EAP-Shared Key Exchange (EAP-SKE): A Scheme for Authentication and Dynamic Key Exchange in 802.1X Networks

Status of this memo

This document is an individual contribution for consideration by the PPPEXT Working Group of the Internet Engineering Task Force (IETF). Comments should be submitted to the ietf-ppp@merit.edu mailing list.

Distribution of this memo is unlimited.

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at: http://www.ietf.org/ietf/1id-abstracts.txt The list of Internet-Draft Shadow Directories can be accessed at: http://www.ietf.org/shadow.html.
Abstract

This note describes two methods for authentication of Mobile Nodes (MN) and generation of per session, per node encryption keys, suitable for use at layer-2 in 802.1x networks (such as 802.11 wireless LANs). The methods apply to scenarios where a Mobile Node (MN) is in a foreign network such as public 802.3 network that uses Home-AAA and Foreign-AAA services. The methods assume that a pre-deployed cryptographically secure pre-shared key is present on the MN, and use of the 802.1X standard [1], Extensible Authentication Protocol (EAP) [2] messages, and RADIUS Authentication Servers. The first method offers a full set of security features, and requires the introduction of relatively minor changes to RADIUS and EAP. The second method is a simplified version of the first. It requires minimal modifications to existing RADIUS and EAP standards, but provides a lower level of security. The protocol can easily be extended to support the migration from RADIUS to DIAMETER [3].

1 Introduction

In recent years ubiquitous Internet access from public places such as hotels, malls, airports, etc., has become feasible. Popular wired and wireless LAN technologies such as IEEE 802.3 Ethernet and IEEE 802.11b wireless are some of the attractive options available to provide such access. A typical scenario, wherein a user has pre or post-pay access to the network, must guarantee the following: (1) The user has accurate pre-established credentials to access the network, and (2) the user session is secure for its lifetime. The first requirement is satisfied using authentication, whereas the second requirement can be satisfied using key-based encryption schemes that use per-session encryption key setup at the start of an authenticated session.

The IEEE 802.1x Port Based Access Control standard [1] has emerged as a preferred mechanism for authenticating users before providing network access to IEEE 802 LANs. It employs the Extensible Authentication Protocol (EAP) [2] originally standardized for PPP link layer authentication to support multiple authentication methods.

In this document, we describe a new EAP authentication and key exchange method, EAP-Shared Key Exchange (EAP-SKE), which (1) avoids transmission of critical authentication information such as
password or encryption key(s) in the clear on the wired or wireless medium; (2) supports efficient mutual authentication between the MN and a Home-AAA (H-AAA); (3) provides per-user, per-session dynamic encryption keys that are guaranteed to be fresh; and (4) efficiently supports roaming across multiple network provider networks. In addition, we also describe a simplified version of such protocol which requires minimal modifications to existing RADIUS and EAP standards, at the cost of providing a lower level of security.

2 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [4].

2.1 Notation

The following notation is used throughout this document:

A_(A,B) A security association between nodes A and B, as defined in [5], which includes a shared secret key known to A and B, and allows the secure exchange of information between nodes A and B.

AAA Authentication, Authorization and Accounting (server). Same as the Authentication Server in the IEEE 802.1x terminology. For simplicity, we will use the term AAA server throughout this document.

H-AAA: Home AAA; F-AAA: Foreign AAA.

Authenticator System (AS) An IEEE 802 LAN entity that uses port based network access control. For example, in case of an IEEE 802.11 LAN, an Access Point (AP) may be an authenticator system. An AS consists of (1) a Port Access Entity (PAE) that plays the role of an authenticator to authenticate a user, and (2) an entity that provides the network access service offered by the AS. In the remaining document, we will refer to the PAE entity as AS-PAE.

AS-PAE See Authenticator System above.

K_(A,B) A shared secret key known to nodes A and B.
MAC(K,.)  Message Authentication Code (or integrity check function), which is applied to a piece of information for authentication using a key K. Examples include keyed cryptographic hash functions (e.g., keyed-MD5 [6], keyed-SHA-1 [7, 8], HMAC [9, 10], etc.), and block ciphers (e.g., AES in CBC-MAC mode [11]).

MN  Mobile Node.

NSP  Network Service Provider.

N_1  A nonce, in this case a freshly generated (unstructured) random number. Nonces are typically implemented as pseudo-random bit strings of length 64-128.

N_2  A nonce, in this case a freshly generated random number, or a monotonically incremented counter.

N_3  A nonce, in this case a freshly generated random number, a monotonically incrementing counter, or a pre-defined constant.

PRF(K,.)  A pseudo-random function with key K. Pseudo-random functions [12] are characterized by the pseudo-randomness of their output, namely, each bit in the output of the function is unpredictable if K is unknown. We use pseudo-random functions for the derivation of session keys given the shared key K. In practice, pseudo-random functions are realized using AES in CBC-MAC mode (and other block ciphers), or keyed one-way hash functions (see examples of MAC functions above).

Supplicant  A 802 LAN port that wishes to obtain (network) services offered by the AS. The mobile node (MN) in an 802.1X network will contain an entity that serves as a supplicant. For brevity, we refer to such an entity as MN-SUP.

Authentication and Dynamic Key Exchange in IEEE 802 networks with Port Access Control

It is desirable that when a MN roams into a foreign 802 network, its supplicant should be able to establish credentials with the NSP of the foreign network to obtain network access. One example
of this is as follows: user John Doe who has an account with, say, carrier.com roams into a public network in a mall or airport run by an NSP such as (hypothetical) airport-mall.net. It is desirable that John be able to present his credentials with carrier.com to airport-mall.net to authenticate himself and obtain network access. The access charge for this service is later posted to John’s monthly access bill with his carrier via a revenue settlement agreement between the two NSPs. This requires (a) that the AAA services employed by carrier.com and airport-mall.net peer with each other using pre-established secure channels and (b) that a database of AAA services be exchanged among the providers. Such a scenario already exists between providers which provide network access to roaming dial-up customers.

3.1 Assumptions

We assume that as a part of a service contract with a network provider (say carrier.com), the supplicant has (1) a pre-configured network access identifier (NAI) (e.g., john.doe@carrier.com), and (2) a pre-configured security association with its Home AAA server (H-AAA), which includes a sufficiently long (say, 128 bit) key K_(MN,H-AAA), as shown in Figure 1. At the same time, we assume that each AS-PAE in the foreign domain has a pre-configured security association A_(AS-PAE,F-AAA) with the F-AAA, which allow the F-AAA and the AS-PAE to authenticate and encrypt the messages that they exchange. The mechanism by which these security associations are setup is outside the scope of this document.

We also assume that a security association A_(F-AAA,H-AAA) exists between the F-AAA and H-AAA, which allows the F-AAA and H-AAA to authenticate and encrypt each other’s messages. This association is setup as part of the roaming agreement between the foreign and home domains, and allows the home domain to trust the AAA servers and the AS-PAEs of the foreign domain. Also in this case, the mechanism by which this security association is setup is outside the scope of this document.

Furthermore, we assume that the F-AAA and the AS-PAE are trusted network elements, and that they will not deviate from the execution of the protocol.
Figure 1: Entities in the proposed 802.1X architecture
3.2 Security requirements

Given the assumptions outlined in section 3.1, our protocol must meet the following high level objectives:

1. Fraud protection, i.e., prevent unauthorized users from receiving service from visited networks without paying for it.

2. Prevent session hijacking (as defined in [5]), i.e., prevent users from seizing control of a communication association previously established by another user.

At a more refined level, the goals of our scheme can be specified as follows:

Secrecy and Authenticity: Guarantee the participating entities that only the intended party learns the key exchanged, and that this key is fresh, random and unique. Specifically, the scheme should support the following:

- Requirement 1 (Authenticate MN-SUP): Allow H-AAA to authenticate and authorize that the MN-SUP has rights to establish a security association with, and receive service from the AS in a foreign domain with which the home domain has a roaming agreement.

- Requirement 2 (Authenticate H-AAA): Allow the MN-SUP to establish that it is authenticating to a trusted H-AAA that is in possession of K_(MN-SUP,H-AAA);

- Requirement 3 (Session Key Establishment): Establish per session dynamic shared secret key K_(MN-SUP,AS-PAE). Guarantee both MN supplicant and H-AAA that this key is fresh, random and unique.

Forward Secrecy: When used in this document forward secrecy refers to the notion that compromise of a session key will permit access only to data protected by that key. In other words, even if an attacker is eventually able to derive the key for one session, future (and past) session keys (and, of course, the pre-shared key) are not compromised. See, e.g., [13 , 14 , 15 ] for more general notions of forward secrecy.
Efficiency: Keep both the number of messages exchanged between the parties and the computational overhead to a minimum. Simplicity, in turn, makes the scheme amenable to analysis and more formal proof.

3.3 Protocol description

Conceptually, our method consists of two phases:

1. Exchange and authentication, where parties (MN-SUP and H-AAA) exchange nonces and MACs applied to arguments consisting of nonces and IDs using their pre-shared key ($K_{(MN-SUP,H-AAA)}$), and

2. Generation of the session key, where parties apply a pseudo-random function to a common argument pertaining to the phase 1 exchange.

This approach follows the known techniques from the two-party shared-key model originating in [16] and further developed and analyzed in, e.g., [17, 18, 19, 14, 20]; however, it is extended here to accommodate the scenario of relaying agents -- such as the AS-PAE and F-AAA -- with whom the MN does not share any security association, but which are trusted by the H-AAA.

We now describe the method in detail. Note that all communication between the RADIUS client on AS-PAE and F-AAA is secured using a pre-established or dynamic security association $A_{(AS-PAE,F-AAA)}$. Similarly, all communication between F-AAA and H-AAA is secured using a pre-established or dynamic security association $A_{(F-AAA,H-AAA)}$.

1. Discover the AS-PAE and associate or reassociate: In case of wired LAN technologies such as 802.3 Ethernet, this step is null. However, in case of wireless 802.11b LANS, the MN listens to the beacons sent out by the AS (the AP) or uses network probes to locate the AP from which it can receive sufficiently strong radio signal. The MN begins the authentication and key establishment session by sending an EAPOL-Start packet with Ethernet encapsulation, specifically, [Protocol Type= 0x888E (EAPOL),ProtoVersion=0x0000 0001, PktType= 0x0000 0001(EAPOL-Start), PktBodyLength=0].
Figure 2: 802.1X authentication and key exchange
2. AS-PAE obtains the NAI: The AS-PAE issues a EAP request frame (Figure 2 - Step 1) with EAP Pkt=[Code=1, Identifier=X1, Length=l1,[Type= Identity, OptionalMsg="Please Enter Your NAI"]). This packet is encapsulated in the 802.1X frame format as [Protocol Type= 0x888E (EAPOL),ProtoVersion=0x0000 0001 (EAP), PktType= 0x0000 0001(EAP), PktBodyLength= M, PktBody= EAPReqPkt] and handed over to the EAP software stack on the MN, which could be included with the vendor supplied driver for the 802.1X-compliant card. The MN responds with an EAP response [Code=2, Identifier=X1, Length=l2, [Type=NAI, (john.doe@carrier.com)]] (Figure 2 - Step 2). We call the "carrier.com" portion of the NAI as the realm. Note that the EAP response packet is again encapsulated as an EAPOL packet [Protocol Type= 0x888E (EAPOL),ProtoVersion=0x0000 0001 (EAP), PktType= 0x0000 0001(EAP), PktBodyLength= N, PktBody= EAPRespPkt]. For sake of brevity, henceforth, we will not show the EAPOL packet encapsulation for remaining EAP packets in the protocol exchange.

3. AS-PAE presents the challenge N_1: The AS-PAE generates a (128 bit) random challenge (nonce) N_1. The AS-PAE then issues an EAP request frame with [Code=1, Identifier=X2, Length=l3,[Type= EAP-SKE (0x20), Sub-type= AS-PAE-Challenge (0x01), Type-Data=[Value-Size= sizeof(N_1), Value= N_1, Name= "String identifying AS-PAE")]} (Figure 2 - Step 3). The name string should not be null or CR/LF terminated, and should be preferably ASN.1 format. An example string could describe the name of service provider the AS-PAE belongs to and some additional information such as "Carrier.com’s AS-PAE (RegionCode=0x20, ID=0x1234)". The length field l4= 7+sizeof(N_1) + sizeof (Name)) can be used to find the exact length of name field.

Reception of this packet signals to the MN-SUP that the AS-PAE is requesting authentication scheme EAP-SKE. In the event MN-SUP does not support the scheme, it will send EAP Response of type NAK, specifically, [Code=2,Identifier=X2,Length=l3-1, [Type=NAK (3), Type-Data= 'Desired Authentication Type’]]. The ‘Desired Authentication Type’ is a number (> 3) standardized for the authentication scheme that MN-SUP supports. For example, if MN-SUP prefers One-Time Password (OTP) scheme, it sets Type-Data=5. The length field l3-1 is set to 6 bytes.
4. The MN-SUP also generates a nonce $N_2$ and computes the authenticator

$$AUTH_1 = MAC(K_{(MN-SUP,H-AAA)}, \ N_1 \ | \ N_2 \ | \ NAI).$$

The MN-SUP concatenates $N_2$ with the authenticator $AUTH_1$ and sends back an EAP response packet $Resp = [\text{Code}=2, \Identifier=X2, \text{Length}=14, [\text{Type}=EAP-SKE (20), \text{SubType}=MN-Sup-Auth1-Challenge (0x02), \text{Type-Data}=[\text{Value-Size}=16 + sizeof(N_2), \text{Value}= Authenticator AUTH1 \ | \ N_2, \text{Name}=\text{String identifying MN}]]$ (Figure 2 - Step 4). Note that another possibility would be for MN to generate a counter-based value instead of a nonce. That would simplify the operations that the MN would be required to carry out, without compromising security.

5. Forward challenge response to H-AAA server: Each AS-PAE runs a RADIUS client and maintains a security association with a local AAA server F-AAA. When the MN response is received, the AS-PAE removes the 802.1X encapsulation and generates a RADIUS access-request with $[\text{User-Name}=NAI, \text{EAP-Authenticator}= AUTH1, \text{EAP-Challenge}=N_1, \text{MN-Nonce}= N_2]$, and sends the access-request to F-AAA (Figure 2 - Step 5).

6. The F-AAA server resolves the user NAI to an authoritative H-AAA server using a local database or a network resident LDAP server. It then uses the RADIUS protocol to forward the access-request to the H-AAA server (Figure 2 - Step 6).

7. Processing at H-AAA (Figure 2 - Step 7): On receipt of the access-request, the H-AAA server does the following: (1) Look up $K_{(MN-SUP,H-AAA)}$, the key for user ‘NAI’; (2) calculate the Authenticator $AUTH1’ = MAC(K_{(MN-SUP,H-AAA)}, \ N_1 \ | \ N_2 \ | \ NAI)$ as explained in item 4 above; and (3) compare $AUTH1’$ with the received $AUTH1$. If the two do not match, authentication fails and the H-AAA server relays the failure message to F-AAA, which will forward it to the RADIUS client in the AS-PAE. The AS-PAE eventually sends an EAP Failure response packet $[\text{Code}=4, \Identifier=X3, \text{Length}=15]$ to MN. If the authentication succeeds, the H-AAA undertakes the following steps:

1. Computes the authenticator

$$AUTH2 = MAC(K_{(MN-SUP,H-AAA)}, \ N_2 \ | \ N_1 \ | \ NAI).$$

(Note change in the order of arguments with respect to $AUTH1$.)
(2) Generates $N_3$, which can be either (a) a random number, (b) a monotonically increasing integer or (c) a pre-configured constant. Refer to section 5 for an explanation on the effects of the form $N_3$ on the key generation process.

(3) Computes the session key $K_{(MN-SUP,AS-PAE)}$ as

$$K_{(MN-SUP,AS-PAE)} = \text{PRF}(K_{(MN-SUP,H-AAA)}, N_3 \mid \text{AUTH2})$$

(4) Forwards $K_{(MN-SUP,AS-PAE)}$, AUTH2 and $N_3$ to F-AAA in a RADIUS access-response (Figure 2 - Step 8).

8. Processing response at AS-PAE: The F-AAA forwards the response to the RADIUS client on AS-PAE (Figure 2 - Step 9). The AS-PAE extracts the $K_{(MN-SUP,AS-PAE)}$ key allocated for the MN and sends an EAP request packet [Code=0x01, Identifier= X3, Length=15, [Type= EAP-SKE (0x20), Subtype= AS-PAE-Auth2-N2 (0x03), [TypeData=[Auth2-len= k, AUTH_2, NonceLength= h, N_3]]]) to MN (Figure 2 - Step 10).

In both cases, MN does two things:

(1) It first uses $N_1$ and $N_2$ to compute AUTH2’ as in item 7 and compares it with the supplied AUTH2. If the two match, it concludes that its request was processed by a valid H-AAA. If the two do not match, MN may send a EAP-SKE-failure message [Code=2, Identifier= X3, Length=16, [Type=0x20, Subtype=SKE-Fail (0x05), [FailReason=''Auth2 match failed'']].

(2) Using $K_{(MN-SUP,H-AAA)}$, $N_3$ and AUTH2 it generates $K_{(MN-SUP,AS-PAE)}$ following the exact same procedure used at the H-AAA server. Note that the session key is not transmitted from the AS-PAE to the MN but is locally computed.

If both the steps complete without error, MN-SUP sends a EAP-SKE-success message [Code=0x02 Identifier=X3, Length=17, [TypeData=0x20, Subtype= SKE-Success (0x04), [SuccessMsg=''Setting up the keys'']] (Figure 2 - Step 11). This signals to AS-PAE that EAP-SKE exchange has successfully completed.
9. AS-PAE signals completion of EAP exchange: At this point AS-PAE sends a response packet -- EAP Success [Code=4 Identifier=X3, Length=0] success message to MN to signify that the authentication and key set up is successfully completed (Figure 2 - Step 12).

3.4 Implementation Considerations

The following points must be taken into account when implementing the new protocol.

1. Modifications to EAP: In Section 3.3), we defined a new EAP Type EAP-SKE (0x20) for steps 3, 4, 10, 11. This new type and the corresponding number must be allocated by IANA. Any EAP compliant supplicant MUST support this new EAP type and corresponding EAP-SKE subtypes AS-PAE-Challenge (0x01), MN-Sup-Auth1-Challenge (0x02), AS-PAE-Auth2-N2 (0x03), SKE-Success (0x04), SKE-Fail (0x05)

2. Modifications to RADIUS: Also, in steps 5 and 8 (Section 3.3), the new RADIUS access request and response attributes must be defined and must be supported by the H-AAA server and RADIUS client in the AS-PAE. We propose using the vendor extensions defined in [22 ].

3. Client capabilities: The client devices must be capable of computing MAC and PRF functions. Normally the nonce N_2 SHOULD be a freshly generated (unstructured) random number, of length 64–128 bit. In case the MN does not possess the hardware and/or software capabilities to generate such random, N_2 MAY be a monotonically increasing counter.

3.5 Optional Optimizations

We can consider following optimization in Step 11 to allow MN-SUP to establish to AS-PAE that it has generated the same session key
K_(MN-SUP, AS-PAE). MN can encrypt a mutually known quantity such as NAI, N_3, MN MAC or their combinations using the new session key and enclose that in the EAP-SKE-Success response. AS-PAE can decrypt and verify this quantity before sending a EAP-Success packet. We list this optimization as optional for two reasons:
1. The probability of generating different keys will be very small.
2. If the two keys are different, AS-PAE (AP) will eventually detect it and disassociate the MN (as the count of incorrect decrypted packets increases).

For mediums that specify that AS-PAEs must use beacons to advertise their presence to potential MN-SUPS, and in case such beacons can be extended to include a "nonce" field, it is possible to further optimize our scheme. In this case, the AS-PAE would periodically re-generate a random nonce N_1, and would broadcast such nonce in the beacons it emits. All MNs wishing to authenticate during the period of validity of N_1 would use such value in the computation of AUTH1.

4 Recommendation for the MAC and PRF algorithms

EAP-SKE implementations compliant with this document MUST implement HMAC-MD5 [10] as MAC function and MD5-prefix-suffix [23] as PRF, as a minimum. This allows RADIUS servers that are already compliant with the requirements of dynamic key distribution for Mobile IP [24] to support EAP-SKE without implementing new MACs or PRFs. However, any EAP-SKE implementation can optionally implement other MAC and/or PRF algorithms.

5 Security Considerations

Let us recall the assumptions we made in section 3.1. In particular, we assume that AS-PAE and F-AAA are network elements which are trusted by the H-AAA, by means of the existence of A_(H-AAA,F-AAA) and A_(AS-PAE,F-AAA).

Consider authenticators AUTH1 and AUTH2 in steps 4 and 7, respectively. The nonces N_1 and N_2 in the authenticators act as challenges to H-AAA and MN to "prove" to each other the possession of the pre-shared key K_(MN-SUP,H-AAA). Moreover, including N_1 (respectively, N_2) assures the H-AAA (resp., the MN) that the authenticator is fresh for every session. The fact that N_1 is generated by the AS-PAE and not by the H-AAA does not invalidate this claim, since the AS-PAE is trusted by the H-AAA by virtue of
The included identities (i.e., the username and realm parts of the NAI) serve to reassure the parties of the correct binding between the shared key and their identities.

The authenticity, freshness and randomness of the session key follow from the authenticity and freshness of AUTH2, and the properties of pseudo-random functions; specifically, the value \( \text{PRF}(K_{(MN-SUP,H-AAA)}, N_3 | \text{AUTH2}) \) is (computationally) independent of any other value output by the function. Thus, the protocol reveals no information to an adversary on the value of the session key \( K_{(MN-SUP,AS-PAE)} \) (forward secrecy).

Replay attacks by illegitimate network elements are detected by the MN and the H-AAA by the application of MAC functions to \( N_1 \) and \( N_2 \), given that both nonces are freshly generated every time by the AS-PAE and MN. Denial Of Service (DOS) attacks are alleviated, because if they are mounted by replaying these authentication messages, they would be detected as described above.

A full adversarial model can also be considered, where the trusted AS-PAEs and F-AAAs might misbehave. In particular, since both \( N_1 \) and \( N_2 \) are sent in clear between the MN-SUP and the AS-PAE, a misbehaving AS-PAE could replay an access-request with the same \( N_1 \) and \( N_2 \) as a previous request. In this case, the inclusion of a fresh value for \( N_3 \) (either randomly generated or monotonically increasing counter) in step 7, Section 3.3 would counteract the replay, by guaranteeing that each generated session key is different even in cases where \( N_1 \) and \( N_2 \) are replayed. If the H-AAA decides not to consider such cases where AS-PAEs might misbehave, \( N_3 \) MAY be set to a pre-defined constant value.

The prevention of attacks pertaining to a full adversarial model, other than the one mentioned above, is outside the scope of this document.

6 Simplified Scheme for Existing RADIUS Servers

The protocol specified in Section 3.3 satisfies the basic requirements outlined in Section 3.2. However, as explained in Section 3.4, to satisfy these requirements, our protocol requires modifications to the existing IETF standards such as EAP and RADIUS. By relaxing Requirements 2 and 3 (freshness of the exchange) in Section 3.2, our scheme can be simplified further to
allow its operation with minimal modifications to existing EAP and RADIUS specifications. This simplified version of the protocol would work as the one specified in Section 3.3, except for the following:

1. In step 4, the MN does not generate $N_2$. Also, the MN calculates $AUTH_1$ as

   
   $AUTH_1 = MD5(\text{high-order byte from } N_1 \mid K_{(MN-SUP,H-AAA)} \mid \text{least order bytes from } N_1)$.

   The MN forwards $AUTH_1$ to the AS-PAE in an EAP packet as $\text{Resp} = [\text{Code}=2, \text{Identifier}=\text{TBD}, \text{Type}=4(\text{MD5-Challenge}), \text{Type-Data}=[\text{Value-Size}=16, \text{Value}=AUTH_1])$.

2. In step 5, the AS-PAE builds a RADIUS access-request as

   $[\text{User-Name}=\text{NAI}, \text{CHAP-Password}=\text{high-order byte from } N_1$ followed by $AUTH_1$, $\text{CHAP-Challenge}=\text{least order bytes from } N_1]$,

   sends the access-request to the F-AAA.

3. In step 7, the H-AAA checks the validity of $AUTH_1$ by following the same procedure as specified by RADIUS CHAP. It then generates a random $R$ and generates $K_{(MN-SUP,AS-PAE)}$ as:

   
   $K_{(MN-SUP,AS-PAE)} = PRF(K_{(MN-SUP,H-AAA)}, R \mid \text{NAI})$.

   The H-AAA forwards $(K_{(MN-SUP,AS-PAE)}, R)$ to the F-AAA in a RADIUS access-response.

4. In step 9 (after message 10 in Figure 2), MN does not have to check $AUTH_2' = AUTH_2$, and can proceed to generate $K_{(MN-SUP,AS-PAE)}$ from $R$ and $K_{(MN-SUP,H-AAA)}$ by following the same procedure as described in the previous step.

By relaxing the security requirements, and by using the same MD5 algorithm and mode of operation that CHAP uses, this simplified version of the protocol requires minimal changes to existing RADIUS servers and clients. In particular, servers and clients that already support Mobile-IP and some of its extensions [24,25] will need no modifications to work with this version of the protocol.
However, given the importance of the much higher security requirements that the full version of this protocol satisfies, we feel that the deployment of its simplified version should be limited. In our opinion, it should be primarily considered in environments that are transitioning to the full version of this protocol. In fact, in scenarios where roaming agreements between multiple providers are used to serve each other’s customers, both key freshness (Requirement 3) and mutual authentication (Requirement 2) are essential.

7 Migration to DIAMETER

In this document the protocol used to transfer the authentication information and the key material from AS-PAE to F-AAA to H-AAA and back is RADIUS, given its wide installed base. Migration to DIAMETER [3] should not present any difficulties, since DIAMETER already provisions mechanisms to collect the same authentication information as RADIUS, and to distribute key material to interested parties. The details of such mechanisms, and how they would be applied to EAP-SKE are outside the scope of this document.

8 Applicability to private networks

Our scheme can be readily applied to NAI-based authentication and dynamic key exchanges in private enterprises or stand-alone public networks without roaming agreements. In such scenarios, for example, users would buy prepaid wireless-internet access from a local ISP at public facilities like airports and shopping malls. The ISP would then use a local AAA server to authenticate and authorize the user, and to generate and distribute the required layer-2 keys.

In such cases the step 1 to 5 in Figure 2 apply to this scenario as well, with the only difference that the F-AAA would simply become "AAA". Steps 6 and 8 are eliminated and step 7 is carried out by the local AAA. Steps 9, 10, 11, 12 from Figure 2 for the distribution of session key and/or authenticators to AS-PAE and MN are identical in this case.

9 Open Issues

The proposed scheme can be modified to include an option that addresses the cryptographic bootstrapping problem associated with
the pre-existence of the shared key $K_{(MN-SUP, H-AAA)}$ of 128 bits or more, by specifying a mechanism by which $K_{(MN-SUP, H-AAA)}$ can be dynamically derived either from a password, from a secure token, or from other kinds of one-time password mechanisms. This way the system would support authentication schemes that ultimately rely on the user for inserting some form of user-id and password.

In addition, the scheme as it is described in this document would require the MN to re-authenticate to its H-AAA every time a handoff occurs. The latency of this operation would clearly present a problem, in particular for QoS-aware traffic. One possible solution would be for the AS-PAE, MN and H-AAA to generate array of ($N_1$, $N_2$, $N_3$) nonces, and arrays of corresponding ($AUTH$, $K$). Such arrays could be cached at the F-AAA and at the MN, so that authentications subsequent to the first one could be performed without the involvement of the H-AAA. Another possible solution would be to perform subsequent authentications between the MN and the AS-PAEs by using a MAC function applied to random numbers and $K_{(MN-SUP, AS-PAE)}$, which would get transferred among AS-PAEs by means of a context-transfer protocol such as the one being defined in the SeaMoby IETF working-group [26].

Another issue is that in both schemes the NAI of the user is sent in "clear" before a security association is established with AS-PAE in the network, therefore potentially exposing some information about the user (his/her user-name and realm). A simple but partial solution to this problem is to use meaningless, long ASCII strings as user identification (i.e. q124abce356@carrier.com, IMSI@carrier.com or TIMSI@carrier.com of the MN if such identification exists instead of john.doe@carrier.com) and minimize information an eavesdropper can gather.

Appendix A - Patent Issues

This is to inform you that Lucent Technologies has applied for and/or has patent(s) that relates to the attached submission.

This submission is being made pursuant to the provisions of IETF IPR Policy, RFC 2026, Sections 10.3.1 and 10.3.2.

Lucent Technologies Inc. will offer patent licenses for submissions made by it which are adopted as a standard by your organization as follows:
If part(s) of a submission by Lucent is included in a standard and Lucent has patents and/or pending applications that are essential to implementation of the included part(s) in said standard, Lucent is prepared to grant - on the basis of reciprocity (grantback) - a license on such included part(s) on reasonable, non-discriminatory terms and conditions.

10 Acknowledgments

We would like to thank Peretz Feder, Pete McCann, Simon Mizikovsky and Reuven Shapira from Lucent Technologies for useful discussions on this topic and comments on this draft.

References


Authors’ Addresses

Luca Salgarelli, Milind M. Buddhikot, Juan Garay, Scott Miller, Uri Blumenthal, Sarvar Patel
Bell Laboratories - Lucent Technologies
101 Crawfords Corner Rd.
Holmdel, NJ 07733, USA
Voice: +1-732-332-6870
Fax: +1-732-949-7397
E-mail: {salga,mbuddhikot,garay,scm,uri,sarvar}@lucent.com

Dorothy Stanley
Agere Systems
2000 North Naperville Rd, Room 5B-441
Naperville, IL 60566, USA
Voice: +1 630 979 1572
Fax: +1 630 979 1572
E-mail: dstanley@agere.com