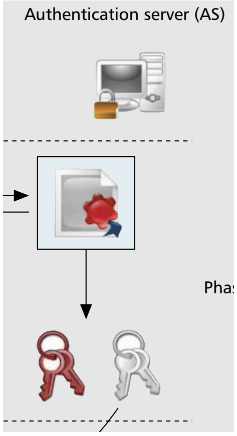
ANALYSIS OF SECURE HANDOVER FOR IEEE 802.1X-BASED WIRELESS AD HOC NETWORKS

ROMANO FANTACCI, LEONARDO MACCARI, AND TOMMASO PECORELLA,
UNIVERSITY OF FLORENCE
FEDERICO FROSALI, TELECOM ITALIA LAB



The authors present the 802.1X model and evaluate its application to ad hoc networks based on IEEE 802.11i or IEEE 802.16e standards, focusing on the problems that must be evaluated when designing handover procedures, and suggesting guidelines for securing handover procedures.

ABSTRACT

The handover procedure in secure communication wireless networks is an extremely time-consuming phase, and it represents a critical issue in relation to the time constraints required by certain real-time traffic applications. In particular, in the case of the IEEE 802.1X model, most of the time required for a handover is used for packet exchanges that are required for authentication protocols, such as Extensible Authentication Protocol Transport Layer Security (EAP-TLS), that require an eight-way handshake. Designing secure re-authentication protocols to reduce the number of packets required during a handover is an open issue that is gaining interest with the advent of a pervasive model of networking that requires real-time traffic and mobility. This article presents the 802.1X model and evaluates its application to ad hoc networks based on *IEEE 802.11i* or *IEEE 802.16e* standards, focusing on the problems that must be evaluated when designing handover procedures, and suggesting guidelines for securing handover procedures. It also presents a novel protocol to perform secure handovers that is respectful of the previous analysis and that has been implemented in a mesh environment.

INTRODUCTION

In a wireless network, a lack of a defined geographical border makes the network subject to attacks from enemies outside the area of control of the administrator of the network. To meet this problem, modern wireless communication standards include security functions directly in the medium access control (MAC) layer. Mobility introduces new challenges, because users roam across networks that are managed by different entities, and the same local area network (LAN) could be composed of clients not trusted by each other. Then, security protocols in the MAC layer also must be as resistant as possible to attacks coming from inside of the network.

To manage access control at the MAC layer, standards such as WiFi (*IEEE 802.11i*; see [1]) and WiMax (*IEEE 802.16e*; see [2]) introduced the *IEEE 802.1X* [3] model that guarantees a high robustness and manageability to the network. However, the *IEEE 802.1X* authentication phase introduces long delays and does not scale well in mobile networks, where clients often perform handoffs between different access points and must re-authenticate on each handoff.

Both WiFi and WiMax have an ad hoc/mesh mode of use. WiFi ad hoc networks are the most widely used today, but there also are great expectations for WiMax for mesh mode (see [4]). If applied to mobile ad hoc networks (MANET), *IEEE 802.1X* affects the handover phase even more, because authentication must be performed over a multihop path with long delays.

In this article, we give an overview of the *IEEE 802.1X* security protocol, focus on its application to mobile ad hoc networks, and approach the problem of secure handovers. We describe common problems that must be faced when designing security re-authentication protocols, with special attention to issues related to ad hoc networks. We outline guidelines for a developer to produce protocols that offer a secure design. Lastly, we introduce an example of a scheme for re-authentication in ad hoc *IEEE 802.1X* networks that was designed and implemented and has demonstrated good performance results, while maintaining a high level of security.

SECURITY PARADIGM OF *IEEE 802.1X* NETWORKS

In this section, we describe the *IEEE 802.1X* standard, introduce its terminology, and illustrate how it has been applied to *IEEE 802.11i*. We also describe how this model was imported into *IEEE 802.16e* and the difficulties of using it in a MANET due to the loss of performance it produces.

IEEE 802.1X specifies the following three roles for agents that are involved in an authentication process:

In IEEE 802.11i ad hoc mode, every node of the network plays the role of supplicant when it enters the network, but afterwards it should be able to play the role of authenticator for other nodes. This condition raises several problems.

- *Supplicant*: the terminal entering the network that must be authenticated; we abbreviate supplicant as a generic station.
- *Authenticator* (Au): the terminal that is directly connected with the supplicant and that acts as a proxy to the *authentication server*; in *IEEE 802.11* networks, it is the access point.
- *Authentication server* (AS): the database containing the credentials for all the users; it must be reachable by the authenticator.

IEEE 802.1X also defines how to envelop the Extensible Authentication Protocol (EAP) into 802 frames, with an EAP over LAN protocol. EAP (see [5]) is a protocol for transporting *authentication methods*; it offers a framework in which many different authentication methods — certificate-based Extensible Authentication Protocol Transport Layer Security (EAP-TLS [6]) or password-based Microsoft Challenge-Handshake Authentication Protocol (EAP-MS-CHAP) — can be used. Using EAP improves extendibility and does not limit the standard to a specific method.

To better clarify the structure of *IEEE 802.1X*, we introduce another concept: a security association (SA) is a set of policies, algorithms, and keying material used to protect the communication. For the sake of simplicity, we use the terminology used by *IEEE 802.11i* in infrastructure networks to describe keys and SA.

Whenever a new station enters the network, it starts an authentication phase with an access point with which it enters in contact. However, as a matter of fact, the access point is acting as a proxy to the AS of the network — tunneling EAP messages between the two endpoints — so that *IEEE 802.1X* authentication is performed between supplicant and authentication server. An SA called pairwise master key SA (PMKSA) is generated between these two endpoints; it contains a key named PMK. Another key is generated from the first handshake, called extended master session key (EMSK), that currently is unused.

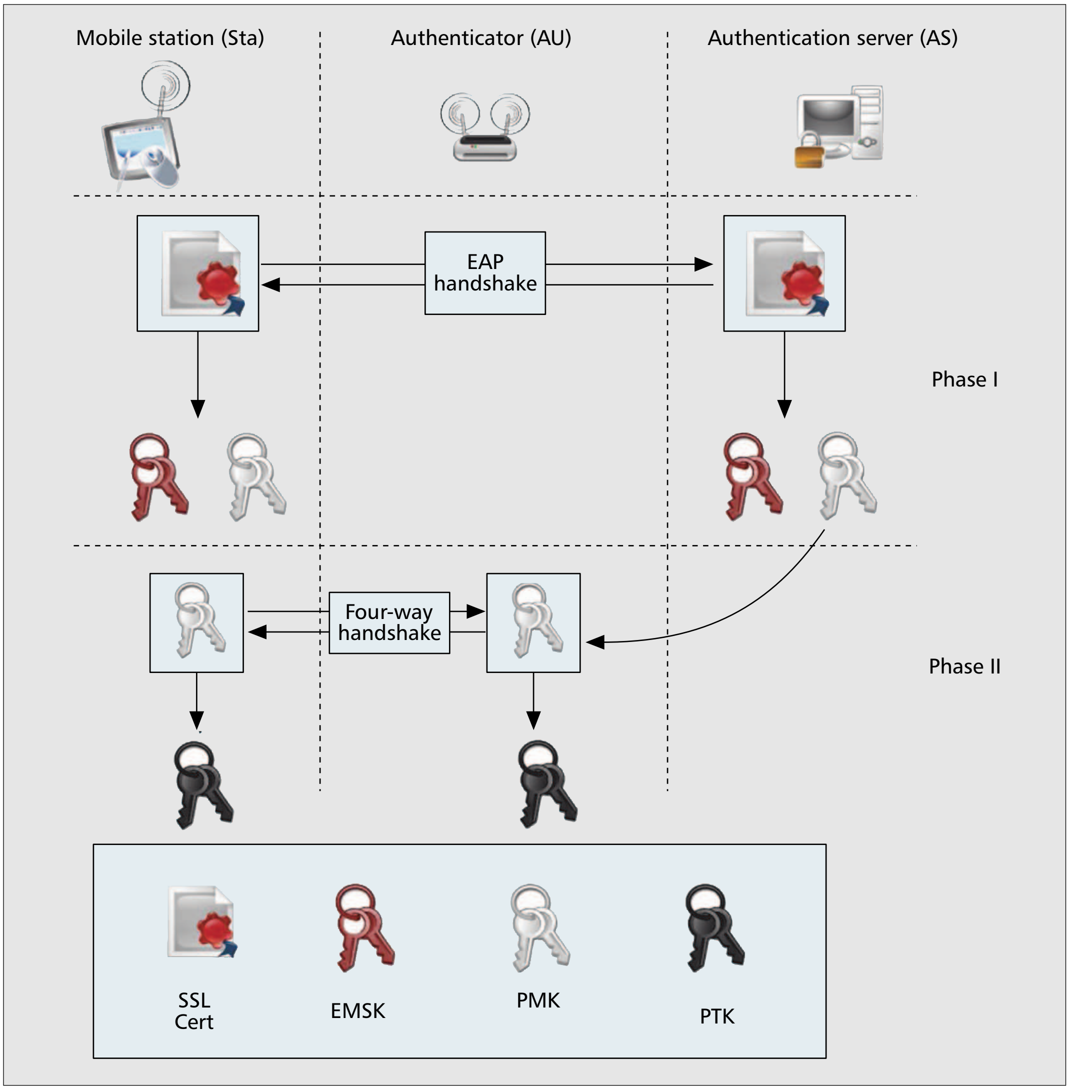
After the authentication phase, the authentication server moves the PMK into the authenticator. This is possible if the authentication server and the authenticator share a secure channel to move the key. *IEEE 802.11i* states that a back-end protocol such as remote authentication dial in user service (RADIUS) [7] must be used to perform this exchange. RADIUS uses a shared secret to cipher and authenticate the communications between the authenticator and the authentication server.

Then, another phase starts in which supplicant and authenticator perform the so-called four-way handshake that is required to derive a new SA called pairwise transient key (PTKSA). This SA includes the PTK that is finally used to cipher and authenticate traffic. This mechanism is described in Fig. 1. It is important to understand that when a supplicant has obtained a PMKSA, it is allowed to enter the network but still cannot send traffic. Traffic encryption is a link layer procedure, so it can be performed only after it also has obtained PTKSA from the access point.

When a station moves from one access point to another (i.e., performs a handover), it never really leaves the network, so it is correct to state that only a PTKSA must be renewed, while a PMKSA can be maintained. The problem that must be faced is how to transmit to the second authenticator the PMK to negotiate a new PTKSA. It must be noted that although the generation of a PTKSA involves two machines that are only one hop away, renewal of a PMKSA (re-authentication) requires a time consuming multihop handshake. If the supplicant must repeat the generation of a new PMKSA every time it performs a handover, that handshake can introduce significant delay. We note that *IEEE 802.1X* defines a framework that can be used to perform access control, but leaves unsolved the problem of how to move credentials from one authenticator to another to avoid the repetition of the entire user-authentication phase. This problem is approached differently in each implementation of *IEEE 802.1X*, but there is room for much improvement, especially in mobile ad hoc networks.

In *IEEE 802.11i* ad hoc mode, every node of the network plays the role of supplicant when it enters the network, but afterwards it should be able to play the role of authenticator for other nodes. This condition raises several problems. First, it is not clear if the node should be in possession of a RADIUS key that it will use after it becomes an authenticator. We can imagine that every node will be equipped with a different pre-shared key before network entrance, or the key might be derived from the first phase of Fig. 1, for example, from EMSK. Secondly, because the path from the authenticator to the authentication server might be several hops long, EAP authentication should not be repeated, or it could introduce intolerable delays (for example, the Transport Layer Security (TLS) authentication requires at least an eight-way exchange between supplicant and authentication server that might take several seconds). The standard defines a pre-authentication procedure, in which a supplicant can pre-authenticate with a different access point through the one with which it is currently associated. This implicitly assumes that a supplicant knows where it will roam and what access point it will find along its path, and it is difficult to use in a more dynamic scenario, such as a MANET.

A similar situation can be found in *IEEE 802.16e* networks. Amendment e introduces mobility and security enhancements to the first version of the standard; the base station behaves as an authenticator between the mobile station and a centralized database of authentication credentials. EAP can be used for authentication and generation of SA, but without explicitly mentioning *IEEE 802.1X*, EAP packets are encapsulated in layer II frames from and to the base station. At the network entrance, user authentication is performed with a certificate-based method (it might be EAP-TLS or a custom defined one). After user authentication, the new client receives a key, which in mesh mode is called operator shared secret (OSS). This key plays the same role as the PMK key for *IEEE 802.11i* but with a huge difference: it is the same for all the nodes in the network. After the reception of the OSS, a procedure to derive link keys follows that plays the same role of a four-way handshake in Fig. 1.



■ **Figure 1.** *Different phases of authentication and key management. In phase I, with a first handshake the PMKSA is created between Sta and AS through Au then PMK is moved to Au. In phase II, with a second handshake PTK is derived from PMK and data communication can start. In IEEE 802.11i the access point plays the role of authenticator.*

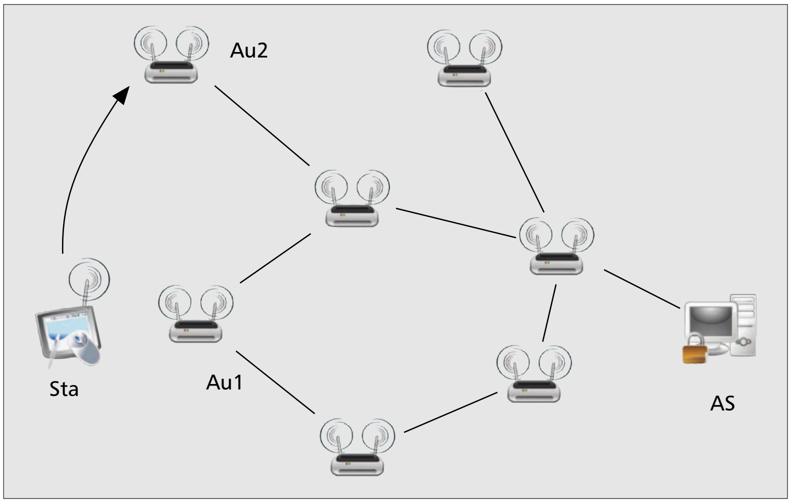
There is an IETF working group devoted to the extension of the EAP key framework to support handover, but it has published no draft protocols yet. In any case, the trend of the working group is toward the use of the EMSK for re-authentication purposes.

A handshake called *neighbor link establishment* is used by the mobile stations to perform roaming in mesh mode. When a node roams, it performs the handshake with the next target authenticator to verify that both stations own the OSS. An in-depth analysis of the neighbor link establishment can be found in [8], showing that the handshake itself is not well designed from a security point of view. Apart from implementation details, we note that the handshake is an attempt to substitute EAP authentication with a single-hop exchange, but it is based on the assumption that every node shares the same key. This introduces great difficulties in key refresh (how is the same key refreshed for all the nodes at the same time and if the refresh is not performed at the same time, how is cryptographic synchronization retained between nodes?) so that it is unlikely that this model will ever be used in a MANET. More likely, a more secure approach, with a distinct key for each node will be introduced.

Approaches in Literature

The problem of a secure, fast handover is well know for ad hoc networks. As seen, it has been partly addressed in the standards but in real applications, it is still an open issue. There is little literature on the topic; for example, in [9] a proactive solution for static networks is proposed. The article focuses on wireless networks composed of static multiple access points with mobile clients roaming through the network and performing handovers. The proposed solution is based on the fact that the paths of the clients are predictable (e.g., in indoor environments), so when a client enters the network, its access points preemptively can distribute keys to other access points that are on the possible path the client may use.

There is an Internet Engineering Task Force (IETF) working group devoted to the extension of the EAP key framework to support handover (see [10]), but it has published no draft protocols yet. In any case, the trend of the working group is toward the use of the EMSK for re-authenti-



■ **Figure 2.** *A mesh network composed of access points, with a mobile client performing a handover. If instead of access points there are mobile devices as in a pure MANET, handovers are more frequent.*

cation purposes, as proposed in this article. IEEE has created a working group to produce a new amendment *r* to IEEE 802.11 that should implement fast and secure handover. The development is still in progress. From documents publicly available at this stage, we can say the work focuses on an effort to modify the key tree defined by *IEEE 802.11i* to generate new keys to deliver to the access points to enable them to manage client handovers. These new keys are moved between access points belonging to a same mobility domain, possibly in two modes (push or pull). Later we examine how these modes can impact the security of the system.

Other technologies, such as cellular networks, support faster handover than wireless mesh networks, but the infrastructure they are based on is not comparable. More specifically, in wireless mesh networks, the link between each node is wireless and is used also for backhaul traffic, so that frames carrying authentication must pass over a busy multihop path. Moreover, there is no trust requirement on the base station itself, which means that the possibility of a corrupted base station is not considered.

Handoff Specific Security Issues

As observed, *IEEE 802.1X* defines roles and protocols to be used for access control but leaves unspecified the handover procedure, and *IEEE 802.11i* and *IEEE 802.16e* deliver solutions of limited applicability. In this section, we give an in-depth analysis of the security problems faced when designing security protocols for handover to be applied to mobile ad hoc networks.

Figure 2 represents a handover in a mobile ad hoc network. Using *IEEE 802.11i* terminology, we focus on the following problems:

- If Au_1 is an insider attacker working together with Sta, it might convince Au_2 to let Sta enter, even if it does not have the correct credentials, pushing a false PMK key. Alternatively, if Sta is not an attacker, Au_1 may push a false key with the aim of producing a denial of service attack.
- If Au_2 itself is an insider enemy, it could try to pull PMK keys from Au_1, thus gathering decryption keys to decipher traffic it should not be capable of accessing.

When using the terms, *compromised station* or *insider attacker*, we refer to two possibilities: a malicious authorized user of the network or a terminal that has fallen under the control of an external attacker. In both cases, the enemy always can perform certain actions:

- It can decrypt traffic that is passing over its direct links.
- It can masquerade a hidden network, allowing an unwanted machine to access the services of the network.
- It can create denial of services.

It is impossible to completely avoid these problems, but secure authentication and re-authentication protocols must be capable of limiting the impact these problems have on the network. We define the following guidelines for the design of a secure, fast handover authentication protocol:

G1 *A compromised station must not be capable of admitting into the network other unauthorized stations, if not physically connected to the compromised one.*

G2 *A compromised station must not be capable of decrypting all the traffic of the network, even considering the possibility of an off-line attack.*

G3 *A compromised machine should not be capable of performing denial of service attacks using the handover procedure.*

Let us specify these conditions. An attacker that successfully takes control of a station always can physically attach a subnet and masquerade it to the rest of the network. This is unavoidable; what we want to avoid with G1 is the capability of an attacker to generate valid credentials for other unauthorized stations not directly attached to it. Consider WiMax: if one node is compromised, the attacker can obtain the OSS key, and once obtained, the attacker can pass it to other stations to let them enter because the neighbor link establishment uses only the OSS key. This is exactly what we want to prevent. Moreover, it must be noted that WiMax uses digital certificates for initial user authentication, but they are not used in neighbor link establishment, so that the disclosure of the OSS key can completely bypass the security level achieved with certificates.

The second condition deals with the speed of penetration of an attacker into the network. Because we do not consider that wireless networks are as secure as wired ones, we join a model in which wireless security is not provided by a shield that avoids successful attacks, but more is a multi-fence organization of the network, where each element (network protocol, firewall, intrusion detection system, etc.) makes a contribution to minimize the impact of an attack in a dynamic and reactive way as described in [11]. It is fundamental that if a network station has been successfully attacked, this must not imply loss of trust in the whole network, but the disclosure of traffic and loss of trust in other machines can follow a graceful degradation scheme. We specify this condition stating that intrusion in a single terminal should imply disclosure of traffic passing directly through that machine only, and not of traffic passing through other terminals even if previously collected (off-line). The third condition is straightforward, as mentioned when commenting on Fig. 2, introducing an excessive trust relationship between authenticators may give Au2 the opportunity to inject unusable keys, thus producing a denial of service.

An important issue to be considered is whether the authentication server must be contacted on every handover, or if the handover procedures can be dealt with by the authenticators independently. The main advantage of not including the authentication server is that then the handover does not require a multihop communication, and re-authentication is accomplished with handshakes involving only the moving station and the authenticator involved. The main advantage of always contacting the authentication server is that there is stricter control over re-authentication. We now discuss in more detail the two strategies, using *IEEE 802.11i* terminology for clarity:

Strategy One: Inter-Authenticator Re-authentication

Enforcing this strategy means excluding the AS from decisions made during re-authentication. Referring to Fig. 2, Au_2 may decide to accept Sta, using only communication with neighbor Au_1. With this strategy, a compromised authenticator can deliver fake keys to its neighbor, with the aim of letting other unauthenticated stations enter the network (breaking G1) or making a handover fail, causing a denial of service (breaking also G3).

Another issue introduced by this strategy is the definition of a policy for the transmission of the keys that can be *push* or *pull*.

With push key distribution, whenever a station reaches an access point, the access point itself sends the PMKSA material to its neighbors so that when the station decides to roam to one of the neighbors, they already possess the PMK and the handover is quicker. The drawback of this solution is that an access point must have knowledge of the physical distribution of neighbor access points, for example, where the station most likely will move. If there are no fixed paths, the access point might try to guess which are its closest neighbors; the more neighbors it contacts, the higher the possibility of guessing the next step for the roaming station. From the point of view of security, separating the key distribution phase from the handover, implies the possible distribution of keys to machines that do not require them, decreasing the overall security of the system.

With pull key distribution, whenever Au_2 is reached by Sta, it requests the PMKSA material from Au_1 (if reachable). In this situation, there is no need for a fixed topology, but there might be more security risks. It is important that PMK keys not be moved in this process but only refreshed. Otherwise, if compromised, Au_2 may ask from Au_1 any PMK key it owns and repeat the same request to any access point of the network, giving the attacker opportunities to decrypt all the traffic passing over the network.

In general, performing the handover without contacting the authentication server provides better performance but is more complicated to deal with. It is advisable in environments when:

- Handover speed is extremely important, and the path to the authentication server is long.
- The network cannot depend on a single point of failure (note that this strategy permits handovers even if the link to the authentication server is temporarily broken).

In any case, when designing secure handover protocols, the possibility that an attack can be led in cooperation with a compromised access point and a roaming station must be considered.

Strategy Two: Re-Authentication Involving the Authentication Server

If the authentication server must be contacted on every handover, in a Kerberos-like authentication (see [12] for the Kerberos protocol) and even if there is a loss of performance, the authentication server can control the movements of the stations belonging to the network, and it can allow or disallow the handover (e.g., basing its decision on the time passed since the first authentication). Still, there could be danger if the protocol that is used to move the PMKSA is not strong enough. Imagine a simple mechanism where Au_2 may request and receive only the correct PMK from the authentication server whenever Sta performs the handover. If this request is not strongly motivated and authenticated, but is based on the sole presence of a security association between Au_2 and AS (i.e., the RADIUS channel), then we have the same problems as outlined in the pull method described previously.

A Fast Re-Authentication Scheme for *IEEE 802.1X* Networks

We developed a fast re-authentication protocol following the guidelines explained in the previous paragraphs. It is based on the second strategy and has demonstrated good performance in a working implementation in a prototype testbed. Because the testbed is composed of WiFi nodes, we focus on *IEEE 802.11i* protocols and terminology. In particular, we compared our solution to EAP-TLS, which is used mainly in enterprise networks.

When using EAP-TLS, each handover requires the repetition of the full authentication, and the exchange is composed of eight multihop packets and two one-hop packets as described in Fig. 3. Before the EAP-TLS phase, standard specific frames are transmitted, but they affect the overall performance much less than the EAP-TLS phase, being only one-hop exchanges.

The goal of our proposal is to reduce the number of packets required for TLS exchange, by re-using information generated in the first authentication. *Fast re-authentication* is based on the following principle, referring to Fig. 2: whenever Sta moves from Au_1 to Au_2, Au_2 must request and receive the PMKSA material from the AS, motivating its request with an authentication token. A token is cryptographic material tes-

An important issue to be considered is whether the authentication server must be contacted on every handover, or if the handover procedures can be dealt with by the authenticators independently.

tifying that Au_2 is in contact with a station that was previously authenticated, so it must contain data from the supplicant that the authentication server can validate. These can be keying materials derived from a PMK or EMSK; because EMSK never leaves the authentication server and the supplicant, we decided to use this key.

The token is generated by the supplicant in the following way:

$$token = [RANDOM, Au_{id}, H(Au_{id}, RANDOM, EMSK), EMSKID] \quad (1)$$

where:

- *RANDOM* is a sufficiently large pseudo-random value; a 160-bit string should fit. We will see how accurate generation of this number can improve security.

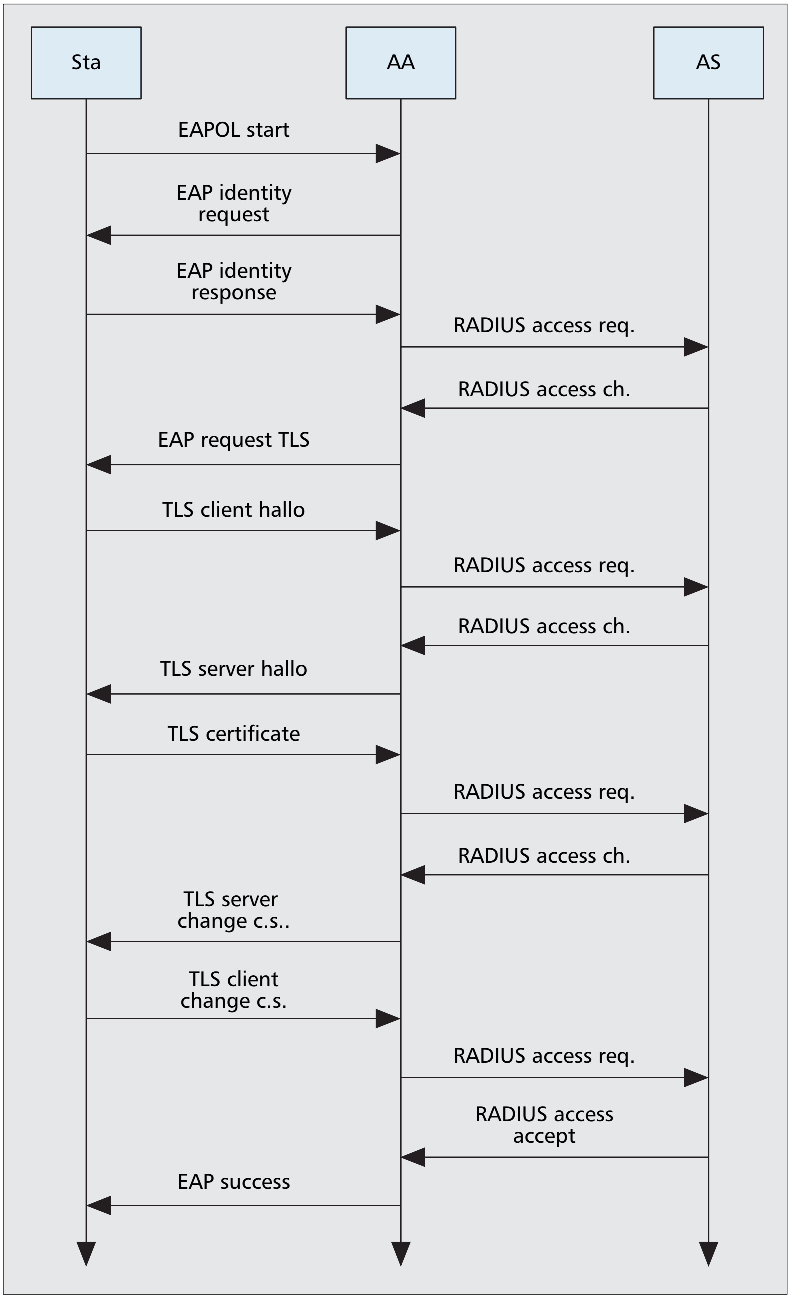


Figure 3. *Packet exchange sequence for EAP-TLS authentication. The last RADIUS packet contains the PMK key being moved from AS to Au.*

- Au_{id} is an identifier of the target *Au* known to the supplicant and to the authentication server. For example, in an 802.11 network, the extended service set ID (ESSID) or the basic service set ID (BSSID) value of the authenticator could also be chosen as a RADIUS login for the authenticator to the authentication server. Because the ESSID and the BSSID are sponsored in beacon frames, it would be known to both the authentication server and to the supplicant. Referring to the example of Fig. 2, it should contain the ESSID of Au_2.
- *H*() is a secure hash function, such as an SHA-2 function.
- EMSKID is an identifier of the EMSK, used only for indexing purposes.

The authentication server verifies the hash using the EMSK key corresponding to EMSKID. If verification succeeds, the authentication server forwards the PMK' key to the authenticator. PMK' is a fresh key that will be used as a PMK key for the following four-way handshake. It is derived as follows:

$$PMK' = H(RANDOM, EMSK) \quad (2)$$

Figure 4 shows the simple message exchange that realizes the token-based re-authentication. The packets are defined as follows:

- **Token**: a packet containing the token.
- **AuthenticatedToken**: a packet containing the token but also a form of authentication that identifies authenticator to authentication server. If RADIUS protocol is used, packets are already authenticated into a RADIUS access request.
- **Keymessage**: a packet containing the PMK′ key. It also can be included into a RADIUS packet.

Security analysis: To forge a token, an attacker should be in possession of a valid EMSK key. If the attacker is an outsider attacker, it does not own one, so it can't forge a new valid token. It still could attempt other attacks, such as the following:

- It could replay a previous token, as tokens are transmitted on a wireless channel in clear, but it could not follow PTKSA generation. This is done through the four-way handshake that is based on knowledge of the PMK′, so this handshake (or any equivalent one) will not meet with success.
- It could try to brute force the AS, sending multiple requests, but the EMSK key is sufficiently long (a minimum of 64 octets in length) to make this attack unfeasible.
- It could forge tokens to produce a denial of service against the AS. This attack is feasible, but it also is present in EAP-TLS authentication in a stronger form, because it involves public key verification. In our protocol, only hash functions are used.

If the attacker is an insider, we have the following:

- Authenticators do not generate or move PMK keys, so they cannot generate valid credentials to enable other unauthenticated stations to enter the network (G1) or to produce denial of services (G3).
- Authenticators can obtain a key only when they can prove that they are in contact with a

station that already has been authenticated. Moreover, the received keys are freshly generated, so they cannot be used to decipher past traffic (G1, G2).

However, an insider attacker could forge a valid token and give it to an unauthenticated attacker that could use it to log in to a different access point, breaking G1. To address this issue, the token should be generated in cooperation between the supplicant and the authenticator, so that when the authenticator receives it, it is assured that it has not been replied or previously generated by an insider attacker. This issue could be addressed introducing an exchange of nonces (fresh random values) before the transmission of the token, but this would increase the handover time. Alternatively, we can use two different policies depending on the MAC layer used:

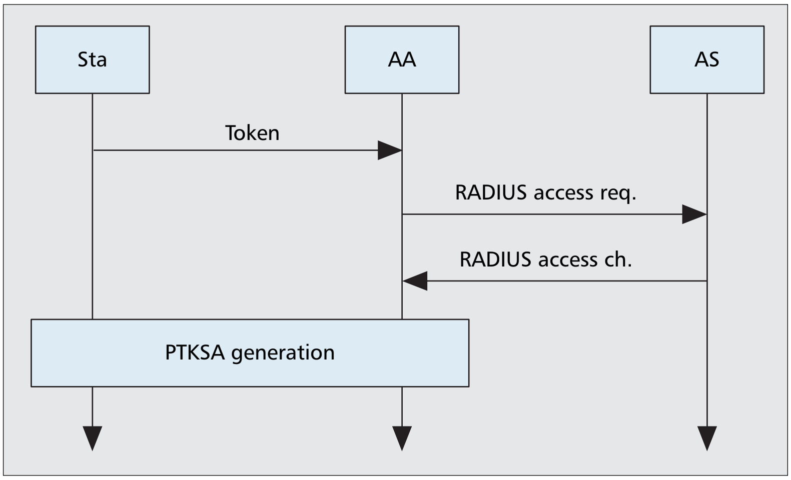
- As stated previously, before the authentication, synchronization packets are used. These carry unique identification numbers (such as sequence control in WiFi, frame number in WiMax, or the *challenge text* field in the standard authentication of WiFi). If the MAC provides enough material to securely feed the pseudo random number generator, that material could be used for the generation of the RANDOM field of the token.
- If the first policy is not possible, two simple modifications could be applied to the token generation. The token can include a counter V that is increased at each re-authentication and is kept synchronized between authentication server and supplicant, so that each re-authentication has a unique number and invalidates previously generated tokens. Moreover, any of the values used in the synchronization phase (information that is not long enough to be a good seed for random number generators can be long enough to dissuade preemptive generation of tokens) can be replied in the token to guarantee its freshness to the authenticator. We call this value S. The token could be generated as:

$token = [RANDOM, Au_{id}, V, S, H(Au_{id}, RANDOM, EMSK, V, S), EMSKID]$

thus, the *Au* will not forward a token that is not fresh, and the authentication server will check its authenticity.

Implementation

This re-authentication method was first implemented in an infrastructure environment, and results are reported in [13]. In this article, we illustrate the implementation of the fast re-authentication algorithm in a mesh access point (AP) environment, that is, a network composed of access points connected to each other in a mesh topology; each one serving a separate wireless LAN. In an infrastructure network, the results of the tests verified that there is a great advantage in reducing the number of packets exchanged compared to EAP-TLS (we measured a reduction of 89 percent of the time required for the EAP phase of the handover). With the extremely low delays introduced by the wired medium, this was due to two main factors: reduction of the total number of packets and use of a lightweight hash function. In a mesh network, we expect to have a longer total handover time, so that although the relative gain will be less (time due to computation will be lower compared to total handover time), we obtain a more realistic evaluation of absolute time. The protocol was tested on a real testbed composed of standard x86 processor terminals and a primsII wireless network interface card (NIC). All the nodes used a GNU/Linux operating system and hostAp driver and relative applications. For a RADIUS server, free RADIUS was chosen.



■ **Figure 4.** *Packet exchange sequence for fast re-authentication.*

In our experiments, we forced a client of the network to perform a re-authentication with another access point. The packets of the authentication were required to cross the whole backbone network across a multihop path (a total of three wireless hops and a wired hop) to reach the authentication server. We planned to have a comparison over 25 total re-authentications but due to the instability of the prototypal code for fast re-authentication, we were required to perform 50 attempts, 27 of which were successful. Note that the instability resided in the interaction with existent software, so fast re-authentication was not triggered for every handover, but it did not affect the performance of the algorithm itself.

In Table 1, we report the average inter-arrival time (IAT) and the total time measured from the client perspective over 50 re-authentications with EAP-TLS (odd lines refer to packets leaving the Sta, and even lines refer to packets reaching the Sta). The packets match the description of the protocol given in Fig. 3. From the client point of view, IAT between packets: (1:2), (2:3), (4:5), (6:7), (8:9) refers to packets traversing a one-hop link (between Au_1 and Sta) and the time required for calculating the response. The others (grayed out in the table) refer to packets crossing the whole network back and forth, and the time required by the AS to evaluate the information and forge a response.

The same values are reported for fast re-authentication in Table 2, measured over 27 re-authentications. The token was inserted into the EAP-Identity message making it 97 B larger.

When we compare the performance of the two methods, we see total time drops of 82.2

	Packet	IAT	Arrival time	Size
1	EAPOL Start	—	0.0000	36
2	EAP Identity Request	0.0023	0.0023	46
3	EAP Identity Response	0.2819	0.2842	51
4	EAP Request TLS	0.6122	0.8964	42
5	TLS Client Hello	0.3843	1.2807	142
6	TLS Server Hello	0.4499	1.7306	695
7	TLS Certificate	1.2161	2.9467	927
8	TLS server change C.S.	0.7516	3.6983	316
9	TLS server change C.S.	0.3729	4.0712	317
10	EAP Success	0.6805	4.7517	40

■ **Table 1.** *Measured interarrival time between packets of an EAP-TLS authentication.*

	Packet	IAT	Arrival Time	Size
1	EAPOL Start	—	0.0000	36
2	EAP Identity Request	0.002360	0.00236	46
3	EAP Identity Response	0.601722	0.60408	148
4	EAP Response, FA	0.239926	0.844008	41

■ **Table 2.** *Measured interarrival time between packets of a fast re-authentication.*

percent when using fast re-authentication. It is interesting to note that if we limit the comparison to the runs of the experiment that produced no layer II retransmissions (25 over 27 for fast re-authentication and 29 over 50 for EAP-TLS), we see a lower gain (average time is 0.62 for fast re-authentication versus 2.34 seconds for EAP-TLS, a difference of about 73.5 percent). This is close to what we expected. Because we reduced the total number of packets required by 75 percent, the value of 82.2 percent tells us that more packets imply higher probability of retransmissions and longer delays. Also, multiple retransmissions during the same re-authentication lead to delays of longer than 17 seconds (the maximum recorded for EAP-TLS is 17.4792 seconds and 3.6150 seconds for fast re-authentication). Without retransmissions, the range for average total time is (2.3660–2.6068) seconds for EAP-TLS and (0.6093–0.6956) seconds for fast re-authentication. The following additional factors can affect the results:

- If applied to a real MANET, mobility of mesh nodes could require higher set-up time for the routing protocol.
- Link level encryption of packets in the mesh backbone was disabled, but it could increase round trip time.
- No other traffic was present in the network, so there were few collisions (request to send/clear to send (RTS/CTS) were active).

This last point is quite important. In our opinion, a handshake made of only two packets is the shortest possible handshake to have in an authentication procedure. Because the network load was low and all the cryptography used in the fast re-authentication computationally lightweight, the performed measure represents a lower limit for any centralized authentication algorithm applied to a testbed with this topology and properties.

Referring to Table 1, it can be noted that the average IAT is higher for packets crossing a multihop path (grayed out lines), with the exception of line 7, due to the certificate verification used by TLS, which is computationally heavy and time consuming. In fast re-authentication, the absence of public/private key cryptography avoids this delay.

Conclusion

In this article we analyzed the security requirements of the handover procedure as a re-authentication algorithm in a hostile environment. We defined three guidelines that can help the designer of a re-authentication protocol in realizing a secure procedure — including against insider attackers — that always must be considered when the focus of the protocol is distributed networks in which terminals do not have secure trust relationships.

We also presented an implementation of a re-authentication algorithm based on the use of tokens, and we measured its performance in comparison with EAP-TLS protocol. The protocol is an application of the guidelines proposed and greatly reduced the number of required packets, not only decreasing the overall time required to complete the handshake but also the probability of retransmissions, thus improving general performance.

References

[1] IEEE Std., "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, Amendment 6: Medium Access Control (MAC) Security Enhancements," 2004.

[2] IEEE Std., "Local and Metropolitan Area Networks Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1," 2005.

[3] IEEE Std., "Local and Metropolitan Area Networks Port-Based Network Access Control," 2001.

[4] V. Gunasekaran and F. Harmantzis, "Affordable Infrastructure for Deploying WiMAX Systems: Mesh vs. Non Mesh," *IEEE VTC 2005-Spring*, vol. 5, 2005.

[5] B. Aboba and L. Blunk, "Extensible Authentication Protocol (EAP)," RFC 3748, 2004.

[6] B. Aboba and D. Simon, "PPP EAP TLS Authentication Protocol," RFC 2716, 1999.

[7] C. Rigney and A. Rubens, "Remote Authentication Dial in User Service (Radius)," RFC 2138, 1997.

[8] L. Maccari, R. Fantacci, and M. Paoli, "Security Analysis of IEEE 802.16," *ICC '07*.

[9] A. Mishra *et al.*, "Proactive Key Distribution Using Neighbor Graphs," *IEEE Wireless Commun.*, 2004.

[10] IETF Handover Keying Working Group, http://www.ietf.org/html.charters/hokey-charter.html

[11] Y. Hao *et al.*, "Security in Mobile Ad Hoc Networks: Challenges and Solutions," *IEEE Wireless Commun.*, vol. 11, 2004, pp. 38–47.

[12] C. Neuman and T. Yu, "The Kerberos Network Authentication Service (v. 5)," RFC 4120, July 2005.

[13] L. Maccari *et al.*, "Secure, Fast Handoff Techniques for 802.1x Based Wireless Network," *ICC '06*, 2006.

Biographies

Romano Fantacci (fantacci@lart.det.unifi.it) graduated from the Engineering School of the Université di Firenze, Italy with a degree in electronics in 1982. He received his Ph.D. degree in telecommunications in 1987. After joining the Dipartimento di Elettronica e Telecomunicazioni as an assistant professor, he was appointed associate professor in 1991 and full professor in 1999. His current research interests are digital communications, computer communications, queuing theory, satellite communication systems, wireless broadband communication networks, and ad hoc and sensor networks. He has been involved in several European Space Agency (ESA) and INTELSAT advanced research projects. He is the author of numerous articles published in communication science journals. He has guest edited special issues of IEEE journals and magazines, and served as symposium chair of several IEEE conferences, including VTC, ICC, and GLOBECOM. He received the IEE IERE Benefactor premium in 1990 and IEEE ComSoc Award for Distinguished Contributions to Satellite Communications in 2002. He is currently serving as an Editor for *Telecommunication Systems*, *International Journal of Communications Systems*, *IEEE Transactions on Communications*, and *IEEE Transactions on Wireless Communications*.

Leonardo Maccari (maccaripecorella@lart.det.unifi.it) graduated from the Engineering School of Florence University with a degree in computer engineering in 2004. He joined the Dipartimento di Elettronica e Telecomunicazioni in 2005 as a research fellow. His current research interests are security aspects of wireless telecommunications, with a special focus on mesh, sensor, and P2P networks.

Tommaso Pecorella received a Dr.Ing. degree in electronic engineering from the University of Firenze in 1996 and a Ph.D. degree in telecommunications engineering in 1999. Since 2001 he has been working as an adjunct professor at the University of Firenze. In 2000 he joined the CNIT — Italian University Consortium for Telecommunications as a scientific researcher. He is involved in a number of projects including ASI/CNIT ACE (real-time emulation of a generic OBP GEO satellite) and various COST actions. He is the author of articles published in communication science journals and delivered at international conferences. He was a member of the Technical Program Committee of ICC 2004 and VTC 2003.

Federico Frosali (federico.frosali@tilab.com) received a Laurea degree in electronic engineering in 1999 from the Université degli Studi di Firenze. Currently, he is a member of R&D staff at Selex Communications in the role of system engineer. His current research interests are in mobile ad hoc communications, broadband wireless communications, and information security. From 2000 to February 2006 he worked at Telecom Italia Laboratories (TILAB) in the IT Security Research Laboratory and was involved in several projects concerning Internet security. In 2004 he was a team leader for the TILAB ad hoc network security group and project leader for a Telecom Italia project in the field of wireless network security. He is author or co-author of two papers and four patents in the field of security of wireless mobile networks.