Secure Protocol for Fast Authentication in EAP-Based Wireless Networks

Rafa Marin, Santiago Zapata, and Antonio F. Gomez

Dept. de Ingeniería de la Información y las Comunicaciones,
New Faculty of Computer Science , Campus de Espinardo, Murcia, Spain
{rafa,canela,skarmeta}@dif.um.es

Abstract. In this paper we present a solution for providing a fast transition in heterogeneous mobile networks which involve network access control based on the *Extensible Authentication Protocol*. The goal is to reduce the time spent on providing access and smooth transition, between different technologies which require to perform authentication to allow network access. We propose and describe an architecture and secure protocol, which reduce the number of round trips during authentication phase, and verify its secure properties with a formal tool.

Keywords: Fast handover, security, network access control, authentication.

1 Introduction

Nowadays, authentication and authorization to control access to the infrastructure is one of the key elements in which mobile operators are interested in. The goal is to guarantee that only authenticated nodes are allowed to communicate with external hosts in both directions. Traditionally, this problem has been solved in fixed IP networks through the deployment of *Authentication*, *Authorization* and *Accounting* (AAA) infrastructures [1]. However, the authentication and network control access are time-consuming processes which can last several hundreds of milliseconds with the corresponding high delays and packet losses in on-going communications. This is in contrast with service providers' desire of provisioning a fast seamless mobility. Thus, there is an increasing demand for studying a solution which achieves the goal of reducing the impact of authentication and network access control in mobile users.

In particular, authentication in wireless networks is typically based on the *Extensible Authentication Protocol* (EAP) [2] which provides a flexible way to perform authentication through the so-called *EAP authentication methods*. These EAP methods usually generate cryptographic material after a successful authentication. However, the EAP method execution is also a time-consuming process, as involves several round trips between two entities, the EAP peer and the EAP server. This latency specially appears in roaming scenarios where the EAP peer may be far from the EAP server and where each round trip may last hundred of milliseconds. Besides, each time the user accesses a new point of attachment, an EAP authentication is performed even if the user owns unexpired cryptographic material from the previous EAP authentication. In this way, research and standardization community have tried to reduce that time through different approaches. For example, context transfer based mechanisms [3], [4] where the cryptographic state in the current point of attachment (where the peer ended up the authentication process) is transferred to a new point of attachment. In that way, when the peer moves to the new authenticator, it is not required to perform a full EAP authentication. However, context transfer has raised some security issues in the community [5] which has lead to alternative solutions. In particular, some research work has proposed to modify the EAP stack [6] to the cost of updating the existing EAP implementations. Other solutions have designed new EAP methods such as [7] which avoids any modification in the existing EAP deployments but which adds additional round trips when providing a fast network access during handover. Finally, Kim et al. [8] have designed their own secure protocol between the mobile and the authenticator avoiding to run EAP. The proposal implicitly implies, however, a modification at link-layer level, in particular IEEE 802.11i [9] where the protocol is applied.

Our proposed solution is also based on a secure protocol but works on top of IP being independent of the underlying technology. Moreover, it is able to leverage the cryptographic material generated during an initial EAP authentication to bootstrap new security associations with new authenticators without performing EAP. Therefore, our secure protocol provides a quick authentication and network access control. In the authors' opinion, the main contributions of this paper are: the definition of a secure protocol which reduces the number of round trips used for authenticating the user during roaming and is technology independent; the definition of a proper key hierarchy that enables further optimization during handover and a demonstration how our protocol achieves secure properties, through the use of a model checker as a proper formal tool.

The remainder of the paper is organized as follows: in section 2 we analyze EAP since it is the basis where our solution stems. Section 3 describes the protocol and the defined key hierarchy to support a fast secure handover. In section 4 we analyze the security aspects of our protocol and show by using a formal tool how it meets certain security properties. Section 5 compares our alternative with other proposals in the literature. Finally, section 6 concludes the paper and provides some future directions.

2 EAP Key Management Framework

The *Extensible Authentication Protocol* (EAP) has been designed to permit different kind of authentication mechanisms through the so-called *EAP methods*. The EAP methods are run between an EAP peer (the mobile node) and an EAP server (typically co-located with the AAA server) through an EAP authenticator which simply forwards EAP packets back and forth between the EAP peer and the EAP server in order to complete the authentication process. On the one hand, an EAP lower-layer is used to transport the EAP packets between the

EAP peer and EAP authenticator. On the other hand, an AAA protocol, such a RADIUS [10] or Diameter [11], is used for the same purpose between the EAP authenticator and the EAP server.

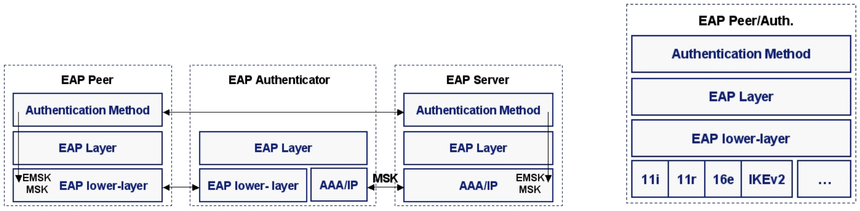
The EAP methods not only provide authentication, but also are able to generate keying material to establish a security association between the EAP peer and the EAP authenticator through a *security association protocol*. The key material, exported from the EAP methods and described in the EAP Key Management Framework [12], is mainly composed by the *Master Session Key* (MSK) and the *Extended Master Session Key* (EMSK). As depicted in Fig. 1(a), both keys are exported to the EAP lower-layer in the EAP peer side and to the AAA protocol in the EAP server side. The MSK is sent out to the EAP authenticator from the AAA server (where the EAP server is co-located) and used for establishing a security association between the EAP peer and the EAP authenticator.

Unlike MSK, the EMSK must not be provided to any other entity outside the EAP server or peer, so this key will not be sent to the EAP authenticator. The EAP server may well hold and use the EMSK for further key derivation. Recent work [13] has shed some light about how to use EMSK in order to derive further keys for different purposes. For backward compatibility, the EMSK has been intended to work as the root key in a key hierarchy applicable to a fast handover solution. Therefore, we will also use EMSK as a root key for our own key hierarchy.

3 Secure Protocol for Fast Network Access

The EAP Key Management Framework mandates that the keys generated during an EAP authentication are exported to the EAP lower-layer in the EAP peer. The implication is the EAP lower-layer is in charge of handling the generated keys in order to establish security associations between the peer and the authenticator. Therefore, when the EAP lower-layer for a specific technology (e.g. 802.11i) receives keys from the EAP methods after a successful EAP authentication, it can use them only within that particular technology. This is reasonable since that specific EAP lower-layer knows the details of that particular technology. Thus, the keys provided to that EAP lower-layer can be only used for that technology and no other. Unfortunately, this makes no possible, to leverage the keys generated during an initial EAP authentication for other technologies since the specific EAP lower-layer does not know how to provide cryptographic material to other EAP lower-layers.

However, let us consider the model presented in the Fig. 1(b) where the EAP lower-layer is independent of the underlying technology. In this case, a single EAP lower-layer receives the MSK and EMSK after a successful EAP authentication. The EAP lower-layer can use those keys for generating a security association at the EAP lower-layer itself but additionally can distribute additional keys derived from EMSK and/or MSK to bootstrap security at other different technologies. These technologies are named *ports* and shall require a symmetric key to perform a security association protocol (SAP) (at port level) in order



(a) EAP Key Mng. Fwk. Model (b) Single EAP lower-layer

Fig. 1. EAP Key Management models

to protect data traffic between EAP peer and authenticator. As an examples of ports, we may find, at network-layer, IKEv2 [14] which is used for establishing an IPsec tunnel to protect data traffic between peer and authenticator. Similar examples can be found at link-layer, such as IEEE 802.11i [9] , which implements a *Pre-Shared Key* (PSK) mode that allows to establish a security association through a *4-way handshake protocol* by using a symmetric key (PSK). In this way, the single EAP lower-layer concentrates the key management between the EAP peer and EAP authenticator and allows to bootstrap security to different (heterogeneous) technologies (ports).

Additionally, we have the important aim of reducing the number of round trips dedicated to provide authentication and access control each time the EAP peer switches the EAP authenticator during a handover. In order to achieve this objective, we propose that the EAP lower-layer itself leverages keys generated during initial EAP authentication (MSK and EMSK) to mutual authenticate and generate session keys between the peer and the authenticator, in only one trip and without the need of running EAP again.

Indeed, when the EAP peer moves to a new EAP authenticator both entities must engage a mutual authentication process and derive, as a result, keys to protect EAP lower-layer messages and data between them. Therefore, we require to run an authentication and key establishment protocol that initially involves two parties: the EAP peer and the EAP authenticator. Typically, the new EAP authenticator shall not own any key associated to that particular EAP peer. However, as pointed in section 2, the EAP server involved during the initial EAP authentication shall hold the EMSK (or a key derived from it). As a result, the EAP authenticator may contact the AAA server (where the EAP server is co-located) in order to fetch (*pull method*) some cryptographic material to authenticate the user and derive new session keys. This cryptographic material is generated from the EMSK and the key hierarchy we have designed is detailed in section 3.3.

Thus, this case involves 3 parties: EAP peer, EAP authenticator and the EAP server (distributing keys). This implies that well-known 3-party key distribution protocols [15] may be well considered in this case. However, the features we want to provide in our architecture does not completely fit with typical 3-party key distribution protocols for several reasons. For example, the server does not need to distribute any key to the EAP peer, since it can derive the session keys from those ones generated during initial EAP authentication (MSK,EMSK). In this manner, the server only needs to distribute keys to the EAP authenticator. Additionally, we desire an optimization consisting on the AAA server pre-installing (*push method*) a few keys in different authenticators even before the EAP peer is associated to it [16]. In such a case, the EAP authenticator already owns the required keys to authenticate and establish session keys with the EAP peer. An important design principle is that EAP peer shall not need to know whether the required keys were pre-installed at the authenticator or, on the contrary, they still reside at the AAA server.

For these reasons, we propose that the single EAP lower-layer uses and implements a modified version of a well proved and secure two-party protocol, the REKEY protocol [17], by using the keys generated during the initial EAP authentication, but considering that there may be a server (the AAA server) in the backend, which is in charge of distributing certain keys. In this way, if the EAP authenticator already owns a share secret with EAP peer, the REKEY protocol is immediately run. However if the EAP authenticator does not have any keys for the EAP peer, it will request and fetch them from the AAA server in order to complete the authentication process.

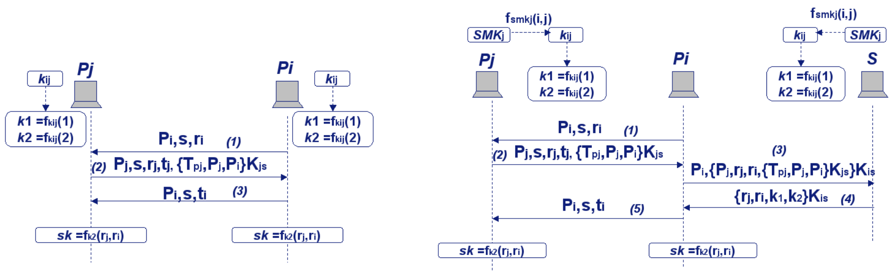
3.1 The REKEY Protocol

The REKEY protocol belongs to the family of provable secure protocol [18] which have been verified to be secure under a complexity-theoretic proof. It is very similar to AKEP2 presented by Bellare at el in [18] and who introduced the first mathematical proof that the protocol was secure. It accomplishes the minimum number of messages (3) for authentication and key establishment when random numbers (*nonces*) are used for freshness. As depicted in Fig. 2(a), two parties P_i and P_j engage a mutual authentication process based on a shared secret k_{ij}. From that key, they both derive two keys $k_1 = f_{k1}(1)$ and $k_2 = f_{k2}(2)$ through a secure Message Authentication Code (MAC) function f. Whereas k_1 is used for party authentication, the second one (k_2) works as root key for session key sk derivation. The distinction made between the key for authentication (k_1) and the key for key derivation (k_2) is due to that, when using the session key also for authentication, it does not accomplish the security proof in [18].

In REKEY protocol, P_i starts sending an initial message (1) with its identity (P_i), a value s which identifies that particular session and a random number r_i. P_j answers (2) with its identity (P_j), the same session identifier s, its own random value r_j and an authentication tag t_j which is result of $f_{k1}(r_j, s, r_i)$. Upon receiving this message, P_i verifies the authentication tag t_j with its key k_1 and sends a new message (3) that includes s and an authentication tag t_i generated as $f_{k1}(r_i, s, r_j)$. Party P_j can verify t_i with k_1 completing the mutual authentication process (if the tag is successfully verified). After a successful mutual authentication, P_j and P_i can generate session keys by using k_2 as root key for key derivation. In particular, the session key sk is generated applying f to both random values r_i,r_j: $sk = f_{k2}(r_j, r_i)$.

3.2 The REKEY Protocol with a Server

We propose an extension of the REKEY protocol which considers the possible participation of a server in the backend, in charge of distributing keys. In our particular case, the AAA server where the initial EAP authentication was performed, acts as this entity. As we will see later, this AAA server may delegate this function to other AAA servers for further optimization. A simplified version showing the most important parameters of the modified REKEY protocol is outlined in Fig. 2(b). For sake of clarification, $\{X\}_k$ represents the encryption of message X with key k providing integrity and confidentiality.



(a) REKEY protocol (b) REKEY with an AAA server

Fig. 2. The modified REKEY protocol

The party P_j shall represent the EAP peer and the party P_i refers to the EAP authenticator, which is in contact with an AAA server. After some initial event, (e.g. link-layer signal or some message sent from the EAP peer), P_i starts the REKEY protocol, shown in section 3.1, sending the message 1. The EAP peer (P_j) answers with message 2 but now including a token $\{T_{pj}, P_j, P_j\}_{Kjs}$ where K_{js} is a shared secret between P_j and the server S and T_{pj} is a timestamp sent by P_j for freshness. It is worth nothing though, it may be replaced for a sequence number *seq* managed between the EAP peer and the EAP server for the same freshness purpose.

At this point, the authenticator P_i may not have the key k_1 required for verifying the authentication tag t_j. In such a case, the authenticator forwards (3) the token $\{T_{pj}, P_j, P_j\}_{Kjs}$ to the server as well as the random values r_i, r_j. The token and these random numbers are protected with a key K_{is} shared between authenticator i and the server, which defines the security association established between both entities. Then, the token can be verified by the server, which shares the key K_{js} with the P_j. It can also verify that it is fresh thanks to the timestamp or sequence number contained in this token. In this way, the server can verify that EAP peer is really requesting the key distribution process. This is important since it avoids a denial-of-service attack where an attacker, which has taken control of an authenticator, continuously requests keys for other authenticators and peers even when they are not required. So, only upon a proper token verification, the server sends back (4) the key k_1 for authentication and

k_2 for further key derivation. Alternatively, the server can distribute the k_{ij} instead of both k_1, k_2. The difference is that the authenticator will be in charge of deriving both keys from k_{ij}.

It is possible to obtain further optimization during handover, by delegating to another AAA server located near to the access device, the key distribution process [19]. In this way, the delay between the authenticator and server is further reduced and overall access latency is decreased. In this optimization, the key k_{ij} is, in fact, derived by another intermediate server S_v from a root key SMK_j^v provided by the home AAA server (AAAh). We have also considered this optimization by designing a key hierarchy which includes the possibility of a trusted intermediate entity (usually the AAA in the visited domain) carries out authentication tasks and key distribution in a particular domain.

3.3 Key Hierarchy and Key Derivation

The key hierarchy stems from the EMSK generated during initial EAP authentication. We have followed the recommendations in [13] in order to derive the whole key hierarchy. In general, the reference [13] explains a general key derivation framework based on a Key Derivation Function (KDF) which derives further keys from the EMSK (root key). These keys are named *User Specific Root Keys* (USRKs) and derived using the following general way:

$$USRK = KDF(rootkey, keylabel, optionaldata, length) \qquad (1)$$

where *root key* is the EMSK. This derivation also includes a *key label*, *optional data*, and output *length*. The KDF is expected to give the same output for the same input. By default this KDF is taken from the Pseudo Random Function+ (PRF+) key expansion defined in [14], being HMAC_SHA_256 [20] the default PRF. We have used this general framework to build our key hierarchy by replacing the *root key* for different keys in the hierarchy.

The Table 1 shows the different parameters which have been considered in formula 1 in order to derive the key hierarchy. We have indexed each key with index j,i and v to refer that key is associated to the *j-th* EAP peer and *i-th* EAP authenticator and generated by the *v-th* server, respectively.

Table 1. Parameters for the Key Hierarchy Derivation

Key Deriv.	**Root Key**	**Key Label**	**Opt. Data**	**Length**
RMK_j^v	$EMSK_j^v$	"root_master_key@domain"	–	64
SMK_j^v	RMK_j^v	"server_master_key@domain"	v	64
k_{ij}^v	SMK_j^v	"authenticator_master_key@domain"	$P_i\|P_j$	64
k_1^{ij}	k_{ij}^v	"authentication_key@domain"	1	32
k_2^{ij}	k_{ij}^v	"key_derivation_key@domain"	2	32
IK_{ij}	k_2^{ij}	"integrity_key@domain"	$r_i\|r_j$	32
EK_{ij}	k_2^{ij}	"encryption_key@domain"	$r_i\|r_j$	32
BK_{ij}	k_2^{ij}	*bk-label*	$r_i\|r_j$	*length*

The Root Master Key RMK_j^v is an USRK derived from the $EMSK_j^v$ and associated to the *j-th* EAP peer and generated by both the EAP peer and the EAP server. In this way, both entities do not need to hold the EMSK anymore as [13] recommends.

The Server Master Key SMK_j^v is derived by the home AAA server from the RMK_j^v for a specific AAA server v where the home AAA server (AAAh) may delegate key distribution and management tasks. It is assumed that the SMK_j^v is transported to the AAA server v through a proper security association which avoids the key to be revealed. It is worth noting that the AAAh also derives its own SMK for its own use (SMK_j^{home}). This gives certain symmetry and hygiene to the key hierarchy. Finally, as reflected in Table 1, the AAA server's identifier (v) is required for SMK derivation. This value may be an IP address or the AAA server's Fully Qualified Domain Name (FQDN) and is used to bind the AAA's identity to the SMK_j^v.

The Authenticator Master Key k_{ij}^v is derived by AAA server v from the SMK_j^v for the *i-th* EAP authenticator and *j-th* EAP peer and may be transported to the authenticator. It is used as root key by both the peer and the authenticator in order to derive further keys that allow a mutual authentication (k_1^{ij}) between each other and generate a key (k_2^{ij}) for further session key derivation.

The Authentication Key k_1^{ij} is exclusively used for authenticating the peer and the authenticator and it is derived from the authenticator master key k_{ij}^v. It may be transported to the authenticator together with k_2^{ij} instead of k_{ij}^v.

The Key Derivation Key k_2^{ij} is used as root key in order to derive session keys that permit to establish an authenticated channel between the peer and the authenticator. This key is used to derive a set of three keys: the *Integrity Key* (IK_{ij}) and the *Encryption Key* (EK_{ij}) used to provide integrity and encryption at EAP lower-layer level, respectively; and the *Bootstrapping Key* BK_{ij} used as a shared secret in a security association protocol run (such as IKEv2 or 4-way handshake in IEEE 802.11i) intended to protect data traffic between the peer and authenticator.

3.4 Process Outline

The complete process is based on two main phases. Firstly, in a *bootstrapping phase*, the EAP peer engages a full EAP authentication with the authenticator by using the single EAP lower-layer. After a successful EAP authentication, EAP peer and EAP server derive a key hierarchy explained in the previous section 3.3. Additionally, in this phase the EAP peer may be provided with a value that allows to loosely synchronize the clocks of both the EAP peer and the EAP server (when timestamps are used for freshness). Alternatively, if sequence number is used for freshness, a random sequence number (*seq*) can be provided to the EAP peer from the EAP server in this phase. Finally, the bootstrapping phase is also needed to know what specific AAA server v will be in charge of key distribution and management during user roaming.

After the bootstrapping phase, which happens only once during the initial EAP authentication, the peer may start a *handover phase* and move to a new EAP authenticator. In order to get access and avoid another EAP authentication, both entities (EAP peer and EAP authenticator) engage a mutual authentication process based on our modified REKEY protocol and derive, as a result, keys to protect EAP lower-layer messages and data between them. This mutual authentication process is based on the key k_1^{ij} derived from key hierarchy stemmed from EMSK. After a successful mutual authentication, the session keys IK_{ij}, EK_{ij} and BK_{ij} are derived from k_2^{ij}.

4 Security Details

The REKEY protocol with a backend server outlined in section 3.2 has been checked against the *Automated Validation of Internet Security Protocols and Applications* (AVISPA) [21] tool which allows, through the *High Level Protocol Specification Language (HLPSL)*, to specify a protocol in order to find possible attacks. It uses several model checkers which analyze possible protocol behaviors and allow to know if they accomplish certain correctness conditions or goals.

We have specified our protocol in the formal language HLPSL and check it against AVISPA. The Fig. 3 shows HLPSL specification per each party involved (the peer, the authenticator and the server) and the authentication requirements or *goals*. For simplicity, we have only included the server that distributes k_1^{ij} and k_2^{ij}.

Although no attacks have been found in the protocol, it is important to make some additional comments related with the security properties in our modelled protocol. If we take a look at section 3.2, we shall realize that the same keys k_1^{ij} and k_2^{ij} are transported to the same *i-th* authenticator for the *j-th* peer, as long as the EAP authentication lifetime is valid. In other words, the server will distribute the same couple of keys k_1^{ij} and k_2^{ij} during the EMSK lifetime and until the next full EAP re-authentication refreshes the whole key hierarchy outlined in section 3.3. This feature allows the authenticator to cache, at most, k_1^{ij} and k_2^{ij} during EMSK lifetime. During this time the EAP authenticator will not contact the server to fetch any keys, saving time to provide access. However, it implies that same k_1^{ij} and k_2^{ij} may be used in different sessions. It does not mean though, that the derived keys (IK_{ij}, EK_{ij}, BK_{ij}) are the same, since their derivation depends on random values generated per session. However, if both k_1^{ij} and k_2^{ij} are revealed, past and present sessions can be compromised. That is, forward secrecy is not provided. Additionally, the attacker can impersonate the authenticator or peer during k_1^{ij} and k_2^{ij} lifetime. The impact of compromising these keys is, fortunately, limited to the particular peer j and authenticator i. Therefore, sessions bound to different peers and same authenticator i are not compromised. It is interesting to mention that this case is also accepted in the EAP keying framework, where the MSK might be cached and used through different sessions during EAP authentication lifetime.

```
role peer (
      Pj,Pi,S : agent,
      F,KDF : hash_func,
      K1,K2,Kjs : symmetric_key,
      T : text,
      SND,RCV : channel (dy))
played_by Pj def=
local
      Rj,Ri,Sd,Tpj : text,
      State : nat,
      Tag_j,Tag_i : {hash(agent.text.text.text)}_symmetric_key,
      TokenAS : {text.agent.agent}_symmetric_key
const
      sec_k2_j,sec_k1_j,sec_sk_j,tag_j,tag_i,sd,ri,rj : protocol_id
init
      State := 1
transition
0.    State= 1 ∧ RCV(Pi.Sd'.Ri') =|>
      State':= 2 ∧ Rj':=new()
                 ∧ Tpj':=T
                 ∧ Tag_j':={F(Pi.Ri'.Sd'.Rj)}_K1
                 ∧ TokenAS':={Tpj'.Pj.Pi}_Kis
                 ∧ SND(Pi.Ri'.Sd'.TokenAS'.Tag_j')
                 ∧ witness(Pj,S,tpj,Tpj')
                 ∧ witness(Pj,Pi,tag_j,Tag_j')
                 ∧ witness(Pj,Pi,rj,Rj')
1.    State=2 ∧ RCV(Pi.Sd'.Tag_i')
                 ∧ Sd=Sd'
                 ∧ Tag_i'={F(Pi.Ri.Sd'.Rj)}_K1 =|>
      State':=7 ∧ secret(K2,sec_k2_j,{Pj,Pi,S})
                 ∧ secret(K1,sec_k1_j,{Pj,Pi,S})
                 ∧ request(Pj,Pi,tag_i,Tag_i')
                 ∧ request(Pj,Pi,sd,Sd)
                 ∧ request(Pj,Pi,ri,Ri)
end role
```

```
role authenticator (
      Pi,Pj,S : agent,
      F,KDF : hash_func,
      Kis : symmetric_key,
      SND,RCV : channel (dy))
played_by Pi def=
local
      K1,K2 : symmetric_key,
      Tag_j,Tag_i : {hash(agent.text.text.text)}_symmetric_key,
      TokenAS : {text.agent.agent}_symmetric_key,
      Rj,Ri,Sd : text,
      State : nat
const
      tag_i,tag_j,rj,sd,ri,k1_s,k2_s: protocol_id
init
      State:=0
transition
0.    State=0 ∧ RCV(start) =|>
      State':=1 ∧ Ri':=new()
                ∧ Sd':=new()
                ∧ SND(Pi.Sd'.Ri')
                ∧ witness(Pi,Pj,ri,Ri')
                ∧ witness(Pi,Pj,sd,Sd')
1.    State=1 ∧ RCV(Pj.Sd'.Rj'.Tag_j'.TokenAS')
                ∧ Sd=Sd' =|>
      State':=3 ∧ SND(Pi.{Pj.Rj'.Ri.TokenAS'}_Kis)
2.    State=3 ∧ RCV({Rj'.Ri'.K1'.K2'}_Kis)
                ∧ Tag_j={F(Pj.Rj'.Sd'.Ri')}_K1' =|>
      State':=6 ∧ Tag_i':={F(Pi.Ri.Sd.Rj)}_K1'
                ∧ SND(Pi.Sd.Tag_i')
                ∧ witness(Pi,Pj,tag_j,Tag_i')
                ∧ request(Pi,Pj,tag_i,Tag_j)
                ∧ request(Pi,Pj,rj,Rj)
                ∧ request(Pi,S,k1_s,K1')
                ∧ request(Pi,S,k2_s,K2')
end role
```

```
role server (
      S,Pi,Pj : agent,
      F,KDF : hash_func,
      K1,K2,Kis,Kjs : symmetric_key,
      T : text,
      SND,RCV : channel (dy))
played_by S def=
local
      Ri,Rj,Sd,Tpj : text,
      State : nat,
      TokenAS : {text.agent.agent}_symmetric_key
const
      sec_k1_s,sec_k2_s,k1_s,k2_s: protocol_id,
init
      State:=0
transition
0.    State=0 ∧ RCV(Pi.{Pj.Rj'.Ri'.{Tpj'.Pj.Pi}_Kjs}_Kis)
                 ∧ Tpj'=T =|>
      State':=1 ∧ SND({Rj'.Ri'.K1.K2}_Kis)
                 ∧ secret(K1,sec_k1_s,{Pj,Pi,S})
                 ∧ secret(K2,sec_k2_s,{Pj,Pi,S})
                 ∧ witness(S,Pi,k1_s,K1)
                 ∧ witness(S,Pi,k2_s,K2)
                 ∧ request(S,Pj,tpj,Tpj')
end role
```

```
goal
      %Peer authenticates Authenticator on tag_i
      authentication_on tag_i
      %Peer authenticates Authenticator on tag_j
      authentication_on tag_j
      %Server authenticates Peer on sd
      authentication_on sd
      %Peer authenticates Server on rj
      authentication_on rj
      %Peer authenticates Server on tpj
      authentication_on tpj
      %Peer authenticates Authenticator on rj
      authentication_on ri
      %Server authenticates Authenticator on k1 and k2.
      authentication_on k1_s
      authentication_on k2_s
      %k1 and k2 remains secret.
      secrecy_of sec_k1_j,sec_k2_j,sec_k1_s, sec_k2_s
end goal
```

Fig. 3. HLPSL specification of REKEY protocol with an AAA server

5 Comparison with Other Proposals

We have made a comparative analysis between existing proposals which reduce the handover latency and our proposal. We have mainly compared the number of round trips involved between the authenticator and the server (which is, actually, the bottleneck in the re-authentication process) under the consideration of the impact on existing EAP deployments. Additionally, we have analyzed the capability of allowing inter-technology handover (commonly named vertical handover) and the level of security provided by the different alternatives. In particular, we have analyzed and compared EAP-ER [6], EAP-EXT [7], and references [4], [8] against our proposal. The Table 2 shows a summary of the comparison.

Table 2. Comparative Table of Different Proposals

Proposal	**Round trips**	**Impact**	**Hand. Inter-Tech**	**Security Level**
EAP-EXT	2 or more	-	No	Medium
EAP-ER	1	High	No	Medium
Aura et al	0	High	No	Low
Kim et al	2	High	No	Medium-High
REKEY with Serv.	1	Low	Yes	High

In case of EAP-ER, the proposal reduces the number of round trips between the authenticator and the AAA server to only one in order to recover a new key (*rMSK*). However, it assumes modifications in the original EAP state machine at the peer, the authenticator and the server. It may create some deployments issues, mainly in the case of the existing EAP authenticators whose firmware should be updated. Alternatively, EAP-EXT defines a new EAP method which is able to transport any other EAP method within, in order to perform the EAP authentication and leverage the MSK generated for the inner EAP method to create a security association at EAP-EXT level. EAP-EXT allows to include some extra-functionality such as fast re-authentication. However, this fast re-authentication process takes two or more round trips between the authenticator and the server. As advantage, it does not need to modify any existing EAP deployment with the cost of additional round trips.

Commonly, solutions based on EAP are intended to work directly on specific link-layer technologies. This however makes more complex the inter-technology handover and it does not help to support certain handover optimizations such as pre-authentication [22].

The solution presented by Aura et al. [4] provides a trade-off between security and fast handover in 802.11 networks. It allows certain data traffic, with a restricted quality of service, to pass through an access point after performing a fast and weak authentication process. After that, a strong authentication process is carried out to enable data traffic with unrestricted quality of service. On the one hand, for the weak authentication process, the solution does not need to contact with the server at all and therefore the number of round trips between authenticator and server is 0. On the other hand, the strong authentication requires to contact the server, e.g. with a full EAP authentication. Apart from the solution mandates some modification at link layer, the weak authentication process violates the principles established in [5] as certain cryptographic material from the previous access point are transferred (through the mobile node) to the new access point.

Another relevant solution is that presented by Kim et al. in [8] which implements a secure protocol verified in BAN logic [23]. The secure protocol implementation is applied to 802.11i networks where modification is required to support the protocol. It also defines a key hierarchy based on a pre-shared key *PK* between the station (EAP peer) and the AAA server. However, no relationship is established between this key hierarchy and keys generated during the initial EAP authentication. Finally, the solution considers context transfer between different domains without contacting the home domain. To achieve this optimization, it is assumed a business agreement between the visited domains involved in the roaming. However, this may not be always true. In fact, a user may roam between two visited domains that, though they do not have a direct business relationship between them, they have both an agreement with the home domain.

In contrast, our solution embraces the advantages of several of these solutions, but adding new improvements. In fact, our secure protocol is based on a well-known provable secure protocol [17] with the inclusion of a server in the backend, which reduces to only one round trip the communications between the authenticator and the server during authentication process as EAP-ER does, but it does not imply any modification in the EAP implementations on existing deployments as EAP-EXT provides. Moreover, although both EAP-ER and EAP-EXT are independent of the EAP lower-layer, it does not allow fast re-authentication between different technologies. As our secure protocol is conceived to be transported over IP, it is independent of the underlying technology. Therefore, this kind of inter-technology handover is possible from the design. Furthermore, our key hierarchy in section 3.3 has been designed for this purpose by defining the bootstrapping key BK, specific for each technology. Finally it provides the additional benefit of avoiding modification at link layer level and existing standards. We require, however, that the ports accept the installation of a pre-shared key which is used for a security association protocol. This is not a strong requirement since there are already some technologies that have the option to start a security association protocol by using a pre-shared key. Therefore, future technologies could easily include a mode that could be fed with a random and dynamically generated key. In terms of performance, the relevant and key point is the number of round trips between the authenticator and the server. For example in an inter-continental communication we have found a mean value of 150 ms per each round trip. Reducing the number of round trips the performance improves. That is the reason that EAP-ER and our solution provides a similar optimization (only round trip). However, as we have mentioned, our solution provide a more neat solution since it does not require EAP modification at all.

6 Conclusion and Future Work

We study the issue of efficient access control in wireless network, which is of paramount importance in many of the operator-oriented applications of these networks. We evaluate the application of traditional authentication schemes based on EAP and show that they can limit very much the overall performance of the system when mobile nodes change their point of attachment to fixed networks. The reason is that an authentication process can take up to a couple of seconds, which means that data traffic may be lost until the authentication with the new authenticator is completed.

Our proposal is the use of a single EAP lower-layer working on top of IP, which leverages the keys generated during an initial EAP authentication, in order to authenticate and derive new session keys with new authenticators. This reduces the number of round trips with the home AAA server, by avoiding to run a complete EAP authentication during initial EAP authentication lifetime. To achieve this objective, we have designed and verified with the formal tool AVISPA, a secure protocol which is integrated in the EAP lower-layer. It is based on a well proved secure two-party protocol but including a server in charge of distributing and managing keys. Our comparison with existing proposals demonstrates that we obtain a minimum number of rounds trips with strong security properties. Furthermore, the improvement can be applied to different technologies. In fact, this scheme finds a good trade-off between the benefit of reducing of the number of round trips, suitable security properties and the impact on existing EAP deployments, in terms of modification or re-design on existing devices.

Acknowledgments. This work has been partially funded by the ENABLE EU IST project (IST2005-027002) and DAIDALOS EU IST project (FP6-IST026943).

References

1. Marin, R., Martinez, G., Gomez, A.: Evaluation of AAA Infrastructure Deployment in Euro6ix IPv6 Network Project. In: Applied Cryptography and Network Security 2004, Technical Track Proceedings, pp. 325-334. Yellow Mountain, China (June 8-11, 2004)
2. Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., Levkowetz, H.: Extensible Authentication Protocol (EAP). RFC 3748 (June 2004)
3. Georgiades, M., Akhtar, N., Politis, C., Tafazolli, R.: AAA Context Transfer for Seamless and Secure Multimedia Services. In: 5th European Wireless Conference (EW'04), Barcelona, Spain (February 2004)
4. Aura, T., Roe, M.: Reducing Reauthentication Delay in Wireless Networks. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks SECURECOMM'05, Athens, Greece, pp. 139–148 (September 2005)
5. Housley, R., Aboba, B.: Guidance for AAA Key Management. draft-housley-aaa-key-mgmt-06, IETF Internet Draft, Work in Progress (November 2006)
6. Narayanan, V., Dondeti, L.: EAP Extensions for Efficient Re-authentication draft-vidya-eap-er-02, IETF Internet Draft, Work in Progress (January 2007)
7. Ohba, Y., Das, S., Marin, R.: An EAP Method for EAP Extension (EAP-EXT). draft-ohba-hokey-emu-eap-ext-01, IETF Internet Draft, Work in Progress (March 2007)
8. Kim, H., Shin, K.G., Dabbous, W.: Improving Cross-domain Authentication over Wireless Local Area Networks. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks SECURECOMM'05, pp. 127-138, Athens, Greece (September 2005)
9. I. of Electrical and E. Engineer: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Specification for Enhanced Security IEEE 802.11i, IEEE std. (July 2005)
10. Aboba, B., Calhoun, P.: RADIUS support for EAP. RFC 3579 (June 2003)
11. Eronen, P., Hiller, T., Zorn, G.: Diameter Extensible Authentication Protocol (EAP) Application. RFC 4072 (August 2005)
12. Aboba, B., Simon, D., Arkko, J., Eronen, P., Levkowetz, H.: Extensible Authentication Protocol (EAP) Key Management Framework. draft-ietf-eap-keying-15.txt, IETF Internet Draft (October 2006)

13. Salowey, J., Dondeti, L., Narayanan, V., Nakhjiri, M.: Specification for the Derivation of Usage Specific Root Keys (USRK) from an Extended Master Session Key (EMSK). draft-ietf-hokey-emsk-hierarchy-00.txt, IETF Internet Draft (January 2007)
14. Kauffman, C.: Internet Key Exchange (IKEv2) Protocol. RFC 4306 (December 2005)
15. Harskin, D., Ohba, Y., Nakhjiri, M., Marin, R.: Problem Statement and Requirements on a 3-Party Key Distribution Protocol for Handover Keying. draft-ohba-hokey-3party-keydist-ps-01, IETF Internet Draft, Work in Progress (March 2007)
16. Mishra, A., Shin, M., Petroni, N., Clancy, C., Arbaugh, W.: Proactive Key Distribution Using Neighbor Graphs. IEEE Wireless Communication 11(1), 26–36 (2004)
17. Canetti, R., Krawczyk, H.: Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 453. Springer, Heidelberg (2001)
18. Bellare, M., Rogaway, P.: Entity Authentication and Key Distribution. In: Stinson, D.R. (ed.) CRYPTO 1993. LNCS, vol. 773, pp. 110–125. Springer, Heidelberg (1994)
19. Marin, R., Bournelle, J., Maknavicius-Laurent, M., Combes, J.M., Gomez Skarmeta, A.F.: Improved EAP keying framework for a secure mobility access service. In: International Conference On Communications And Mobile Computing, Vancouver, British Columbia, Canada, pp. 183–188 (March 2006)
20. National Institute of Standards and Technology: Secure Hash Standard, FIPS 180-2, August 2002. With Change Notice 1 dated (February 2004)
21. Automated Validation of Internet Security Protocols and Applications (AVISPA): IST Project 2001-39252 `http://www.avispa-project.org/`
22. Dutta, A., Zhang, T., Ohba, Y., Taniuchi, K., Schulzrinne, H.: MPA assisted Optimized Proactive Handoff Scheme. ACM Mobiquitous (2005)
23. Burrows, M., Abadi, M., Needham, R.: A Logic of Authentication. ACM Transactions on Computer Systems 8(1), 18–36 (1990)