still in $\mathcal{R}$.
- $((Q^{3*}||X_{\phi_1})\backslash C, \alpha')$. The process on the left will let time pass until 1) either AP stops idling, by performing a receiving action on c_1 (and in this case $X_{\phi 1}$ performs the corresponding sending action), applies inference rules, and possibly sends on $COMMIT$; 2) or $Timer$ sends on $PUBLIC$.
  - if AP performs a receiving action on c_1, and $X_{\phi 1}$ the corresponding sending action, thus the whole process on the left performs τ, and $(Q^{3*}||X_{\phi_1})\backslash C \xrightarrow{tau} (Q^{3**}||X_{\phi_1})\backslash C$. α' is able to simulate such an internal transition.
  - If the process on the left let time pass, $(Q^{3*}||X_{\phi_1})\backslash C \xrightarrow{tick} (Q^{3*}||X_{\phi_1})\backslash C$, with $Clock = Clock(n)$, for some $n \leq exp_{pk^w_{MC}}$. α' is able to let time pass.
  - If $Timer$ sends on $PUBLIC$, it could be:
    - the timeout expires, and AP never stops idling. This is the case in which the intruder dropped the intercepted message. $(Q^{3*}||X_{\phi_1})\backslash C \xrightarrow{PUBLIC!} (Q^4||X_{\phi_1})\backslash C$, and $\alpha' \xrightarrow{PUBLIC!} \iota(0)$. Both the derivatives are in $\mathcal{R}$.
    - the timeout expires. In the meanwhile, AP has received a message on c_1, and it has started to apply the rules for deducting new messages, but something was wrong and it was back to idle. Thus, $(Q^{3**}||X_{\phi_1})\backslash C \xrightarrow{PUBLIC!} (Q^{4**}||X_{\phi_1})\backslash C$, and $\alpha' \xrightarrow{PUBLIC!} \iota(0)$. Both the derivatives are in $\mathcal{R}$.
  - If AP issues a message on $COMMIT$, it means that all the deduction constructs have worked well, meaning that the message received on c_1 is a well formed message (with respect to the expected cryptographic structure), and that the time constraints have been respected. In particular, it was true that no more than $exp_{pk^w_{MC}}$ units of time passed since MC has started the protocol (recall that we are supposing $t_{MC} = 0$), and so $Timer$ has not issued on $PUBLIC$ yet.

$$\frac{x \quad y \quad z \ldots}{Tuple(x,y,z,\ldots)}(tuple) \qquad \frac{Tuple(x_1,x_2,x_3,\ldots)}{x_i}(i) \qquad \frac{x \quad k}{\{x\}_k}(enc)$$

$$\frac{x \quad sk(y)}{\{x\}_{sk(y)}}(sign) \qquad \frac{\{x\}_{sk(y)} \quad pk(y)}{x}(ver) \qquad \frac{x}{h(x)}(hash)$$

Figure 9: Inference system for the TIME$_c$ protocol. We also assume to have rules that handle the boolean relations on arithmetic expressions (*e.g.*, $[t \leq t']P$), as well as equality checks among messages (*e.g.*, $[m = m']P$).

$(Q^{3**}||X_{\phi_1})\backslash C \stackrel{COMMIT!}{\longrightarrow} (Q^{4*}||X_{\phi_1})\backslash C$, and $\alpha' \stackrel{COMMIT!}{\longrightarrow} \alpha_2$. Both the derivatives are in $\mathcal{R}$.

- $((Q^{4*}||X_{\phi_1})\backslash C, \alpha_2)$. The process on the left may perform $PUBLIC!$ by reaching a configuration $(Q^5||X_{\phi_1})\backslash C$. α_2 is able to perform $PUBLIC!$ becoming $\iota(0)$. Both the derivatives are in $\mathcal{R}$.

- The reasoning for $((Q^{3'}||X_{\phi_0})\backslash C, \alpha')$ (and their derivatives) is similar to the one for the pairs in which Q^3 and Q^{3*} appear in the process on the left. The difference is that the intruder remains inactive, *i.e.*, X never eavesdropped on c_1, and thus $\phi = \phi_0$.

- Finally, we consider the following three pairs:

  – $((Q^5||X_{\phi_1})\backslash C, \iota(0))$,
  – $((Q^{4**}||X_{\phi_1})\backslash C, \iota(0))$,
  – $((Q^4||X_{\phi_1})\backslash C, \iota(0))$.

  We are interested in visible actions, *i.e.*, actions observable by the external world in order to infer something about the (in)correct behaviour of the protocol. From an inspection of the specification, we see that all the three processes on the left are processes that either are able to let time pass, or they perform some internal actions not visible to the external world. $\iota(0)$ is able to simulate this behaviour.

6. Evaluation

In this section, we show that our mechanisms satisfy the special requirements relating to the QoS-aware multi-operator environment defined in Section 1.2:

- *Fast authentication method to support user mobility:* Our main design principle was to adopt public key cryptography in the considered multi-operator driven mesh network. However, we proved with an implementation and measuring the authentication delay that our proposed schemes reduce authentication delay to an extent that makes seamless handover possibl despite the usage of public key cryptography.

- *Mutual authentication:* Both the mesh client and the access point checks the authenticity of the other party.
- *DoS resistance:* The authentication is completely distributed, therefore, an attacker can defeat the access points only one by one. We also minimized the computational load of the APs, especially before the MC becomes authenticated.
- *Compatibility with standards:* We implement our proposals according to the EAP standard, therefore, it suits to standard IEEE 802.11i and IEEE 802.11r. Consequently, our authentication scheme can be used both for inter- and intra-domain handover. Note that our mechanism can coexist with other protocols, and the intra-domain handovers can be handled by other protocols, *e.g.*, that defined in IEEE 802.11r standard.
- *Scalability:* There is no central bottleneck because the authentication and the access control is distributed. A mesh network can be extended by installing valid certificates on the new access points. However, the computational overhead can cause delay if a lot of mesh clients associate to a specific access point at the same time.
- *No single trusted entity:* The access points can authenticate the mesh clients locally. The CRLs can be maintained by each operators' CA. Therefore, no single trusted entity is required.
- *Connection keys must not reveal long term keys:* The connection keys are based on random numbers generated for the handover and on timestamps, only. Therefore, they do not reveal any long-term key.
- *Independence of connection keys:* The random numbers are generated and the timestamps are read independently of the previous and upcoming connections.
- *Freshness:* The key is controlled by both participants, however, if the key sent by the access point encrypted is compromised, the connection key can be calculated by other parties, too. But only malicious access points send compromised keys and, in that case, they can reveal any not compromised keys, as well.

7. Conclusion

In this paper, we proposed two authentication protocols that support fast handover in multi-operator maintained wireless mesh networks. The motivation is that we found that none of the current standard and proposed solutions can satisfy the requirements on such network. For both schemes, we proposed two certificate sets: one for powerful mesh clients and one for constrained mesh clients. In the former set, the computationally intensive operations are shifted

to the mesh client, while in the latter certificate set, we proposed the usage of weak keys and short-term certificates for digital signatures.

We investigated how the usage of the weak key mechanism affects the authentication delay by analyzing the processing time of the messages, and we determined when the weak key mechanism is beneficial. We also proved formally that the use of our weak key mechanism on the mesh client side is as secure as the use of some stronger keys.

We created a proof-of-concept implementation, embedded it into the EAP messages, and measured the authentication delay compared to current widely used centralized authentication mechanisms such as EAP-TLS and EAP-TTLS. We found that our mechanism is faster than other certificate based mechanisms even though in our case one party is a constrained access point while the central authentication server is considered to be a powerful PC. We showed that our mechanisms satisfy special requirements relating to the QoS-aware multi-operator environment.

Acknowledgment

This work was supported in part by the European Commission in the context of the 7th Framework Programme through the EU-MESH Project (www.eu-mesh.eu), in part by the European Commission in the context of the 6th Framework Programme through the BIONETS Project (www.bionets.eu), and in part by the Mobile Innovation Center (www.mik.bme.hu). The authors would like to thank Gergely Ács and Péter Schaffer for commenting on an earlier version of this paper.

References

[1] I. Askoxylakis, B. Bencsáth, L. Buttyán, L. Dóra, V. Siris, D. Szili, I. Vajda, Securing Multi-operator Based QoS-aware Mesh Networks: Requirements and Design Options, Wireless Communications and Mobile Computing (Special Issue on QoS and Security in Wireless Networks).

[2] L. Buttyán, L. Dóra, An Authentication Scheme for QoS-aware Multi-operator maintained Wireless Mesh Networks, in: In Proceedings of the First IEEE WoWMoM Workshop on Hot Topics in Mesh Networking (HotMESH'09), Kos, Greece, 2009, pp. 1–6.

[3] P. Calhoun, M. Montemurro, D. Stanley, Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification, RFC 5415 (Proposed Standard) (Mar. 2009).
URL `http://www.ietf.org/rfc/rfc5415.txt`

[4] P. Calhoun, M. Montemurro, D. Stanley, Control and Provisioning of Wireless Access Points (CAPWAP) Protocol Binding for IEEE 802.11, RFC 5416 (Proposed Standard) (Mar. 2009).
URL `http://www.ietf.org/rfc/rfc5416.txt`

[5] IEEE Std 802.11i^{TM}, Medium Access Control (MAC) security enhancements, amendment 6 to IEEE Standard for local and metropolitan area networks part 11: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications. (July 2004).

[6] V. Narayanan, L. Dondeti, EAP Extensions for EAP Re-authentication Protocol (ERP), RFC 5296 (Proposed Standard) (Aug. 2008).
URL `http://www.ietf.org/rfc/rfc5296.txt`

[7] IEEE 802.11r^{TM}-2008, IEEE Standard for Information Technology – Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 2: Fast BSS Transition (July 2008).

[8] Y. Zhang, Y. Fang, A secure authentication and billing architecture for wireless mesh networks, Wireless Networks 13 (5) (2007) 663–678. doi:http://dx.doi.org/10.1007/s11276-006-8148-z.

[9] J.-J. Chen, Y.-C. Tseng, H.-W. Lee, A Seamless Handoff Mechanism for IEEE 802.11 WLANs Supporting IEEE 802.11i Security Enhancements, in: IEEE Asia-Pacific Wireless Communications Symposium, Hsinchu, Taiwan, 2007.

[10] T. Chen, G. Schäfer, C. Fan, S. Adams, M. Sortais, A. Wolisz, Denial of Service Protection for Optimized and QoS-aware Handover Based on Localized Cookies, in: Proc. of European Wireless, Barcelona, Spain, 2004, pp. 155–161.

[11] T. Aura, M. Roe, Reducing Reauthentication Delay in Wireless Networks, in: IEEE International Conference on Security and Privacy for Emerging Areas in Communications Networks, Athens, Greece, 2005, pp. 139–148. doi:http://dx.doi.org/10.1109/SECURECOMM.2005.58.

[12] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, W. Polk, Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC 5280 (Proposed Standard) (May 2008).
URL `http://www.ietf.org/rfc/rfc5280.txt`

[13] D.VAM.14, Performing Benchmarks, IST-2002-507932, ECRYPT, European Network of Excellence in Cryptology (2008).

[14] D.SPA.28, ECRYPT Yearly Report on Algorithms and Keysizes (2007-2008), IST-2002-507932, ECRYPT, European Network of Excellence in Cryptology (2008).

[15] S. Blake-Wilson, A. Menezes, Entity authentication and authenticated key transport protocols employing asymmetric techniques, in: Proc. of the 5th International Workshop on Security Protocols, London, UK, 1998, pp. 137–158.

[16] C. Boyd, A. Mathuria, Protocols for authentication and key establishment, Berlin: Springer-Verlag, 2003.

[17] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, H. Levkowetz, Extensible Authentication Protocol (EAP), RFC 3748 (Proposed Standard), updated by RFC 5247 (Jun. 2004).
URL http://www.ietf.org/rfc/rfc3748.txt

[18] IEEE Std 802.1X-2001, IEEE Standard for Local and Metropolitan Area Networks - Port-Based Network Access Control (June 2001).

[19] J. Malinen, WPA/RSN Supplicant (wpa_supplicant) and WPA/RSN/EAP Authenticator (hostapd) v0.6.7, http://hostap.epitest.fi/ (2009).

[20] OpenSSL, http://www.openssl.org.

[21] A. Alimian, B. Aboba, "analysis of roaming techniques", IEEE Contribution 802.11-04/0377r1 (March 2004).

[22] C. Rigney, S. Willens, A. Rubens, W. Simpson, Remote Authentication Dial In User Service (RADIUS), RFC 2865 (Draft Standard), updated by RFCs 2868, 3575, 5080 (Jun. 2000).
URL http://www.ietf.org/rfc/rfc2865.txt

[23] P. Funk, S. Blake-Wilson, Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0), RFC 5281 (Informational) (Aug. 2008).
URL http://www.ietf.org/rfc/rfc5281.txt

[24] D. Simon, B. Aboba, R. Hurst, The EAP-TLS Authentication Protocol, RFC 5216 (Proposed Standard) (Mar. 2008).
URL http://www.ietf.org/rfc/rfc5216.txt

[25] H. Tschofenig, D. Kroeselberg, A. Pashalidis, Y. Ohba, F. Bersani, The Extensible Authentication Protocol-Internet Key Exchange Protocol version 2 (EAP-IKEv2) Method, RFC 5106 (Experimental) (Feb. 2008).
URL http://www.ietf.org/rfc/rfc5106.txt

[26] T. Clancy, W. Arbaugh, Extensible Authentication Protocol (EAP) Password Authenticated Exchange, RFC 4746 (Informational) (Nov. 2006).
URL http://www.ietf.org/rfc/rfc4746.txt

[27] M. Vanderveen, H. Soliman, Extensible Authentication Protocol Method for Shared-secret Authentication and Key Establishment (EAP-SAKE), RFC 4763 (Informational) (Nov. 2006).
URL http://www.ietf.org/rfc/rfc4763.txt

[28] J. A. Goguen, J. Meseguer, Security policies and security models, in: IEEE Symposium on Security and Privacy, 1982, pp. 11–20.

[29] R. Focardi, F. Martinelli, A uniform approach for the definition of security properties, in: World Congress on Formal Methods, 1999, pp. 794–813.

[30] R. Gorrieri, E. Locatelli, F. Martinelli, A simple language for real-time cryptographic protocol analysis, in: ESOP, 2003, pp. 114–128.

[31] R. Gorrieri, F. Martinelli, A simple framework for real-time cryptographic protocol analysis with compositional proof rules, Sci. Comput. Program. 50 (1-3) (2004) 23–49.

[32] I. Ulidowski, S. Yuen, Extending process languages with time, in: AMAST '97: Proceedings of the 6th International Conference on Algebraic Methodology and Software Technology, Springer-Verlag, London, UK, 1997, pp. 524–538.

[33] G. Lowe, A hierarchy of authentication specification, in: CSFW, 1997, pp. 31–44.

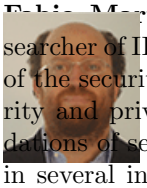
[34] R. Gorrieri, F. Martinelli, M. Petrocchi, Formal models and analysis of secure multicast in wired and wireless networks, J. Autom. Reasoning 41 (3-4) (2008) 325–364.

[35] R. Gorrieri, F. Martinelli, M. Petrocchi, A. Vaccarelli, Formal anaylsis of some timed security properties in wireless protocols, in: FMOODS, 2003, pp. 139–154.

[36] R. Focardi, R. Gorrieri, F. Martinelli, Secrecy in security protocols as non interference, Electr. Notes Theor. Comput. Sci. 32.

Levente Buttyán received the M.Sc. degree in Computer Science from the Budapest University of Technology and Economics (BME) in 1995, and the Ph.D. degree from the Swiss Federal Institute of Technology, Lausanne (EPFL) in 2002. In 2003, he joined the Department of Telecommunications at BME, where he currently holds a position as Associate Professor and works in the Laboratory of Cryptography and Systems Security (CrySyS). His research interests are in the design and analysis of security protocols and privacy enhancing mechanisms for wireless networked embedded systems (including wireless sensor networks, mesh networks, vehicular communications, and RFID systems), and the application of formal methods in security engineering. He is involved in several EU funded research projects, including the EU-MESH project (`www.eu-mesh.eu`), where he leads the work package on security. More information is available at `www.crysys.hu`.

László Dóra received the M.Sc. degree in Computer Science from the Budapest University of Technology and Economics (BME) in 2005. During his M.Sc. he joined the Laboratory of Cryptography and Systems Security (CrySyS) in 2004. Since 2005 he is a Ph.D. student at the same laboratory under the supervision of Levente Buttyán. His research interests are in security of opportunistic and mesh networks. More personal information is available at `www.crysys.hu/dora/`



Fabio Martinelli (M.Sc. 1994, Ph.D. 1999) is a senior researcher of IIT-CNR, Pisa, where he is the scientific coordinator of the security group. His main research interests involve security and privacy in distributed and mobile systems and foundations of security and trust. He serves as PC-chair/organizer in several international conferences/workshops. He is the co-initiator of the International Workshop series on Formal Aspects in Security and Trust (FAST). He is serving as scientific co-director of the international research school on Foundations of Security Analysis and Design (FOSAD) since 2004 edition. He chairs the WG on security and trust management (STM) of the European Research Consortium in Informatics and Mathematics (ERCIM). He usually manages R&D projects on information and communication security and he is involved in several FP6/7 EU projects.

Marinella Petrocchi received her M.Sc. in Telecommunication Engineering from the University of Pisa in 1999, and her Ph.D. in Information Engineering from the same University in 2005. She is currently a researcher of the Information Security Group of the Istituto di Informatica e Telematica of the Consiglio Nazionale delle Ricerche. Her main research interests involve formal models and analysis of security and trust, particularly focused on bio-inspired and distributed systems, and on techniques for context-awareness information sharing. She is co-author of several papers on international journals and conference/workshop proceedings. She is involved in both FP6 and FP7 European projects on information and communication security. She also serves as PC-chair/organizer in several international conferences/workshops.