An EAP-EHash authentication method adapted to resource constrained terminals

Omar Cheikhrouhou, Maryline Laurent, Amin Ben Abdallah, Maher Ben Jemaa

Ann. Telecommun.
DOI 10.1007/s12243-009-0135-9

An EAP-EHash authentication method adapted to resource constrained terminals

Omar Cheikhrouhou · Maryline Laurent · Amin Ben Abdallah · Maher Ben Jemaa

Received: 28 January 2009 / Accepted: 7 October 2009

Abstract In the era of mobile and wireless networks, the growing complexity of end devices and the accentuated tendency towards miniaturization of them raise new security challenges. Authentication is a crucial concern in resource constrained environments, and despite the great number of existing EAP methods, as explained in the article, we are still in need for EAP methods tightly adapted to wireless environments and satisfying heterogeneity of terminals and their limitations of resources. After a first comparative analysis of existing EAP methods, this article presents a new EAP-EHash method (EHash for encrypted hash) that is adapted to the highly vulnerable wireless environment by supporting mutual authentication and session key derivation and offering simplicity, rapidity, and easy-to-deploy features. This EAP-EHash was formally proven to satisfy the claimed security properties, thanks to the AVISPA tool. Implementation of it on an 802.11 testbed platform gave realistic authentication delays averaging 26 ms and thus proved that EAP-EHash is competitive to EAP-MD5 that is known to be the simplest of the EAP methods. Features of EAP-EHash include short execution delays and low bandwidth consumption, and as such, it appears attractive for wireless.

Keywords EAP · EAP methods · EAP-MD5 · EAP-TLS · EAP-EHash · Authentication protocol · Validation · AVISPA

O. Cheikhrouhou · M. B. Jemaa
Ecole Nationale d'Ingénieurs de Sfax,
Unité de recherche ReDCAD,
BP W 3038, Sfax, Tunisia

M. Laurent (✉) · A. B. Abdallah
Institut TELECOM,
TELECOM SudParis,
CNRS Samovar UMR 5157-9 rue Charles Fourier,
91011 Evry, France
e-mail: Maryline.Laurent@it-sudparis.eu

Abbreviations

3DES	Triple DES
AAA	Authentication, authorization, accounting
AK	Authentication key
AP	Access point
AS	Authentication server
AVISPA	Automated validation of internet security protocols and applications
CPU	Central processing unit
DES	Data encryption standard
DoS	Denial of service
EAP	Extensible authentication protocol
EHash	Encrypted hash
EK	Encryption key
EP	Enforcement point
IKEv2	Internet Key Exchange version 2
KDK	Key derivation key
MK	Master key
MD5	Message digest 5
MIC	Message integrity check
MITM	Man-in-the-middle
PKI	Public key infrastructure
PMK	Pairwise master key
PRF	Pseudo-random function
PSK	Pre-shared key
PTK	Pairwise transient key
SHA-1	Secure hash algorithm-1

1 Introduction

In the era of wireless and mobile networks, end devices are becoming more and more complex supporting connectivity to multiple networks, like GSM/GPRS/UMTS, Wi-Fi, and

Bluetooth, and offering multimedia functions, like (video) camera, VoIP, MPEG3 reader, and even GPS geo-localization. The growing complexity of end devices and the accentuated tendency towards miniaturization of them raise new challenging problems for researchers like designing new mechanisms which take into account the limited resources of terminals. This pretty new resource constraint is essential for next-generation devices to benefit from advanced networks functionalities without proceeding to too frequent battery reloading.

Security is reported as one of the most challenging field where strong and rapid advances are needed. Security mechanisms are known to be usually high CPU consuming and are critically required in wireless networks where vulnerabilities are even more important than in wired networks. On the one hand, cryptography serves to introduce robustness into security mechanisms but is costly in terms of CPU and delays. Security mechanisms' heaviness is sometimes even worse when extra administration of cryptographic keys is required for instance. On the other hand, reasons for larger vulnerabilities in wireless networks compared to wired's are related to the wireless shared communication medium (air) that makes it more exposed to external attacks. Where attacks onto wired networks assumed a physical intrusion onto the inside network of a company or an operator, eavesdropping, hijacking of existing sessions, and spoofing authorized end devices are much easier to perform onto wireless networks (with infrastructure). Other attacks are potentially more or less disruptive. Partly or fully scrambling the radio signals, for instance, might result into signal quality degradation or a denial of service. Another one known as man-in-the-middle might consist in a malicious device advertising as an access gateway (802.11 AP, GSM BTS...), so legitimate clients connect to. The interest for the attacker is to collect authentication data like login/passwords that would be helpful for his next spoofing attempts. For any of the hereabove attacks, whatever kinds of wireless networks being under analysis—cellular, Wi-Fi, and WiMAX networks—the only prerequisites for attackers are to be positioned in the coverage zone of the wireless networks, equipped with more or less sophisticated and expensive materials, and helped by experts (for some of the attacks).

As a consequence, today's security challenge is to define strong enough security mechanisms that are robust to most of the attacks and that satisfy the very strong constraints to be lightweight and fast. A number of works are under progress at the IETF standardization body. In [1], a security architecture is defined for controlling access to a network infrastructure, as illustrated in Fig. 1. In this architecture, an enforcement point (EP) under the control of an administrative authority (a private company, an operator...) does filter the inbound and outbound traffic of a client device according to the local administration's security policy. Usually, the EP is configured to accept inbound traffic from authenticated clients and to limit acceptance of messages from unauthenticated clients to the messages supporting the authentication phase. Those latter messages are relayed by EP back and forth between the authentication server (AS) of the network administration and the client. This is the duty of the AS to verify the identity of the EP. As soon as authentication is successfully completed, the EP applies the filtering rules specific to that client.

As shown in Fig. 1, the protocol supporting authentication exchanges between the client and EP is named the access protocol and might be IEEE 802.1X [2] that classically applies in the IEEE 802.11 environment or the generic over-IP protocol PANA (a protocol for carrying authentication for network access) [3] that supports authentication exchanges over IP. The AS is usually associated to an authentication, authorization, accounting (AAA) server centralizing all the authentication requests. The classical AAA protocols for performing authentication might be either the well-known remote authentication dial in user service (RADIUS) [4] or the more recent Diameter [5] that supports inter-administrative domain authentication.

This article focuses on the authentication procedures based on extensible authentication protocol (EAP) methods and refers exclusively to the IEEE 802.11 network and 802.1X environments as EAP testing was conducted on an 802.11 platform. As such, the vocabulary is next IEEE 802.11 specific, and as given in Fig. 1 (text in italics), the EP is referred to as authenticator or AP (for access point), the client device as supplicant, and AS as EAP server.

Let EAP's basic principle and reasons of success first be explained. EAP for extensible authentication protocol as the

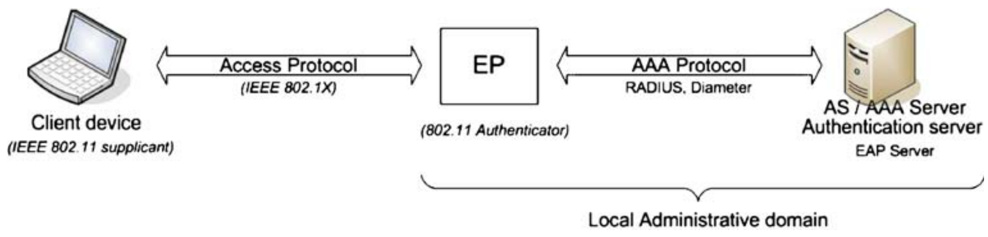


Fig. 1 The security architecture for controlling access to the network infrastructure (with *text in italics* applying to IEEE 802.11 environment)

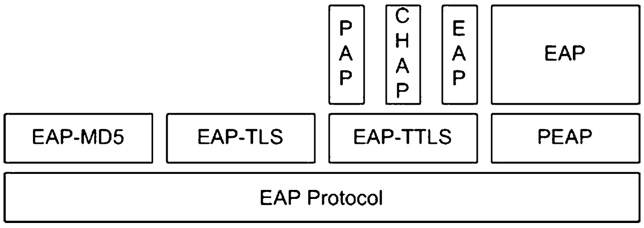


Fig. 2 The EAP stack underlining separation between EAP protocol and EAP methods

first generic protocol for authentication support introduces a lot of flexibility into security architectures and authentication handling. The successful idea, as depicted in Fig. 2, was to separate the EAP protocol [6] that transports the authentication data and the EAP methods that interpret the EAP data and check identity. The direct advantage is the possibility of changing from one EAP method to another while keeping the same security architecture. Additionally, to the large choice of available methods, this permits to rapidly activate a new EAP method in case the older one is detected as vulnerable.

Today's success of EAP is real as a number of network protocols needing authentication integrate EAP for authentication support. PPP [7] mainly for dial-up networks was the first one and was followed not only by IKEv2 [8] for IPsec environment but also RADIUS with its EAP over RADIUS solution [5] and Diameter with its new Diameter application for EAP support [9]. Furthermore, a number of classical authentication protocols were adapted as EAP methods. As such, historical protocols like transport layer security (TLS) [10], challenge handshake authentication protocol (CHAP) [7], subscriber identity modules (SIM) from GSM networks, respectively, became EAP-TLS, EAP-MD5, and EAP-SIM. More than 50 EAP methods [11, 12] have been designed during the past few years; among which, only ten are today published as IETF RFC, and six of them have been designed for the wireless context in the last 2 years, thus proving that the EAP field is still very challenging for industries and researchers. These EAP methods might be based on login/password, electronic certificate, smart card (SIM)...or might be a combination of those elements (e.g., certificates and login/password).

In the next section, historical and recent standardized EAP methods [11] are described to underline their properties and vulnerabilities for use in wireless network context. Table 1 summarizes features of each of them. The need for defining a new, simple, low-powered, and robust EAP method is discussed. Section 3 presents our new method named EAP-EHash (for encrypted hash) with detailed validation handling in Section 4 and description of the testbed platform in Section 5. Note that this EAP method is an extended version of the one presented in [13]. Finally, an analysis of EAP-EHash against current EAP methods is conducted in Section 6. Section 7 concludes the article.

2 Related IETF works on EAP methods and discussions

We consider of high importance the works conducted by the IETF since the IETF standards are largely implemented into security products, and specifications are freely available. In this section, the historical and well-introduced EAP methods—EAP-MD5 and EAP-TLS—are first presented. Even if not designed for wireless, both are worth studying. EAP-MD5 is known as the simplest existing EAP method with very light and fast processing. EAP-TLS is known for its high robustness against malicious attacks. Those two methods serve next in this article as references to evaluate the simplicity and robustness of EAP methods. That is why for a better understanding of the comparative analysis of Section 6, details of both of them are given. Then, newly defined methods for wireless, namely EAP-password authenticated exchange (PAX), EAP-SAKE, EAP-pre-shared key (PSK), EAP-POTP, and EAP-FAST are introduced. Properties and vulnerabilities of all those methods are highlighted. A synthesis is given in Table 1, and a final discussion is proposed in Section 2.4.

2.1 EAP-MD5

The EAP-MD5 method is an adaptation of protocol CHAP [14] that implements challenge–response principle. EAP-MD5 requires a PSK between client and EAP server that generally takes the form of a password bound to a user name or any identity (e.g., an IP or MAC address).

As illustrated in Fig. 3 that applies to IEEE 802.1X environment, after getting the authenticator's EAP request (step 1), the client declines its identity (step 2) in an EAP message which is relayed to the EAP server. The EAP server then sends a random challenge to the client (step 3) which calculates a hash based on the challenge and the pre-shared key. The hash value is returned in an EAP message (step 4). The server performs the same hash computation and compares both values. Two identical hashes mean that the client owns the key; it successfully authenticates itself, and an access-accept is sent (step 5a). Otherwise, the authentication fails, and the server rejects the authentication (step 5b). According to that final decision, the authenticator authorizes or denies future data exchanges of the client.

2.2 EAP-TLS

The EAP-TLS [15] is an adaptation of protocol TLS (IETF RFC [16] obsoleted by [10]) using electronic certificates to implement mutual authentication between client and AS.

Table 1 Comparative synthesis of EAP methods

EAP methods	Authentication principle	Mutual authentication With/without PKI Authentication rapidity (# of messages)	Ciphersuite negotiation Protection of negotiation	Key derivation Effective key length	Fast reconnect Share state equivalence	Identity protection	Robustness
EAP-TLS	Based on TLS and electronic certificates	Yes With PKI Slow (6 messages)	Secure negotiation	Yes 256-bit	Not addressed Yes	No	Robust
EAP-MD5	Challenge-response	No Without PKI Fast (2 messages)	No negotiation	No	Not addressed No	No	Vulnerable to brute-force attacks
EAP-PAX-STD	Achieved by calculating a MIC that proves ownership	Yes Without PKI Fast (4 messages)	No negotiation	Yes 128-bit	Not addressed Yes	No	Robust
EAP-PAX-SEC	of pre-shared key	Yes With PKI Slow (6 messages)				Yes	Robust (if the certificate is signed by a trusted Authority)
EAP-SAKE	Achieved by calculating MIC that proves ownership of pre-shared key	Yes Without PKI Fast (4 messages)	Secure and partial negotiation	Yes 256-bit	Not addressed Yes	Optional	Robust
EAP-PSK	Achieved by calculating a MIC that proves ownership of pre-shared key	Yes Without PKI Fast (4 messages)	No negotiation	Yes 128-bit	Not addressed Yes	No	Brute-force attack on the Authentication Key (AK)
EAP-POTP	Based on OTP tokens to derive key used to compute MAC	Yes Without PKI Middle (4 messages)	Secure negotiation	Yes (in the advanced variant)	Addressed Yes	No	Robust
EAP-FAST	Established TLS tunnel, and further authentication through that tunnel	Yes With PKI Slow (minimum of 9 messages)	Secure negotiation	Yes	Addressed (Fast re-establishment of the secure tunnel) Yes	Yes	Robust
EAP-GPSK	Achieved by calculating a MIC that proves ownership of pre-shared key	Yes Without PKI Fast (4 messages)	Secure negotiation	Yes	Not addressed Yes	No	Robust
EAP-EHash	Challenge-response with encrypted response	Yes Without PKI Fast (minimum of 2 messages)	Secure negotiation	Yes 128-bit	Addressed (based on [30]) Yes	No	Robust

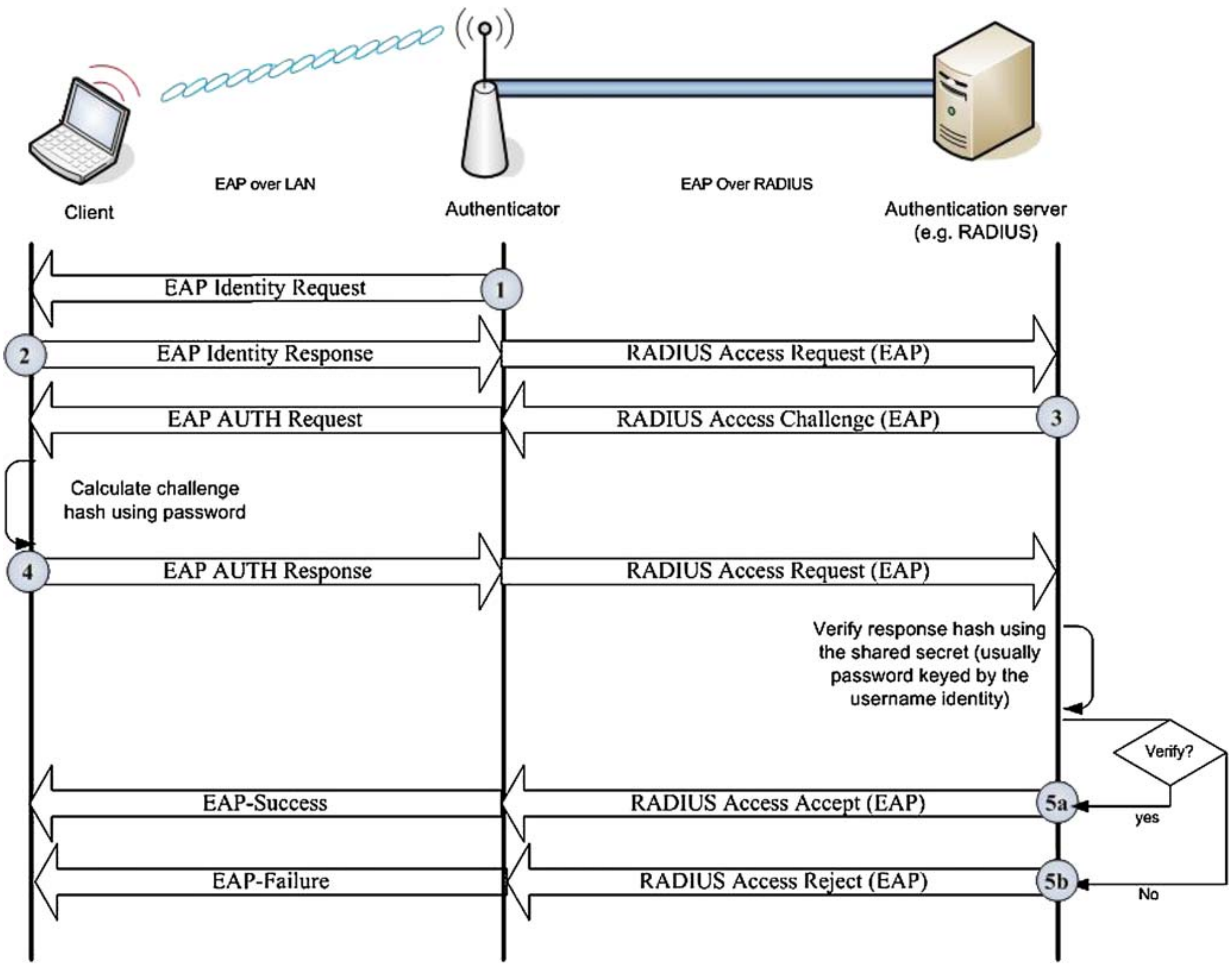


Fig. 3 The EAP-MD5 exchanges in the IEEE 802.11 context

As depicted in Fig. 4, after having detected a new 802.11 client, the authenticator launches the EAP-TLS authentication, requesting first client's identity. The response is then forwarded to the EAP server. Succeeding EAP messages are relayed back and forth by the authenticator between EAP client and server.

The EAP server initiates the authentication procedure which requires seven messages. Messages 2 to 5 are adapted from the TLS handshake protocol [16]. Those messages enable client and server to mutually authenticate, thanks to their public key certificates, to generate a common master key useful for later key derivation, to check EAP-TLS messages integrity, and to agree on a ciphersuite (hashing function, encryption algorithm) that serves to protect next EAP messages. As such, the EAP-TLS exchanges include transmission of both parties' certificates (TLS certificate) with possible explicit request to the other party (TLS_certificate_request). TLS server_key_exchange and TLS client_key_exchange enable parties to agree on a common master key. TLS client_hello and TLS server_hello serve to negotiate the ciphersuite, and TLS change_cipher_spec to activate the newly agreed ciphersuite on next EAP messages. The client authenticates to the server sending its signature in the TLS certificate_verify message and authenticates the server thanks to the TLS finished message that proves the server holds the good private key.

Finally, messages 6 and 7 end the EAP-TLS session. The EAP response message proves the server is successfully authenticated to the client and the EAP_Success message that the authentication was successfully handled by the server.

2.3 Recent IETF-published EAP methods for wireless

Six EAP methods—EAP-PAX, EAP-SAKE, EAP-PSK, EAP-POTP, EAP-FAST, and EAP-generalized pre-shared key (GPSK)—were published as RFC by the IETF in the

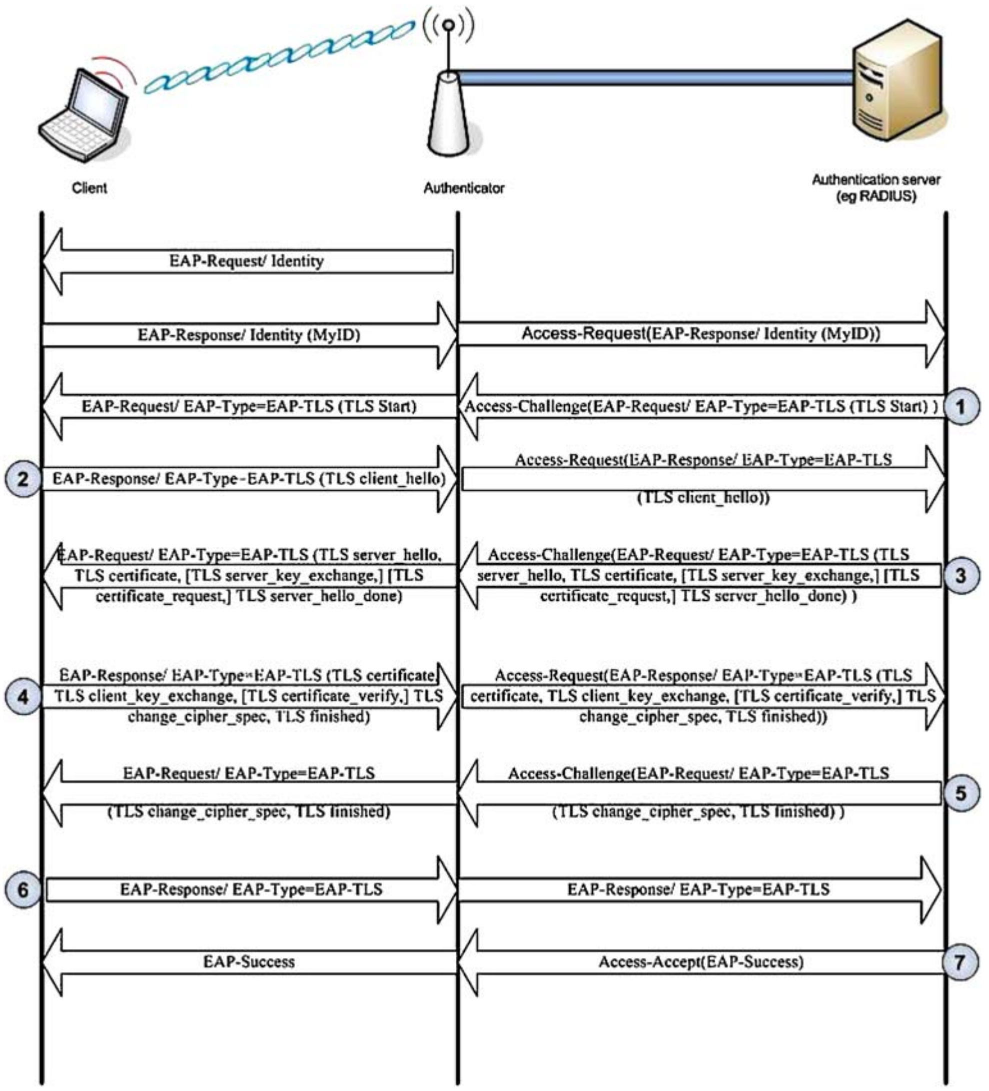

Client
Authenticator
Authentication server (eg RADIUS)
EAP-Request/ Identity
EAP-Response/ Identity (MyID)
Access-Request(EAP-Response/ Identity (MyID))
EAP-Request/ EAP-Type=EAP-TLS (TLS Start)
Access-Challenge(EAP-Request/ EAP-Type=EAP-TLS (TLS Start))
1
2
EAP-Response/ EAP-Type-EAP-TLS (TLS client_hello)
Access-Request(EAP-Response/ EAP-Type=EAP-TLS (TLS client_hello))
EAP-Request/ EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, [TLS server_key_exchange,] [TLS certificate_request,] TLS server_hello_done)
Access-Challenge(EAP-Request/ EAP-Type=EAP-TLS (TLS server_hello, TLS certificate, [TLS server_key_exchange,] [TLS certificate_request,] TLS server_hello_done))
3
4
EAP-Response/ EAP-Type=EAP-TLS (TLS certificate TLS client_key_exchange, [TLS certificate_verify,] TLS change_cipher_spec, TLS finished)
Access-Request(EAP-Response/ EAP-Type=EAP-TLS (TLS certificate, TLS client_key_exchange, [TLS certificate_verify,] TLS change_cipher_spec, TLS finished))
EAP-Request/ EAP-Type=EAP-TLS (TLS change_cipher_spec, TLS finished)
Access-Challenge(EAP-Request/ EAP-Type=EAP-TLS (TLS change_cipher_spec, TLS finished))
5
6
EAP-Response/ EAP-Type=EAP-TLS
EAP-Response/ EAP-Type=EAP-TLS
EAP-Success
Access-Accept(EAP-Success)
7

Fig. 4 The EAP-TLS exchanges in the IEEE 802.11 context

last 2 years, with the clear objective to apply onto wireless media. Features of them are synthesized in Table 1, and their description refers to the properties being identified by the IETF in Fig. 5 for wireless LAN environment.

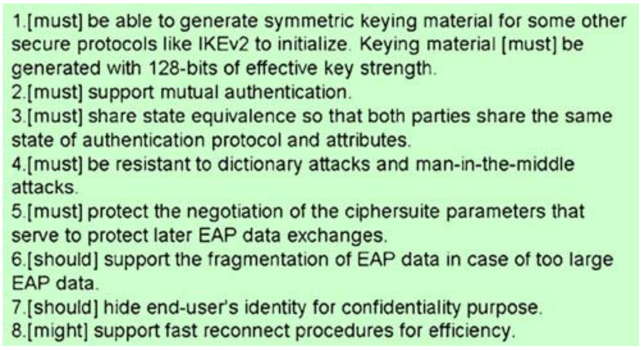

1.[must] be able to generate symmetric keying material for some other secure protocols like IKEv2 to initialize. Keying material [must] be generated with 128-bits of effective key strength.
2.[must] support mutual authentication.
3.[must] share state equivalence so that both parties share the same state of authentication protocol and attributes.
4.[must] be resistant to dictionary attacks and man-in-the-middle attacks.
5.[must] protect the negotiation of the ciphersuite parameters that serve to protect later EAP data exchanges.
6.[should] support the fragmentation of EAP data in case of too large EAP data.
7.[should] hide end-user's identity for confidentiality purpose.
8.[might] support fast reconnect procedures for efficiency.

Fig. 5 List of mandatory ([must]), recommended ([should]), or optional ([might]) properties given in [23] for EAP methods to be eligible to IEEE 802.11 wireless LAN environment

EAP-PSK [17] provides mutual authentication and key derivation based on a pre-shared key. A key hierarchy is defined where two static long-lived keys, 128-bit authentication key (AK) and key derivation key (KDK), are derived from the pre-shared key by both the client and the server. AK serves for server and client to mutually authenticate, thanks to message integrity check (MIC) adjunction to EAP messages. KDK serves to derive other keys that are valid during the EAP session only and that ensure a secure EAP channel (integrity and confidentiality protection). No negotiation of the ciphersuite parameters for securing that EAP channel is provided. Note that AK is not a session key but a static key that directly supports generation of MIC. As such, EAP-PSK is vulnerable to brute-force attacks targeting MIC and to AK key cracking.

Like EAP-PSK, EAP-SAKE [18] assumes a pre-shared key is known to both the client and the server, and it supports mutual authentication and key derivation. The pre-shared key serves to derive two session keys: TEK_Auth for generating the MIC for mutual authentication and TEK_Cipher to optionally encrypt part of the EAP data. The MIC generation function is fixed by the EAP-SAKE method. Only the encryption algorithm for EAP data encryption is able to be negotiated. Like any other pre-shared key-based EAP method, EAP-SAKE is vulnerable to dictionary attacks if the pre-shared key is a password.

EAP-GPSK [19] is a lightweight shared-key authentication method supporting mutual authentication and key derivation and requiring two round-trip times. The session key serves to integrity protect the messages and the protected data encryption key to optionally encrypt some confidential parameters. Any of the algorithms including the key derivation function can be negotiated.

EAP-PAX [20] is based on a pre-shared key that is combined to Diffie-Hellman key exchange for derivation of session keys. The confirmation key is useful for MIC generation by both parties to support mutual authentication. The integrity check key enables each EAP-PAX message to be integrity protected. Two modes of operation are defined: PAX-STD and PAX-SEC. PAX-STD is the basic mode realized in two roundtrip times with simple MICs exchange. PAX-SEC is more secure offering confidentiality of client's identity. PAX-SEC assumes the server owns a public key for the client to encrypt its identity. Note that identity protection is securely provided only if the server's certificate is signed by a trusted authority known by the client.

EAP-POTP [21] is based on one-time password (OTP) tokens, and it has two variants. In the basic variant, only the client authenticates to the server. This mode of operation can only be used within a secured tunnel. Thus, it complements tunneled EAP methods like EAP-PEAP, EAP-TTLS, and EAP-FAST where the secured tunnel enables during establishment to authenticate the server and then to protect the encapsulated authentication of the client. A more advanced variant provides mutual authentication, integrity and confidentiality protection of the exchange, and derivation of keying material.

EAP-FAST [22] consists first in establishing a TLS tunnel with the TLS handshake protocol and Diffie-Hellman key exchange and second in handling further authentication through that tunnel. Fast reestablishment of tunnel is supported thanks to a TLS extension.

2.4 Discussions

As shown in Table 1, the methods EAP-TLS, EAP-PAX-SEC, and EAP-FAST make use of electronic certificates and assume that a public key infrastructure (PKI), to manage the public/private keys, is deployed. PKIs are very expensive in terms of management and maintenance. Moreover, providing each terminal with a certificate is very expensive, and heavy operations of distributing certificates to terminals should not be neglected. Finally, usage of certificates for the terminal to authenticate the server does not fit well the context of mobiles. Indeed, authentication occurs first to being connected to the network, so the mobile is not able to check the non-revocation status of the server's certificate and might accept an invalid server's certificate.

EAP-PSK, EAP-MD5, and EAP-PAX do not support the negotiation of the ciphersuite parameters. All those parameters are fixed, and in case one of the parameters is found vulnerable to some attacks, the method itself is classified as weak and no longer usable.

All the pre-shared key-based EAP methods are vulnerable to dictionary attacks in case the pre-shared key is generated from a guessable quantity such as a human-selected password. Having a high entropy for the pre-shared keys is a matter of security policy.

In the wireless environment, rapidity, lightweight, and limited roundtrips are of high importance to preserve terminals' own resources. Pre-shared key mechanisms are light processing. Roundtrips must be limited to save the network's bandwidth, the power for transmission/reception, and to be as fast as possible for users to get connected to the network. As presented in Table 1, most of the EAP methods described before propose an authentication in four to six roundtrip times. EAP-MD5 is the only existing method with only two roundtrip times. However, EAP-MD5 suffers from the implemented unidirectional authentication that make rogue access equipments undetectable to terminals, and EAP-MD5 is vulnerable to dictionary or brute-force attacks.

The scientific community is still searching for new EAP methods adapted to resource constrained and vulnerable environments. The next sections present the EAP-EHash method and compare the authentication delays to those obtained by EAP-MD5 and EAP-TLS. The comparison to EAP-MD5 is interesting as EAP-MD5 is known to be the simplest method, and measures for EAP-TLS give information on how slow one of the most robust methods is.

3 EAP-EHash method description

We propose a new simple, efficient, and easy-to-deploy EAP method called EAP-EHash for use in a resource-constrained network environment. EAP-EHash is based on symmetric cryptography and satisfies all the mandatory properties defined by [23]. It offers mutual authentication and derivation of strong session keys of at least 128 bits length. It defines mechanisms to mitigate dictionary, brute-force and man-in-the-middle attacks. Negotiation of the ciphersuite to protect EAP exchanges is included and is protected, as explained in end of section 3.2.

The simplicity of EAP-EHash and inherent lightweight computation is due to the challenge/response authentication system. The only one prerequisite for such EAP method is the configuration of a PSK between the client and EAP server.

The EAP-EHash authentication process is composed of three main phases (Fig. 6): a negotiation phase (i.e., a handshake between the client and EAP server), an authentication phase, and a session key derivation phase. Each phase is described into details with format of messages in the following sections.

3.1 Negotiation phase

The client (or 802.11 supplicant) and the EAP server need first to negotiate the ciphersuite, that is, the hash function and the encryption algorithm, both being used to generate EAP-EHash challenge/response messages during the authentication procedure (second phase). Note that the negotiation phase might be fully piggybacked into the authentication phase (i.e., requiring no extra message exchanges) in case the default ciphersuite proposed by the EAP server is accepted as it is by the client.

This negotiation phase is essential so as to fit the technical requirements of low-power mobile terminals and to offer the supplicant and the EAP server the possibility to select the appropriate security level for their authentication session. For instance, the EAP server can register many profiles per user, each one corresponding to one of the user's terminals. Selection of the profile during negotiation phase by the EAP server might be done according to the identity declined by the client. A client identity (ClientID) of the form "UserName.Terminal@realm" is one possibility. Furthermore, according to the authenticator forwarding EAP messages, the EAP server can deduce the type of access network (IEEE 802.11, WiMAX...) from where the client is asking connectivity. As such, the security level can also be adapted according to the access network type that is known as more or less vulnerable to attacks. One step further of this discussion that is worth proposing but out of scope of the article is the possibility for selecting the EAP method itself according to the same criteria.

As illustrated in Figs. 6 and 7, the first exchanges including the negotiation are as follows:

1. Upon receiving the EAP-Identity-Request message (as seen in Fig. 6), the client sends back its identity in an EAP-Identity-Response message.
2. Once having received the client's identity, the intermediate authenticator forwards the message to an EAP server.
3. The EAP server then sends back an EAP-Request packet containing a default EAP-EHash Challenge data to the client through the authenticator. This message contains an "Algo" field that contains the default ciphersuite configured at the EAP server.
4. The client then checks the "Algo" field and answers with EAP-response message. In case of positive negotiation response, that message is part of the authentication phase itself (Section 3.2), and the negotiation phase is completed. Otherwise, the client returns the list of supported functions (hash function and encryption algorithm) in the "Algo" field, and the EAP server detects the unapproved ciphersuite by checking if the "Algo" value is different than the one proposed.
5. In case of the first unapproved ciphersuite, the server sends another EAP-Request message encapsulating a

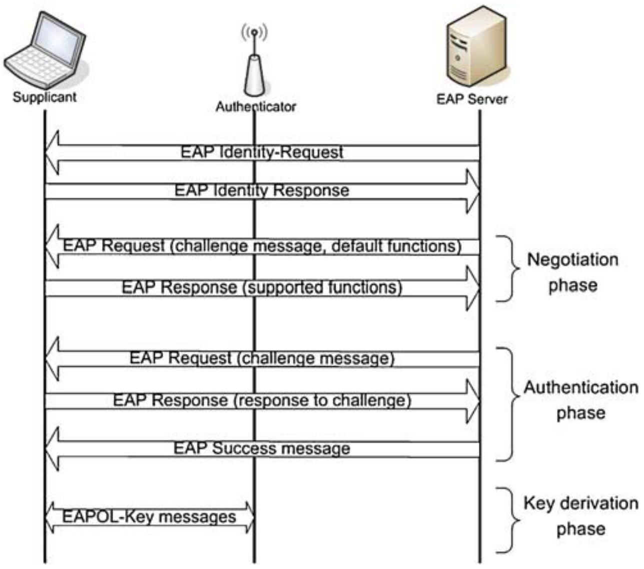


Fig. 6 EAP-EHash main phases

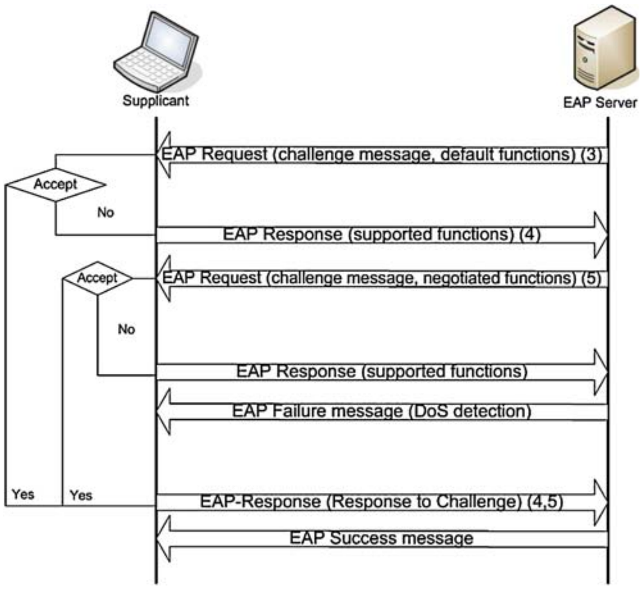

Supplicant
EAP Server
EAP Request (challenge message, default functions) (3)
Accept
No
EAP Response (supported functions) (4)
Accept
EAP Request (challenge message, negotiated functions) (5)
No
EAP Response (supported functions)
EAP Failure message (DoS detection)
Yes
Yes
EAP-Response (Response to Challenge) (4,5)
EAP Success message

Fig. 7 EAP-EHash negotiation phase

new ciphersuite proposal that satisfies both the client and server's security policies. Then, the client proceeds to the message treatment similarly to hereabove step 4.

To avoid denial of service (DoS) attacks against the EAP server, only one negotiation is allowed. This prevents attackers from overloading the EAP server by denying every EAP-Request message. If a malicious client attempts to make such a DoS attack, the EAP server must end the EAP-EHash session with an EAP-Failure message.

3.2 Authentication phase

The supplicant and the EAP server are assumed to share a PSK that serves for mutual authentication, and key derivation. The very first EAP Identity-Request and EAP Identity-Response messages (Fig. 8) include the server's identity (ServerID) that might be an IP address, a hostname, a string, etc., and the client's identity (ClientID) that might be a login, a string, etc.

EAP-EHash defines challenge and response messages that are piggybacked into EAP-Request and EAP-Response messages as depicted in Figs. 6 and 7. Both of them comprise an "Algo" value useful for the negotiation phase (Section 3.1).

For generating the challenge message, the EAP server generates two random numbers referred to as Challenge and RandS, and it derives two keys AK and EK as follows:

- AK: AK is derived from PSK as follows.
  AK=F(PSK, RandS)
  where F() is a one-way function like HMAC-SHA1 and HMAC-MD5.
- EK (Encryption Key): EK is next used to encrypt MIC and Hash values. EK is derived from PSK as follows.
  EK=F(PSK, RandS || ServerID || ClientID)
  where || denotes concatenation

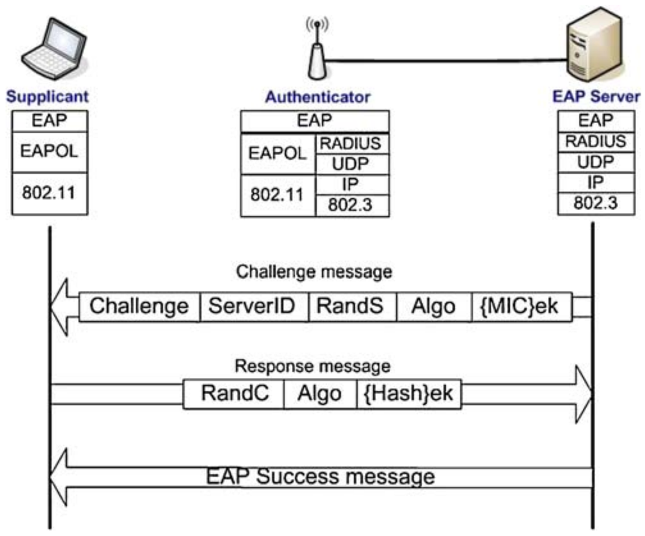


Fig. 8 EAP-EHash authentication phase (in case of successful authentication)

Length of AK and EK keys is implementation dependent but is greater than 128 bits.

Then, the EAP server forges the challenge message (Fig. 8), which comprises the Challenge, ServerID, RandS, ciphersuite (Algo), and a MIC that serves to prove the integrity of the challenge message (including the "Algo" value). The MIC is computed over the message using a one way function F() as follows:

MIC=F(AK, Challenge || ServerID || RandS || Algo)

In order to make brute-force and dictionary attacks more difficult, MIC is also encrypted using EK and the symmetric algorithm. As such, attacks require to derive keys AK and EK to calculate the MIC and then encrypt the resulting MIC with EK.

After receiving the challenge message and in case negotiation is successful, the client generates the same AK and EK keys by itself and authenticates the server by checking the received MIC value is correct. Once the server is successfully authenticated, the client generates a random RandC value and forges a response message that contains RandC, Algo, and a Hash field that is then encrypted with EK.

$$\mathrm{Hash} = \mathrm{F}(\mathrm{AK}, \mathrm{Challenge} \| \mathrm{RandC} \| \mathrm{Algo})$$

Note that the Hash value is computed over the Challenge value to guarantee the server that the response message is fresh, i.e., it was generated in response to the Challenge message.

The EAP server then is able to authenticate the client by checking the correct value of Hash. In case of success, it sends an EAP-Success message to the client, through the authenticator which then grants network access to the client.

As described in Sections 3.1 and 3.2, the ciphersuite negotiation is protected by MIC and Hash that prove integrity and source origin of "Algo" value.

3.3 Key derivation phase

Key derivation is a mechanism providing the client and the authenticator (or any access network equipment) with keying material to secure the traffic being exchanged over the vulnerable radio link. The objective is to prevent eavesdropping and any active attack onto the traffic.

EAP-EHash supports key derivation as depicted in Fig. 9:

1. Once the client is successfully authenticated, the EAP server and the client share a master key MK generated from PSK: MK=F(PSK, RandS || RandC).
2. The supplicant and the EAP server then derive a pairwise master key (PMK) using a PRF over MK.

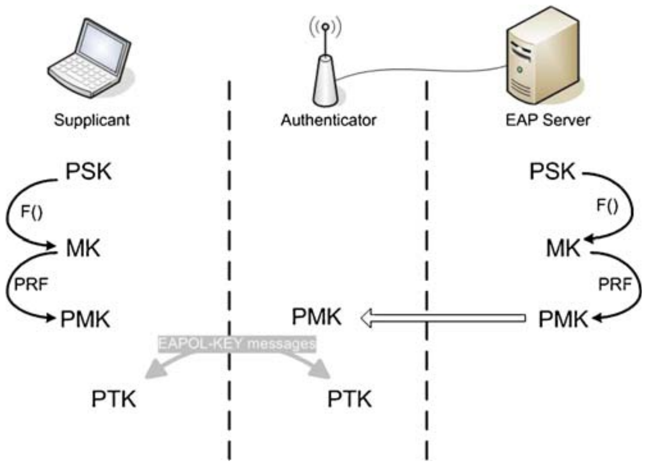
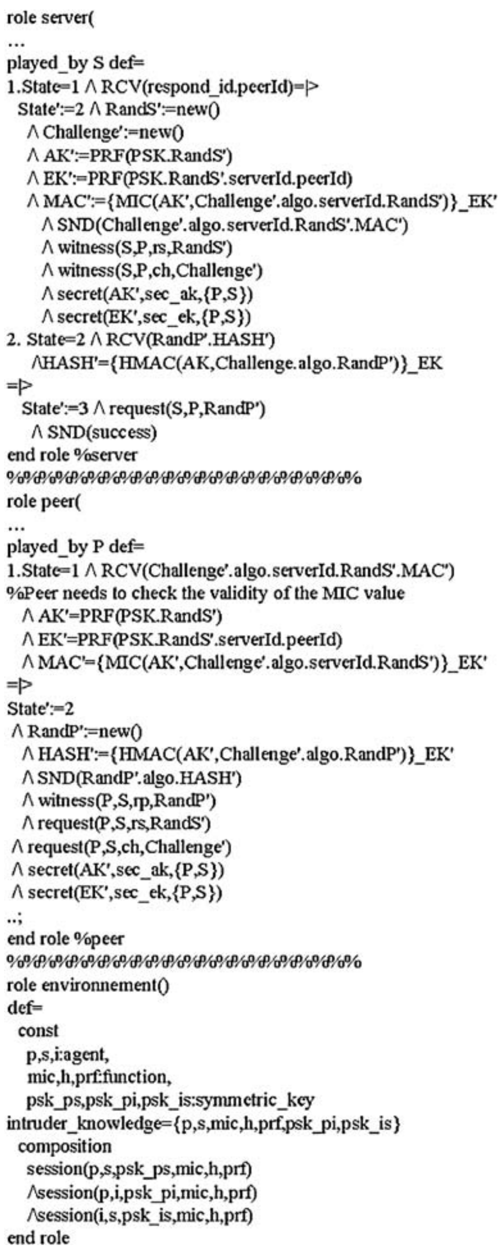


Fig. 9 EAP-EHash key derivation mechanism

3. The EAP server then forwards PMK to the authenticator. In case the EAP server is located on the AAA server, the AAA protocol might support PMK transportation. For instance, RADIUS defined the MS-MPPE Radius attribute for that purpose.
4. A pairwise transient key (PTK) might then be derived from PMK using the IEEE 802.11i four-way handshake protocol [24] between the client and the authenticator. The messages exchanged in this protocol are encapsulated into EAPOL-Key packets. Note that the final PTK is only known to the supplicant and authenticator, even in case of PMK disclosure. Thus, it satisfies the principle of mode independence being defined in the key derivation framework [25].

4 EAP-EHash formal validation

For proving the robustness of the EAP-EHash method, we proceeded to a formal validation of the authentication phase. Many techniques of formal validation can be used like CASPER [26], EVA [27], and automated validation of internet security protocols and applications (AVISPA) [28]. We decided to use AVISPA which is automatic and enough expressive to validate the security properties of authentication methods. AVISPA defines the high-level protocol specification language (HLPSL) language [29] which is based on temporal logic of action. Each entity in EAP-EHash must be specified as a role. For each role, we need to specify the sequence of exchanged messages as transactions. The partial specification of EAP-EHash method is given in Fig. 10, and the following properties are verified:



```
role server(
...
played_by S def=
1.State=1 ∧ RCV(respond_id.peerId)=|>
 State':=2 ∧ RandS':=new()
  ∧ Challenge':=new()
  ∧ AK':=PRF(PSK.RandS')
  ∧ EK':=PRF(PSK.RandS'.serverId.peerId)
  ∧ MAC':={MIC(AK',Challenge'.algo.serverId.RandS')}_EK'
    ∧ SND(Challenge'.algo.serverId.RandS'.MAC')
    ∧ witness(S,P,rs,RandS')
    ∧ witness(S,P,ch,Challenge')
    ∧ secret(AK',sec_ak,{P,S})
    ∧ secret(EK',sec_ek,{P,S})
2. State=2 ∧ RCV(RandP'.HASH')
   ∧HASH'={HMAC(AK,Challenge.algo.RandP')}_EK
=|>
  State':=3 ∧ request(S,P,RandP')
   ∧ SND(success)
end role %server
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role peer(
...
played_by P def=
1.State=1 ∧ RCV(Challenge'.algo.serverId.RandS'.MAC')
%Peer needs to check the validity of the MIC value
  ∧ AK'=PRF(PSK.RandS')
  ∧ EK'=PRF(PSK.RandS'.serverId.peerId)
  ∧ MAC'={MIC(AK',Challenge'.algo.serverId.RandS')}_EK'
=|>
State':=2
 ∧ RandP':=new()
  ∧ HASH':={HMAC(AK',Challenge'.algo.RandP')}_EK'
  ∧ SND(RandP'.algo.HASH')
  ∧ witness(P,S,rp,RandP')
  ∧ request(P,S,rs,RandS')
 ∧ request(P,S,ch,Challenge')
 ∧ secret(AK',sec_ak,{P,S})
 ∧ secret(EK',sec_ek,{P,S})
..;
end role %peer
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
role environnement()
def=
  const
   p,s,i:agent,
   mic,h,prf:function,
   psk_ps,psk_pi,psk_is:symmetric_key
intruder_knowledge={p,s,mic,h,prf,psk_pi,psk_is}
  composition
   session(p,s,psk_ps,mic,h,prf)
   ∧session(p,i,psk_pi,mic,h,prf)
   ∧session(i,s,psk_is,mic,h,prf)
end role
```

Fig. 10 Extract of HLPSL specifications for EAP-EHash

- Mutual authentication: We specify that the server must authenticate the client by the instruction HASH′= {HMAC(AK,Challenge.algo.RandP′)}_EK in the server role. This instruction means that the server must verify that the HASH received by the client is equal to its own locally computed value. Likely, the instruction MAC′= {MIC(AK′,Challenge′.algo.serverId.RandS′)}_EK′ in the client (peer) role specifies that the client must verify that the MAC received by the server is equal to the one calculated locally. This proves that both client and server successfully generated AK and EK keys, and they know the pre-shared secret (PSK).
- Man in the middle protection: a MITM attacker targets sharing a master key (MK_as) with the server and another

one (MK_ac) with the client in order to get full control on encrypted communications. This attack can only be done if parameters used for generating the key are not authenticated. So we verify this property by specifying that each entity must authenticate the received random number. This is done in HLPSL by the two primitives: witness and request. The two instructions, witness (P,S,rp,RandP′) and request (S,P,rp,RandP′), specify that the server must check that the peer is the claimed one in the current session, it reached the expected state, and it agreed on the fresh RandP random value. Likely, witness (S,P,rs,RandS') and request (P,S,rs,RandS') specify that the client must authenticate the random RandS received from the server.

- Replay attack protection: As mentioned above, the primitives witness and request verify that the parameters RandS and RandP are generated for the current session by the claimed entities and are not replayed by an attacker from a previous session. This property was successfully verified on our method.
- Secrecy of keys: In HLPSL, secrecy property is defined with the primitive secret. The two instructions secret (AK',sec_ak,{P,S}) and secret (EK',sec_ek,{P,S}) specify that keys AK and EK must remain secret and known only to the server (S) and the client (P).

5 Implementation and testbed platform

This section presents the implementation and the experimental results of the EAP-EHash authentication procedure latency experimented in an 802.11 wireless architecture. The testbed platform as depicted in Fig. 11 comprises two HP machines equipped with a Centrino 1.7-GHz processor and a 512-MB RAM. Linux Fedora core 6 is used as operating system (kernel 2.6.20). A Cisco Aironet 1100 Access Point serves as the authenticator. FreeRadius software is used as both the EAP server and RADIUS server and the client's software is based on Open1x implementation (XSupplicant).

The implementation of EAP-EHash was integrated as an extension to FreeRadius and XSupplicant softwares. The default ciphersuite (Section 3.1) is fixed to SHA1 hash function and 3DES encryption algorithm, and the list of implemented ciphersuites is given in Table 2. The server's IP address is used as the ServerID and the client's login as the ClientID. The structure of EAP-EHash challenge and response messages is given in Fig. 12. It comprises a 16 bytes Challenge, 8 bytes RandS, 8 bytes RandC, and 1 byte "Algo" value. The four least significant bits of the "Algo" value are relative to the hash functions and the four most significant ones to the encryption algorithms.

The objective of this wireless testbed is to compare the authentication latency of the EAP-EHash method against the two classical EAP methods: EAP-MD5 and EAP-TLS. The reasons for that choice are discussed end of Section 2.4. For each EAP method, 50 authentication latencies have been measured at the client side using "Ethereal" network analyzer (sniffer). For fairness of measurements, the latency corresponds to the difference of time between the receiving "EAP-Success message" and the emitting "EAP-Identity-Response".

The benchmark testing for the EAP-EHash method results in a latency average of 0.026 s against 0.011 and 0.112 s for EAP-MD5 and EAP-TLS, respectively. Note that EAP-EHash measurements were performed with a two extra messages due to the negotiation phase: the server proposes SHA1 and 3DES as default ciphersuite and negotiation of SHA1 and DES is done.

Table 3 summarizes the authentication latencies measured for the three methods. Figure 13 presents the execution time for each of the 50 trials per method. Measurements differ from one trail to another because the processors in terminals are timeshared between the authentication process and any local uncontrolled processes, and at any time during measurements, processors might be solicited by any of those processes. For this reason, we deduced that the authentication delay minimum is more interesting than averages as it represents the closest value to the exact execution time.

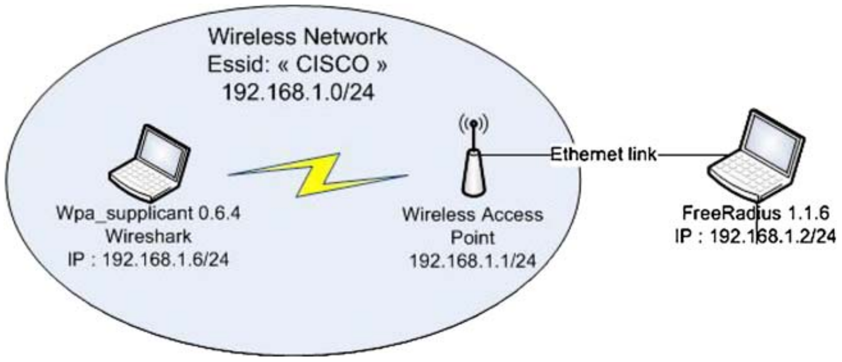


Fig. 11 EAP-EHash 802.11 testbed platform

Table 2 EAP-EHash ciphersuites

Ciphersuite 1 (default)	SHA1, 3DES
Ciphersuite 2	MD5, DES
Ciphersuite 3	SHA1, DES
Ciphersuite 4	MD5, 3DES

As expected, EAP-MD5 is the fastest method as requiring basically only two messages (excluding the EAP identity messages) and two hashing operations. Four messages are needed for EAP-EHash in that testbed (with negotiation phase). Twelve SHA1 hashing functions (four for EK and AK generation in server and client; four for generation of MIC and Hash; four for MIC and Hash checking) and eight DES encryption operations (four for MIC and Hash encryption; four for MIC and Hash decryption) are required. As expected EAP-EHash is slower than EAP-MD5, more than twice slower, but an extra two-way exchange negotiation is included in the results. As such, compared to EAP-MD5 which is really fast, EAP-EHash is very competitive.

EAP-TLS is much more complex than the other two methods. Exchange of six messages and several operations (eight hashing functions, six asymmetric encryption/decryption operations, and four symmetric encryption/decryption operations) are needed. Hence, the obtained testbed results showing that EAP-TLS is the slowest method are consistent with the logical reasoning.

6 EAP-EHash analysis

6.1 EAP-EHash properties

As highlighted by experimental results (Section 5) and the formal validation (Section 4), EAP-EHash is well adapted to wireless networks context as it satisfies the following properties:

- Simplicity and lightness: EAP-EHash which is based on symmetric cryptography functions is not computationally expensive. This property is essential in wireless networks where nodes are resource limited in terms of batteries, computation power, and memory capacities.
- Fast authentication: EAP-EHash is based on a challenge/response authentication mechanism requiring few messages and computation operations. The reduced number of messages ensures a fast authentication. In wireless context, this property is important. Nodes are mobile. At any time, they might lose connectivity. EAP methods are highly likely to be disrupted and ask for the mobile to reinitialize the full procedure.
- Mutual authentication and Protection against MITM attacks: Server and client are authenticating to each other based on a pre-shared key. For the access network, the need is to grant access to authorized clients only. For clients, the need is to mitigate MITM attacks so the client cannot connect to a rogue access point.
- Robustness to brute-force and dictionary attacks: A dictionary attack or brute-force attack is known to be pretty easy when challenge/response, being part of an EAP method, is in clear text: an exhaustive or partial search of possible keys is possible. In EAP-EHash method, those attacks are much more complex because the MIC and Hash digests are encrypted with EK key.
- Symmetric key derivation: A key derivation phase as documented in Section 3.3 is possible for other secure protocols to initialize (like 802.11i four-way handshake or IKEv2).
- Protected ciphersuite negotiation: Negotiation of a ciphersuite to protect EAP exchanges is provided in Section 3.1, and protection of the negotiation is ensured (Section 3.2).
- Fast reconnect: Even if not tested on the platform, the EAP-EHash solution is fully compatible with the re-authentication principle described in the standard EAP re-authentication protocol [30].

6.2 EAP-EHash comparison to other EAP methods and adequacy to wireless environments

EAP-EHash is a challenge/response method like EAP-MD5. EAP-EHash is as fast as EAP-MD5 according to the testbed of Section 5 and is proven by AVISPA as robust as EAP-TLS.

As shown in Table 1, EAP-EHash offers advantages in terms of performances over some other EAP methods designed for wireless. In case no extra roundtrips are requested for negotiation, EAP-EHash authentication is done in only one roundtrip, where other methods require at least two roundtrips like EAP-POTP, EAP-SAKE, and EAP-GPSK. Roundtrips limitation might be important in

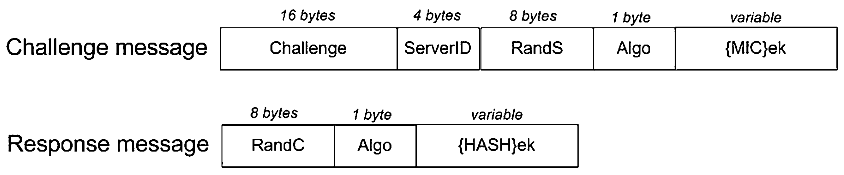


Fig. 12 EAP-EHash 802.11 packet structrures

Table 3 Statistical analysis of authentication latencies with 95% confidence interval (240 samples)

EAP methods	Average	Standard deviation	Minimum	Maximum
EAP-MD5	0.01109	0.00145	0.00912	0.01599
EAP-EHash MD5-DES	0.01817	0.0038	0.01279	0.02825
EAP-EHash MD5-3DES	0.01937	0.00375	0.01369	0.02699
EAP-EHash SHA1-DES	0.02061	0.00493	0.01395	0.04133
EAP-EHash SHA1-3DES	0.01163	0.00120	0.00951	0.01696
EAP-TLS	0.16959	0.01705	0.12653	0.22384

the wireless scenarios where access to the medium takes time. Moreover, total volume of EAP data exchanges must be compared, and EAP-EHash gives good results with only 80 bytes being transmitted over the wireless medium, while EAP-SAKE obtains around 130 bytes, EAP-PSK around 170 bytes, EAP-POTP more than 290 bytes, and EAP-GPSK more than 285 bytes. Processing done by EAP-EHash is lightweight similarly to other methods.

From a security point of view, EAP-EHash was proved to satisfy the claimed security properties—mutual authentication, protection against man-in-the-middle, and replay attacks and secrecy of the keys—thanks to the AVISPA tool. No such kind of validation is known to have been performed on EAP-POTP, EAP-SAKE, EAP-FAST, EAP-PSK, and EAP-GPSK. However, EAP-EHash does not provide identity protection contrary to EAP-PAX-SEC and EAP-FAST, but this property is only stated as "recommended" in the wireless context. EAP-EHash is deduced to be more robust to brute-force attacks than EAP-SAKE, and EAP-GPSK and less robust than EAP-POTP as far as the (usually variable) length of the key is not taken into consideration. That is, more calculations are required for each tested combination of the pre-shared key for EAP-POTP, and lighter calculations are required for EAP-SAKE and EAP-GPSK. For a better security implementation of EAP-EHash, to avoid the pre-shared key to be divulged and cloned into another equipment, the pre-shared key can be safely stored on a physically protected device like a smartcard or a trusted platform module (TPM).

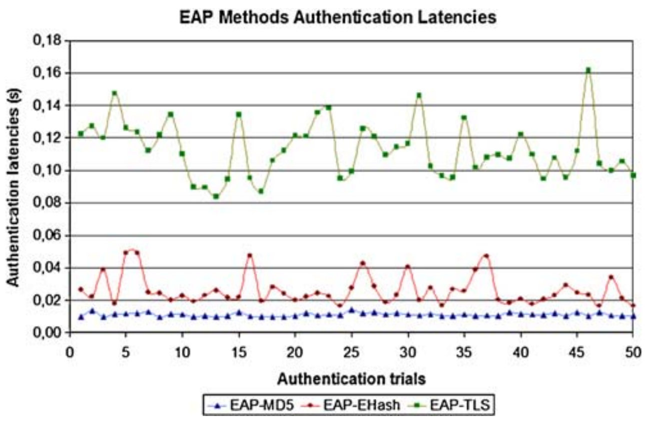


Fig. 13 Authentication latency of each method (over 50 trials)

We can conclude that EAP-EHash is applicable to wireless and mobile network environments for the following reasons:

- Its performances in terms of number of messages, number of transmitted bytes, lightweight computations make EAP-EHash very interesting for use on bandwidth-limited access network (wireless networks), within resource constrained end-devices. That helps getting rapid connectivity to the network, even in case of poor connectivity to the access network.
- It supports ciphersuite negotiation contrary to EAP-PSK and EAP-PAX. This function helps adapting EAP-EHash with suitable algorithms to the capacity of the end-terminals and to the security level awaited on the wireless access link.
- It provides mutual authentication which helps to detect any attacks with rogue access equipments.
- No prerequisites for PKI make the solution easy-to-deploy and cheaper than EAP-TLS and EAP-FAST. Only one pre-shared key has to be installed onto the mobile terminal, whether as a protected file or through a physically secured device (e.g., TPM, smartcard).
- Like any other EAP methods supporting key derivation, EAP-EHash can benefit from the standard [30] to perform fast reconnect of a mobile terminal.

7 Conclusions

Historically, networks were each defined to support one communication usage. There were traditionally separated data networks and voice networks. Each one had its own mechanisms for access control and authentication of users and did not interfere for a long time. For a few years now, operators and Internet access providers have been in competition to provide as many communication services and application contents as possible, with the lowest prices for customers. As a result, all communication services are merged onto the same networks, and only one IP access is enough to access to any communication services. The reason why customers are still paying for several network access subscriptions is the terminal which is still network access technology dependent. However, near future terminals will integrate many access network interfaces, and they will be able to access to any access networks according to the customer's quality of service needs or the prices. Those terminals will be smaller and smaller and more and more resource constraints. As such, CPU consuming mechanisms that were designed in networks because of accessing terminals being resources unlimited need to be changed. Today, there is a strong need to design mechanisms like authentication methods that can satisfy resource constrained terminals and any access network's security.

In this context, our EAP-EHash method was designed to meet strongest constraints of access networks. The resulted EAP-EHash method is simple, fast, easy-to-deploy, robust to MITM attacks, and resistant to dictionary attacks. EAP-EHash serves also to derive some keying material that is very helpful in some vulnerable environments to initialize security protocols (e.g., 802.11i, IKEv2). It supports negotiation of functions used during authentication processing and is compatible with the periodic re-authentication standard [30]. In this article, the formal validation of EAP-EHash using the AVISPA tool is presented that formally proves its security properties like robustness to MITM and brute-force/dictionary attacks. The comparative measurements of authentication delays performed on an 802.11 platform showed that EAP-EHash is fast and competitive compared to the simplest existing EAP-MD5 method.

Acknowledgements We are thankful to Institut TELECOM and TELECOM foundation financing EcoMESH project. We sincerely thank the reviewers for their valuable comments.

References

1. Yegin A, Ohba Y, Penno R, Tsirtsis G, Wang C (2005) Protocol for carrying authentication for network access (PANA) Requirements. IETF RFC 4058, Informational
2. IEEE 802.1X–2004 (2004) IEEE standards for local and metropolitan area networks—port-based network access control. IEEE, Piscataway
3. Forsberg D, Ohba Y, Patil B, Tschofenig H, Yegin A (2008) Protocol for carrying authentication for network access (PANA). IETF RFC 5191, Standards Track
4. Rigney C, Willens S, Rubens A, Simpson A (2000) Remote authentication dial in user service (RADIUS). IETF RFC 2865, Standards Track
5. Aboba B, Calhoun P (2003) RADIUS (remote authentication dial in user service) support for extensible authentication protocol (EAP). IETF RFC 3579, Informational
6. Aboba B, Blunk L, Vollbrecht J, Carlson J, Levkowetz H (2004) Extensible authentication protocol (EAP). IETF RFC 3748, Standards Track
7. Blunk L, Vollbrecht J (1998) PPP extensible authentication protocol (EAP). IETF RFC 2284, Standards Track
8. Kaufman C (2005) Internet key exchange (IKEv2) protocol. IETF RFC 4306, Standards Track
9. Eronen P, Hiller T, Zorn G (2005) Diameter extensible authentication protocol (EAP) application. IETF RFC 4072, Standards Track
10. Dierks T, Rescorla E (2008) The transport layer security (TLS) protocol version 1.2. IETF RFC 5246, Standards Track
11. Dantu R, Clothier G, Atri A (2007) EAP methods for wireless networks. Comput Stand Interfaces 29(3):289–301
12. Lei J, Fu X, Hogrefe D, Tan J (2007) Comparative studies on authentication and key exchange methods for 802.11 wireless LAN. Comput Secur 26(5):401–409
13. Cheikhrouhou O, Laurent-Maknavicius M, Ben Jemaa M (2006) “Nouvelle méthode d'authentification EAP-EHash”, 12ème Colloque Francophone sur l'Ingénierie des Protocoles CFIP'2006, Hermès Science et Publication, ISBN 978-2-7462-1587-0, Tozeur, Tunisie, Octobre 2006
14. Simpson W (2006) PPP challenge handshake authentication protocol (CHAP). IETF RFC 1994, Standards Track
15. Simon D, Aboba B, Hurst R (2008) The EAP-TLS Authentication Protocol. IETF RFC 5216, Standards Track
16. Dierks T, Allen C (1998) The TLS protocol version 1.0. IETF RFC 2246, Standards Track
17. Bersani F, Tschofenig H (2007) The EAP-PSK protocol: a pre-shared key extensible authentication protocol (EAP) Method. IETF RFC 4764, Experimental
18. Vanderveen M, Soliman H (2006) Extensible Authentication protocol method for shared-secret authentication and key establishment (EAP-SAKE). IETF RFC 4763, Informational
19. Clancy T, Tschofenig H (2009) Extensible authentication protocol —generalized pre-shared key (EAP-GPSK) Method. IETF RFC 5433, Standard Track
20. Clancy T, Arbaugh W (2006) Extensible authentication protocol (EAP) password authenticated exchange. IETF RFC 4746, Informational
21. Nystroem M (2007) The EAP protected one-time password protocol (EAP-POTP). IETF RFC 4793, Informational
22. Cam-Winget N, McGrew D, Salowey J, Zhou H (2007) The Flexible authentication via secure tunneling extensible authentication protocol method (EAP-FAST). IETF RFC 4851, Informational
23. Stanley D, Walker J, Aboba B (2005) Extensible authentication protocol (EAP) method requirements for wireless LANs. IETF RFC 4017, Informational
24. IEEE Std 802.11i (2004) IEEE Standard for Information technology—Telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements—part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications. IEEE, Piscataway
25. Aboba B, Simon D, Eronen P (2007) Extensible authentication protocol (EAP) key management framework. IETF RFC 5247, Standards Track
26. Donovan B, Norris P, Lowe G (1999) Analyzing a library of security protocols using casper and FDR. In Workshop on Formal Methods and Security Protocols, Trento, Italy
27. EVA RNTL project, Explication et Vérification Automatique de protocoles cryptographiques. http://www-eva.imag.fr, 2001
28. AVISPA project, http://www.avispa-project.org, 2006
29. Chevalier Y, Compagna L, Cuellar J, Drielsma PH, Mantovani J, Modersheim S, Vigneron L (2004) A high level protocol specification language for industrial security-sensitive protocols. Proc Automated Software Eng 180:193–205
30. Narayanan V, Dondeti L (2008) EAP extensions for EAP re-authentication protocol (ERP). IETF RFC 5296, Proposed standard