Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 4, 2012

V. Cakulev
I. Broustis
Alcatel Lucent

September 1, 2011

An EAP Authentication Method Based on Identity-Based Authenticated Key Exchange

draft-cakulev-emu-eap-ibake-01.txt

Abstract

The Extensible Authentication Protocol (EAP) is an authentication framework which supports multiple authentication methods. This document defines an authentication method for EAP called EAP-IBAKE, which is based on the Identity-Based Authenticated Key Exchange (IBAKE) protocol. The IBAKE method provides mutual authentication through the use of identity-based encryption. In addition to mutual authentication this method also provides perfect forward and backwards secrecy.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on March 4, 2012.

Copyright Notice

Table of Contents

1. Introduction

The Extensible Authentication Protocol (EAP) [RFC3748] provides a standard mechanism for unified support of different authentication methods. EAP has been defined for use with different lower-layer transport methods, such as Point-to-Point Protocol (PPP) [RFC1661], Protocol for Carrying Authentication for Network Access (PANA) [RFC5191], IEEE 802 wired networks [IEEE-802.1X], as well as wireless technologies such as IEEE 802.11 [IEEE-802.11] and IEEE 802.16 [IEEE-802.16]. This document defines a new authentication method for EAP called EAP-Identity Based Authenticated Key Exchange (EAP-IBAKE).

IBAKE is a protocol for mutual authentication and key agreement between two or more endpoints. It is based on a public-key based authentication mechanism, where each message is encrypted with the public key of the corresponding endpoint, as per the Identity Based Encryption (IBE) principles [RFC5091]. As a result of the IBAKE protocol, a shared symmetric key is generated by each endpoint, which can further be used for securing the communication between the endpoints. IBAKE may be applied in a plurality of deployment scenarios that require the generation of a common symmetric key. Hence, IBAKE may be used e.g. for establishing end-to-end secure sessions between peers [I-D.cakulev-mikey-ibake], or for mutually authenticating a peer with a server and deriving a common key. IBAKE offers multiple benefits in terms of facilitating a simplified public-key based mutual authentication and key agreement procedure, which does not depend on the existence of public key infrastructures and the incurred complexities thereof [I-D.cakulev-mikey-ibake]. IBAKE achieves secure mutual authentication between the participants, escrow-free key agreement, as well as perfect forward and backwards secrecy [I-D.cakulev-ibake].

This document specifies IBAKE as a method for the Extensible Authentication Protocol (EAP) [RFC3748]. In the EAP setting that is considered in this document, IBAKE is executed between an EAP peer and an EAP server. While IBAKE may be used for mutual authentication and key agreement between more than two participants, such scenarios are outside the scope of this document.

2. Terminology

2.1. Requirements notation

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 [RFC2119].

2.2. Abbreviations

IBE	Identity Based Encryption
IBAKE	Identity Based Authenticated Key Exchange
IDp	Peer's Identity
IDs	Server's Identity
K_PR	Private Key
K_PUB	Public Key

2.3. Conventions

- E is an elliptic curve over a finite field F
- P is a point on E of large prime order
- [x]P denotes point multiplication on an elliptic curve, i.e. adding a point P to itself total of x times
- H1 is a known hash function that takes a string and assigns it to a point on the elliptic curve, i.e., H1(A) = QA on E, where A is usually based on the identity.
- Encr(k, A) denotes that A is IBE-encrypted with the key k
- s||t denotes concatenation of the strings s and t
- K_PUBx denotes Public Key of x

3. Protocol Description

EAP is a two-party protocol that takes place between an EAP peer and an EAP server (also know as authenticator). An EAP method defines the specific authentication protocol being used by EAP. This document defines IBAKE as the EAP method, i.e., it defines the messages sent between the EAP server and the EAP peer for the purposes of authentication and key derivation.

The peer and the server are attempting to mutually authenticate each other and agree on a key using EAP-IBAKE. This specification assumes that the peer and the server trust a third party, the Key Generation Function (KGF). Rather than a single KGF, several different KGFs may be involved, e.g. one for the peer and one for the server. The peer and the server do not share any credentials prior to execution of EAP-IBAKE. This specification also assumes that the peer and the server have securely obtained their respective Private Keys from their respective KGFs prior to EAP-IBAKE message exchange. The procedures needed to obtain the private keys and public parameters are outside of scope of this specification. The details for these procedures can be found in [RFC5091] and [RFC5408].

3.1. Message Flows

A successful run of EAP-IBAKE consists of up to three message exchanges: an Identity exchange, a Challenge exchange and a Confirm exchange. This is shown in Figure 1.

The peer and the server use the EAP-IBAKE Identity exchange to obtain each other's identities and to agree upon a ciphersuite to use in the subsequent exchanges. In the Challenge exchange the peer and the server exchange information to generate a shared key and also authenticate each other. In the Confirm exchange the peer and the server complete the authentication by proving the knowledge of valid identity-based private keys.

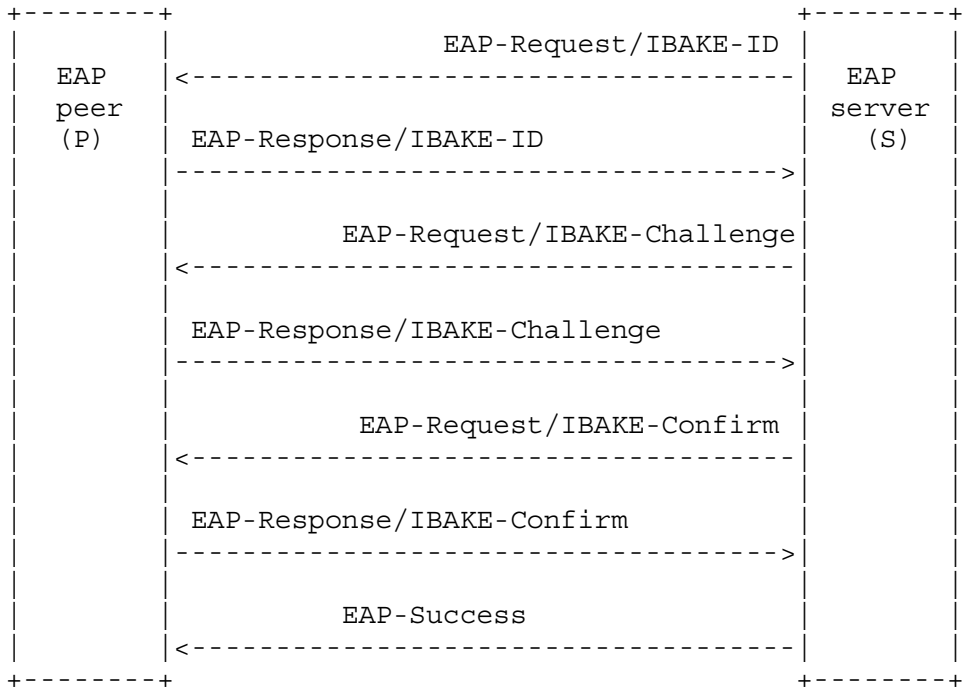

Figure 1: Successful EAP-IBAKE Exchange

The IBAKE exchange, also described in [I-D.cakulev-ibake] is a three-step exchange as follows:

```
Peer                                              Server
----                                              ------

Encr(K_PUBs, IDp || IDs || [Rp]P) ->

              <- Encr(K_PUBp, IDp || IDs || [Rp]P || [Rs]P)

Encr(K_PUBs, IDp || IDs || [Rs]P) ->
```

Where:

- K_PUBp and K_PUBs are the public keys of peer and server, respectively. These public keys are derived from their corresponding identities as follows K_PUBp/s = H1(IDp/s).

- Rp and Rs are random integers, chosen by Peer and Server, respectively.

The EAP-IBAKE method extends the basic IBAKE protocol such that the regular successful EAP-IBAKE exchange takes place as follows.

Message	Server	Peer
ID/Request	IDs, CryptoProposals ->	
ID/Response	<- Encr(K_PUBs,IDp), Encr(K_PUBsCryptoSel)	
Challenge/Request	Encr(K_PUBp, IDs, IDp, [Rs]P) ->	
Challenge/Response	<- Encr(K_PUBp, IDs, IDp, [Rs]P, [Rp]P)	
Confirm/Request	Encr(K_PUBp, IDs, IDp, [Rp]P), Auth_S ->	
Confirm/Response		<- Auth_P

As shown in the exchange above, the following information elements have been added to the original protocol: exchange of identity values for both protocol parties (IDs, IDp), negotiation of cryptographic protocols, signature fields to protect the integrity of the negotiated parameters (Auth_S, Auth_P), and the last message to complete the four-way handshake.

4. Protocol Sequence

This section describes the sequence of messages for the ID, Challenge and Confirm exchanges, and lists the operations performed by the server and the peer.

4.1. IBAKE-ID Exchange

Initially, the server issues EAP-Request/IBAKE-ID, including its identity (IDs) in the Identity payload and optionally ciphersuite proposals in the Proposal payload. Upon receiving the EAP-Request/ IBAKE-ID message the peer uses the received server's identity to generate the server's public key as follows

K_PUBs = H1(IDs).

The peer then encrypts its identity as follows

Encr(K_PUBs, IDp),

and includes it in the Identity payload of the EAP-Response/IBAKE-ID message.

Finally, if the EAP-Request/IBAKE-ID contains Proposals payload, the Peer chooses most preferred proposal, encrypts it as follows

Encr(K_PUBs, Proposal),

and includes it in the Proposal payload of the EAP-Response/IBAKE-ID message.

4.2. EAP-Request/IBAKE-Challenge

Upon receiving EAP-Response/IBAKE-ID message, the server SHALL first decrypt the message as specified in [RFC5091] and [RFC5408], and obtain the identity of the peer. The server then selects a random Rs and computes its Elliptic curve Diffie-Hellman value, [Rs]P. The server MUST use a fresh, random value for Rs on each run of the protocol. The server uses the identity of the peer obtained in IBAKE-ID exchange to generate the IBE public key of the peer as follows

K_PUBp = H1(IDp).

The server then computes the encrypted field (see Section 5.2.2)

ECDHComponent_S = Encr(K_PUBp, IDs || IDp || [Rs]P).

Finally, the server sends EAP-Request/IBAKE-Challenge message to the peer.

4.3. EAP-Response/IBAKE-Challenge

Upon receiving EAP-Request/IBAKE-Challenge message, the peer SHALL first decrypt the message as specified in [RFC5091] and [RFC5408], and obtain [Rs]P. The peer then selects a random Rp and computes its Elliptic curve Diffie-Hellman value, [Rp]P. The peer MUST use a fresh, random value for Rp on each run of the protocol.

The peer then computes the encrypted field (see Section 5.2.2)

ECDHComponent_P = Encr(K_PUBs, IDs || IDp || [Rs]P || [Rp]P).

Finally, the peer sends EAP-Response/IBAKE-Challenge message to the server.

At this point both the Server and Peer can generate the session key [Rs][Rp]P using exchanged Elliptic Curve Diffie-Hellman values. The session key as a point on an elliptic curve is then converted into octet string as specified in [SEC1]. This octet string, K_Session, is then used to generate MSK, EMSK and keys used for protection of IBAKE-Confirm exchange.

4.4. EAP-Request/IBAKE-Confirm

Upon receiving EAP-Response/IBAKE-Challenge message, the server SHALL first decrypt the message as specified in [RFC5091] and [RFC5408], and obtain [Rs]P and [Rp]P. The server MUST verify that the received [Rs]P is the same as the one sent in EAP-Request/IBAKE-Challenge message. If it is not the same, the server MUST abort the protocol with an "Authentication Failure" code. Upon successful verification of the received [Rs]P, the server computes the encrypted field

ECDHComponent_S = Encr(K_PUBp, IDs || IDp || [Rp]P).

The server then computes:

Ka = prf(K_Session, "EAP-IBAKE Ka" || IDs || IDp)

whose length is the preferred key length of the negotiated pseudo-random function (see Section 5.2). It then constructs:

```
Auth_S = prf(Ka, "EAP-IBAKE server" || EAP-Request/IBAKE-ID ||
         EAP-Response/IBAKE-ID || EAP-Request/IBAKE-Challenge ||
         EAP-Response/IBAKE-Challenge).
```

The messages in Auth_S computation are included in full, starting with the EAP header, and including any possible future extensions.

The server then constructs and sends EAP-Request/IBAKE-Confirm message to the peer.

4.5. EAP-Response/IBAKE-Confirm

Upon receiving EAP-Request/IBAKE-Confirm message, the peer SHALL first decrypt the message as specified in [RFC5091] and [RFC5408], and obtain [Rp]P. The peer MUST verify that the received [Rp]P is the same as the one sent in EAP-Response/IBAKE-Challenge message. If it is not the same, the peer MUST abort the protocol with an "Authentication Failure" code. The peer also MUST verify the correctness of the Auth_S value, and MUST abort the protocol if incorrect, with an "Authentication Failure" code.

Upon successful verification of the received [Rp]P and correctness of the Auth_S value, the peer computes Ka as described above, and constructs:

```
Auth_P = prf(Ka, "EAP-IBAKE peer" || EAP-Request/IBAKE-ID ||
         EAP-Response/IBAKE-ID || EAP-Request/IBAKE-Challenge ||
         EAP-Response/IBAKE-Challenge).
```

The peer sends verification message EAP-Response/IBAKE-Confirm to complete the four-way handshake, including AUTH_P value. Upon receiving the message, the server MUST verify the correctness of the Auth_P value, and MUST abort the protocol if incorrect, with an "Authentication Failure" code.

4.6. MSK and EMSK

The authenticated key exchange of EAP-IBAKE generates a shared and authenticated key, K_Session. EAP-IBAKE must export two 512-bit keys, MSK and EMSK. Therefore, following the last message of the protocol, both sides compute and export the shared keys, each 512 bits in length:

```
MSK || EMSK = prf(K_Session, "EAP-IBAKE Exported Keys" ||
              IDs || IDp)
```

At this point, both protocol participants MUST discard all intermediate cryptographic values, including Rs and Rp. Similarly, both parties MUST immediately discard these values whenever the protocol terminates with a failure code or as a result of timeout.

5. Message Formats

EAP-IBAKE defines new message types, with each message consisting of a header followed by a payload. This section defines the header, several payload formats, as well as the format of specific fields within the payloads.

All multi-octet strings MUST be laid out in network byte order.

5.1. EAP-IBAKE Header

The EAP-IBAKE header consists of the standard EAP header (see Section 4 of [RFC3748]), followed by an EAP-IBAKE exchange type. The header has the following structure:

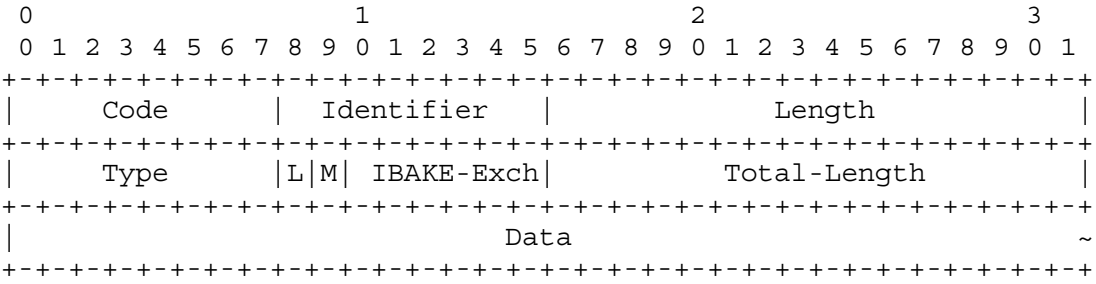

Figure 2: EAP-IBAKE Header

The Code, Identifier, Length, and Type fields are all part of the EAP header as defined in [RFC3748]. The Type field in the EAP header is [TBD by IANA] for EAP-IBAKE version 1.

L and M bits: The L bit (Length included) is set to indicate the presence of the two-octet Total-Length field, and MUST be set for the first fragment of a fragmented EAP-IBAKE message or set of messages. The M bit (more fragments) is set on all but the last fragment.

The IBAKE-Exch (IBAKE Exchange) field identifies the type of EAP-IBAKE payload encapsulated in the Data field. This document defines the following values for the IBAKE-Exch field:

- 0x00: Reserved
- 0x01: IBAKE-ID exchange
- 0x02: IBAKE-Challenge exchange

- 0x03: IBAKE-Confirm exchange

- 0x04: IBAKE-Failure message

Further values of this IBAKE-Exch field are available via IANA registration.

Total-Length: The Total-Length field is two octets in length, and is present only if the L bit is set. This field provides the total length of the EAP-IBAKE message or set of messages that is being fragmented.

5.2. EAP-IBAKE Payloads

EAP-IBAKE messages all contain the EAP-IBAKE header and information encoded in a single payload, which differs for each message. This section defines payloads for EAP-IBAKE messages.

5.2.1. The IBAKE-ID Payload

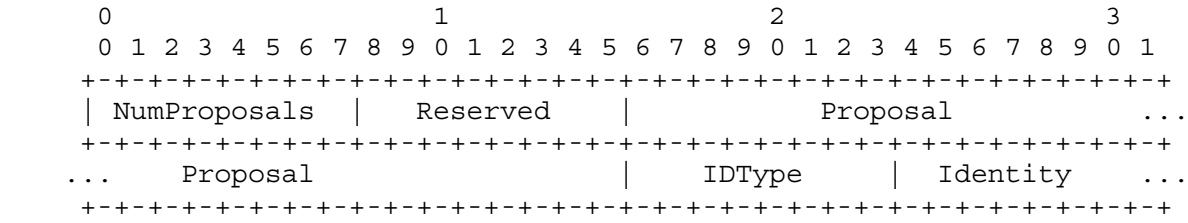

Figure 3: IBAKE-ID Payload

The IBAKE-ID payload contains the following fields:

- NumProposals: The NumProposals field contains the number of Proposal fields subsequently contained in the payload. If in the IBAKE-ID/Request the NumProposals field is set to zero (0) then it is out of scope of this specification how the server and the peer negotiate the elliptic curve and PRF used in the rest of EAP-IBAKE exchange. Otherwise, if in the IBAKE-ID/Request the NumProposals field is not set to zero (0), then in the IBAKE-ID/Response message the NumProposals field MUST be set to one (1). The offered proposals in the Request are listed in priority order, with the most preferable appearing first. The selected proposal in the Response MUST be fully identical with one of the offered proposals.

- Proposal: Each proposal consists of three fields, in this order:
  - Elliptic Curve Description: This field's value is taken from the IANA registry for Elliptic curves defined in Section 7.1. The length of this field is one octet.
  - PRF: This field's value is taken from the IANA registry for pseudo random functions defined in Section 7.2. The length of this field is one octet.
  - P: This field contains the 2-dimensional coordinates of the selected point P on the Elliptic Curve. The length of each coordinate in octets depends on the chosen Elliptic Curve.
- Reserved: By default, this field MUST be sent as zero, and MUST be ignored by the recipient.
- IDType: Denotes the Identity Type. This is taken from the IANA registry defined in Section 7.3. The server and the peer MAY use different identity types. All implementations MUST be able to receive two identity types: ID_NAI and ID_FQDN.
- Identity: The meaning of the Identity field depends on the values of the Code and IDType fields.
  - IBAKE-ID/Request: server ID
  - IBAKE-ID/Response: peer ID

  The length of the Identity field MUST be computed from the Length field in the EAP header. This field, like all other fields in this specification, MUST be octet-aligned.

5.2.2. The IBAKE-Challenge Payload

This payload allows both parties send their encrypted ephemeral keys.

In addition, a small amount of data can be included by the server and/or the peer, and used for channel binding. This data is sent in IBAKE-Challenge exchange unprotected, but is verified when it is signed by the Auth_S/Auth_P payloads of the IBAKE-Confirm exchange.

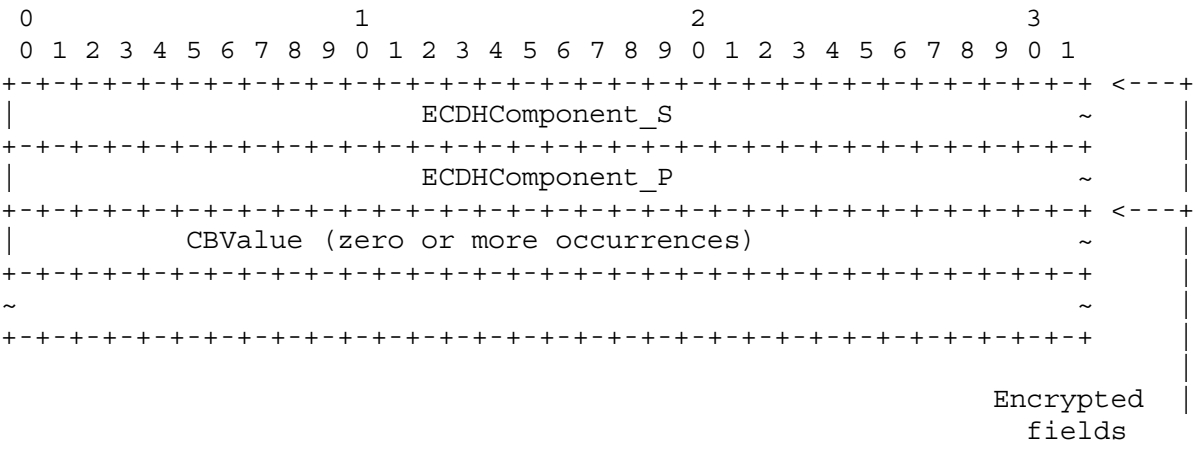

Figure 4: IBAKE-Challenge Payload

The IBAKE-Challenge payload contains the following fields:

- ECDHComponent_S/ECDHComponent_P: This field contains Server's/ Peer's IBE-encrypted Elliptic Curve Diffie-Hellman value. The peer's Elliptic Curve Diffie-Hellman value DHComponent_P is included only in the response message (i.e. EAP-Response/ IBAKE-Challenge). ECDHComponent field contains the identities of the server and the peer, as exchanged in IBAKE-ID exchange.

- CBValue: This structure MAY be included both in the request and in the response, and MAY be repeated multiple times in a single payload (see Section 5.2.5). Each structure contains its own length. The field is neither encrypted nor integrity protected; instead it is protected by the AUTH payloads in the Confirm exchange.

Note that as shown in Figure 4, Identity fields and ECDHComponent fields are IBE-Encrypted.

5.2.3. The IBAKE-Confirm Payload

Using this payload, both parties complete the authentication by mutually authenticating each other and generating key material for the EAP consumer protocol.

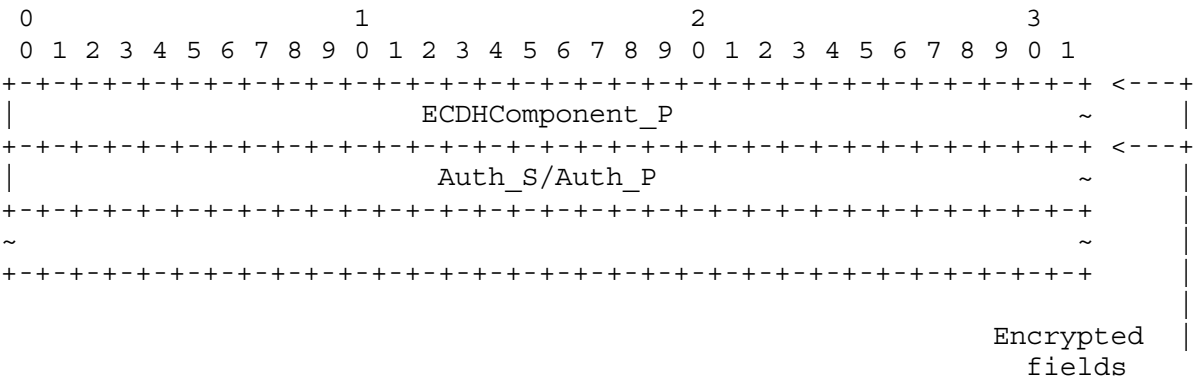

Figure 5: IBAKE-Confirm Payload

The IBAKE-Confirm payload contains the following fields:

- ECDHComponent_P: This field contains the peer's IBE-encrypted Elliptic Curve Diffie-Hellman value. The peer's Elliptic Curve Diffie-Hellman value DHComponent_P is included only in the request message (i.e. EAP-Request/IBAKE-Confirm). ECDHComponent field contains the identities of the server and the peer, as exchanged in IBAKE-ID exchange.

- Auth_S/Auth_P: This field contains a signature of the preceding messages, including the Identity and the negotiated fields. This prevents various possible attacks, such as algorithm-downgrade attacks. Auth_S is included only in the request message (i.e. EAP-Request/IBAKE-Confirm), while Auth_P is included only in the response message (i.e. EAP-Response/IBAKE-Confirm).

5.2.4. The IBAKE-Failure Payload

The EAP-EKE-Failure payload format is defined as follows:

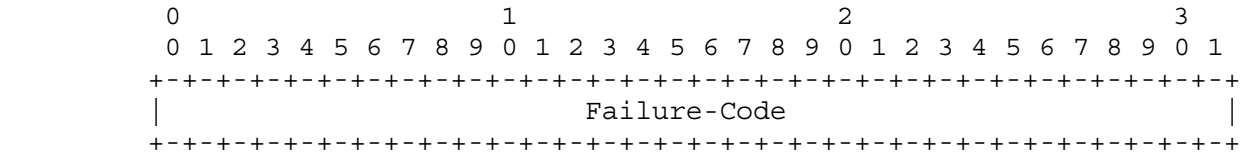

Figure 6: IBAKE-Failure Payload

The payload's size is always exactly 4 octets.

The following Failure-Code values are defined:

Value	Name	Meaning
0x00000000	Reserved	
0x00000001	No Error	This code is used for failure acknowledgement, see below.
0x00000002	Protocol Error	A failure to parse or understand a protocol message or one of its payloads.
0x00000003	Authentication Failure	Failure in the cryptographic computation.
0x00000004	Authorization Failure	While the cryptographic computation is correct, the user is not authorized to connect.
0x00000005	No Proposal Chosen	The peer is unwilling to select any of the cryptographic proposals offered by the server.

Additional values of this field are available via IANA registration.

When the peer encounters an error situation, it MUST send IBAKE-Failure. The server MUST reply with an EAP-Failure message to end the exchange.

When the server encounters an error situation, it MUST send IBAKE-Failure. The peer MUST send back IBAKE-Failure message containing a "No Error" failure code. Then the server MUST send an EAP-Failure message to end the exchange.

5.2.5. Channel Binding Values

This protocol allows higher level protocols that are using it to transmit opaque information between the peer and the server. This information is integrity protected but not encrypted, and MAY be used to ensure that protocol participants are identical at different protocol layers. See Section 7.15 of [RFC3748] for more details on its use.

EAP-IBAKE neither validates nor makes any use of the transmitted information. The information MUST NOT be used by the consumer protocol until it is verified in the IBAKE-Confirm exchange (specifically, it is integrity protected through the use of the Auth_S, Auth_P payloads). Consequently, it MUST NOT be relied upon in case an error occurs at the EAP-IBAKE level.

An unknown Channel Binding Value SHOULD be ignored by the recipient.

Some implementations may require certain values to be present, and will abort the protocol if they are not. Such policy is out of scope of the current specification.

Each Channel Binding Value is encoded using a simple TLV structure:

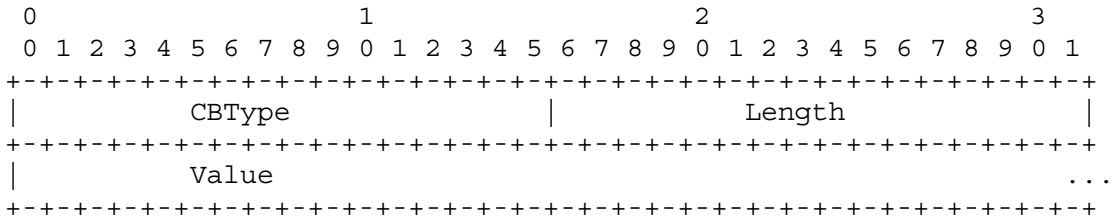

Figure 7: Channel Binding Value

- CBType: This is the Channel Binding Value's type. This document defines the value 0x0000 as reserved. Other values are available for IANA allocation.

- Length: This field is the total length in octets of the structure, including the CBType and Length fields.

6. Fragmentation

EAP [RFC3748] is a request-response protocol. The server sends requests and the peer responds to those requests. These request and response messages are assumed to be limited to at most 1020 bytes. Messages in EAP-IBAKE can be larger than 1020 bytes and therefore require support for fragmentation and reassembly.

Since EAP is a simple ACK-NAK protocol, fragmentation support can be added as follows. In EAP, fragments that are lost or damaged in transit will be retransmitted, and sequencing information is provided by the Identifier field in EAP.

EAP-IBAKE fragmentation support is provided through addition of a "flags" octet within the EAP-Response and EAP-Request packets, as well as a Total-Length field of four octets. Flags include the Length included (L) and More fragments (M) bits. The L flag is set to indicate the presence of the two-octet Total-Length field, and MUST be set for the first fragment of a fragmented EAP-IBAKE message or set of messages. The M flag is set on all but the last fragment. The Total-Length field is two octets, and provides the total length of the EAP-IBAKE message or set of messages that is being fragmented; this simplifies buffer allocation.

When an EAP-IBAKE peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-IBAKE and no data. This serves as a fragment ACK. The EAP server MUST wait until it receives the EAP-Response before sending another fragment. In order to prevent errors in processing of fragments, the EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the fragment ACK contained within the EAP-Response. Retransmitted fragments will contain the same Identifier value.

Similarly, when the EAP server receives an EAP-Response with the M bit set, it MUST respond with an EAP-Request with EAP-Type=EAP-IBAKE and no data. This serves as a fragment ACK. The EAP peer MUST wait until it receives the EAP-Request before sending another fragment. In order to prevent errors in the processing of fragments, the EAP server MUST increment the Identifier value for each fragment ACK contained within an EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment contained within an EAP-Response.

7. IANA Considerations

IANA shall allocate the EAP method type TBD from the range 1-191, for "EAP-IBAKE Version 1".

This document requests that IANA create the registries described in the following sub-sections.

7.1. Elliptic Curve Registry

IANA is to create and maintain Elliptic Curve Registry. Registry consists of an ECC curve and its associated value. Values in the range 1-239 SHOULD be approved by the process of Specification Required, values in the range 240-254 are for Private Use, and the values 0 and 255 are Reserved according to [RFC5226]. The initial contents of the registry should be as follows:

Value	ECC curve
0	Reserved
1	ECPRGF192Random / P-192 / secp192r1
2	EC2NGF163Random / B-163 / sect163r2
3	EC2NGF163Koblitz / K-163 / sect163k1
4	EC2NGF163Random2 / none / sect163r1
5	ECPRGF224Random / P-224 / secp224r1
6	EC2NGF233Random / B-233 / sect233r1
7	EC2NGF233Koblitz / K-233 / sect233k1
8	ECPRGF256Random / P-256 / secp256r1
9	EC2NGF283Random / B-283 / sect283r1
10	EC2NGF283Koblitz / K-283 / sect283k1
11	ECPRGF384Random / P-384 / secp384r1
12	EC2NGF409Random / B-409 / sect409r1
13	EC2NGF409Koblitz / K-409 / sect409k1
14	ECPRGF521Random / P-521 / secp521r1
15	EC2NGF571Random / B-571 / sect571r1
16	EC2NGF571Koblitz / K-571 / sect571k1
17-239	Unassigned
240-254	Private Use
255	Reserved

7.2. Pseudo Random Function Registry

This section defines an IANA registry for pseudo random function algorithms:

Name	Value	Definition
Reserved	0	
PRF_HMAC_SHA1	1	HMAC SHA-1, as defined in [RFC2104]
PRF_HMAC_SHA2_256	2	HMAC SHA-2-256 [SHA]
	3-127	Available for allocation via IANA
	128-255	Reserved for private use

7.3. Identity Type Registry

In addition, an identity type registry is defined:

Name	Value	Definition
Reserved	0	
ID_OPAQUE	1	An opaque octet string
ID_NAI	2	A Network Access Identifier, as defined in [RFC4282]
ID_IPv4	3	An IPv4 address, in binary format
ID_IPv6	4	An IPv6 address, in binary format
ID_FQDN	5	A fully qualified domain name, see note below
ID_DN	6	An LDAP Distinguished Name formatted as a string, as defined in [RFC4514]
	7-127	Available for allocation via IANA
	128-255	Reserved for private use

An example of an ID_FQDN is "example.com". The string MUST NOT contain any terminators (e.g., NULL, CR, etc.). All characters in the ID_FQDN are ASCII; for an "internationalized domain name", the syntax is as defined in [RFC5891], for example "xn--tmonesimerkki-bfbb.example.net".

7.4. EAP-IBAKE Channel Binding Type Registry

This section defines an IANA registry for the Channel Binding Type registry, a 16-bit long code. The value 0x0000 has been defined as Reserved. All other values up to and including 0xfeff are available for allocation via IANA. The remaining values up to and including 0xffff are available for private use.

7.5. Exchange Registry

This section defines an IANA registry for the EAP-IBAKE Exchange registry, an 8-bit long code. Initial values are defined in Section 5.1. All values up to and including 0x7f are available for allocation via IANA. The remaining values up to and including 0xff are available for private use.

7.6. Failure-Code Registry

This section defines an IANA registry for the Failure-Code registry, a 32-bit long code. Initial values are defined in Section 5.2.4. All values up to and including 0xfeffffff are available for allocation via IANA. The remaining values up to and including 0xffffffff are available for private use.

8. Security Considerations

This section discusses the claimed security properties of EAP-IBAKE as well as vulnerabilities and security recommendations.

8.1. Identity Protection

EAP-IBAKE includes identity privacy support that protects the privacy of the subscriber identity against passive eavesdropping. Observe that in all the messages, the identity of the peer is sent in the encrypted part of the payload, therefore an attacker cannot discover user identities by snooping authentication traffic.

8.2. Mutual Authentication

Payloads in the protocol exchange are encrypted using IBE. In particular, only the peer can decrypt the contents of the ECDHComponent_S payload sent by the server, and similarly only the server can decrypt the contents of the ECDHComponent_P and the encrypted part of the EAP-Response/IBAKE-ID payload sent by the peer. Moreover, upon receiving EAP-Response/IBAKE-Challenge, the server can verify the authenticity of the peer, since [Rs]P could have been sent in EAP-Response/IBAKE-Challenge only after decryption of the contents of EAP-Request/IBAKE-Challenge by the peer. Similarly, upon receiving EAP-Request/IBAKE-Confirm, the peer can verify the authenticity of the server, since [Rp]P could have been sent back in EAP-Request/IBAKE-Confirm only after correctly decrypting the contents of EAP-Response/IBAKE-Challenge and this is possible only by the server. In other words, the protocol provides mutual authentication based on Identity Based Encryption.

8.3. Key Derivation

EAP-IBAKE supports key derivation with 512-bit effective key strength. The key hierarchy is specified in Section 4.

The transient EAP key used to protect EAP-IBAKE packets (Ka), the Master Session Keys, and the Extended Master Session Keys are cryptographically separate. An attacker cannot derive any non-trivial information about any of these keys based on the other keys.

8.4. Brute-Force and Dictionary Attacks

The effective strength of EAP-IBAKE values is 512 bits, and there are no known, computationally feasible brute-force attacks to date. Because IBAKE is not a password protocol (the private keys are not a passphrase, or derived from a passphrase), EAP-IBAKE is not vulnerable to dictionary attacks.

8.5. Protection, Replay Protection, and Confidentiality

ECDHComponent and Auth payloads are used to provide integrity, replay, and confidentiality protection for EAP-IBAKE Requests and Responses. Integrity protection with Auth includes the EAP header. Integrity protection (Auth) is based on a keyed message authentication code. Confidentiality (ECDHComponent) is based on Identity Based Encryption.

Because the session key is not available in the beginning of the EAP method, the Auth payload cannot be used for protecting EAP/IBAKE-ID and EAP/IBAKE-Challenge messages. However, the EAP/IBAKE-ID and EAP/IBAKE-Challenge exchanges are integrity protected through the Auth payloads exchanged in the Confirm exchange.

Confidentiality protection is applied only to a part of the protocol fields. Section 4 describes in detail which fields are confidentiality protected. Identity protection is discussed in Section 8.1.

Because EAP-IBAKE is not a tunneling method, EAP-Request/Notification, EAP-Response/Notification, EAP-Success, or EAP-Failure packets are not confidentiality protected, integrity protected, or replay protected. On physically insecure networks, this may enable an attacker to mount denial-of-service attacks by spoofing these packets. However, the peer will only accept EAP-Success after the peer successfully authenticates the server. Hence, the attacker cannot force the peer to believe that successful mutual authentication has occurred before the peer successfully authenticates the server, or after the peer fails to authenticate the server.

An eavesdropper will see the EAP Notification, EAP-Success and EAP-Failure packets sent in the clear. With EAP-IBAKE, confidential information MUST NOT be transmitted in EAP Notification packets.

8.6. Trust Model

It is assumed that the KGFs that are used for deriving IBE private keys are secure, not compromised, trusted, and will not engage in launching active attacks independently or in a collaborative environment. However, any malicious insider could potentially launch passive attacks (by decryption of one or more message exchanges offline). While it is in the best interest of administrators to prevent such threats, it is hard to eliminate this problem. Hence, it is assumed that such problems will persist, and hence the session key agreement protocols are designed to protect participants from passive adversaries.

The EAP peer and the EAP server need to trust their respective KGF entities. Such a KGF may be owned/operated by third parties; in such cases, the peer and the server need to maintain trust relationships with those third parties. Note here that in scenarios where the EAP peer and the EAP server are associated with the same KGF, while such a KGF is owned by the server owner (or operator), there is no implicit or explicit trust to third parties.

In addition, it is assumed that the communication between participants and their respective KGFs is secure. Therefore, in any implementation of the protocols described in this document, administrators of any KGF have to ensure that communication with participants is secure and not compromised.

8.7. Forward Secrecy

The MSK and EMSK are extracted from the session key, K_Session, which is derived from exchanged Elliptic Curve Diffie-Hellman values. The peer and the server choose random values with each run of the protocol. Hence, even if an attacker is able to somehow obtain the session key from an earlier run, the attacker will be unable to determine any future session keys, or the MSK or EMSK. In other words, EAP-IBAKE provides perfect forward secrecy.

8.8. Reflection Attacks

The use of identities within the encrypted payload is intended to eliminate some basic reflection attacks. For instance, suppose that identities were not used as part of the encrypted payload, in EAP-Request/IBAKE-Challenge message. Furthermore, assume an adversary who has access to the conversation between the server and peer and can actively snoop into packets and drop/modify them before routing them to the destination. As an example, consider the case where the IP source address and destination address can be modified by the adversary. After the EAP-Request/IBAKE-Challenge message is sent by the server (to the peer), the adversary can take over and trap the packet. Next, the adversary can modify the IP source address to include the adversary's IP address, before routing it onto the responder. The peer will assume that the request for an IBAKE session came from the adversary, and will send EAP-Response/ IBAKE-Challenge, but encrypt it using the adversary's public key. The above message can be decrypted by the adversary (and only by the adversary). In particular, since the EAP-Request/IBAKE-Challenge message includes the challenge sent by the server to the peer, the adversary will now learn the challenge sent by the server. Following this, the adversary can carry on a conversation with the server "pretending" to be the peer. This attack is eliminated with EAP-IBAKE, since identities are used as part of the encrypted payload.

8.9. Negotiation Attacks

EAP-IBAKE does not protect the EAP-Response/Nak packet. Given that EAP-IBAKE does not protect the EAP method negotiation, EAP method downgrading attacks may be possible. In addition, EAP-IBAKE does not support EAP-IBAKE protocol version negotiation.

EAP-IBAKE supports ciphersuite negotiation through the use of Proposal payload during IBAKE-ID exchange. The ciphersuite proposal made by the server is not protected from tampering by an active attacker. However, the Proposal payload in EAP-Response/IBAKE-ID message containing the peers choice of ciphersuite is sent in the encrypted part of the message. Furthermore, if a proposal was modified by an active attacker, it would result in a failure to confirm the message sent by the other party, since the proposal is bound by each side into its Confirm message, and the protocol would fail as a result.

9. Security Claims

This section provides the security claims required by [RFC3748].

Authentication Mechanism: EAP-IBAKE is based on the IBAKE mechanism, which is an authentication and key agreement mechanism based on Identity Based Encryption.

Ciphersuite negotiation: Yes

Mutual authentication: Yes (Section 8.2)

Integrity protection: Yes (Section 8.5)

Replay protection: Yes (Section 8.5)

Confidentiality: Yes, except method-specific failure indications (Section 8.5)

Key derivation: Yes

Key strength: EAP-IBAKE supports key derivation with 512-bit effective key strength.

Description of key hierarchy: Please see Section 4.

Dictionary attack protection: N/A (Section 8.4)

Fast reconnect: No

Cryptographic binding: Yes

Session independence: Yes (Section 8.7)

Fragmentation: Yes (Section 6)

Channel binding: Yes

Indication of vulnerabilities: Vulnerabilities are discussed in Section 8.

10. References

10.1. Normative References

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC5091] Boyen, X. and L. Martin, "Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems", RFC 5091, December 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5408] Appenzeller, G., Martin, L., and M. Schertler, "Identity-Based Encryption Architecture and Supporting Data Structures", RFC 5408, January 2009.

[SEC1] Standards for Efficient Cryptography Group, "Elliptic Curve Cryptography", September 2000.

[SHA] Standards for Efficient Cryptography Group, "Secure Hash Standard", Federal Information Processing Standards Publication 180-2, August 2002, with Change Notice 1, February 2004.

10.2. Informative References

[I-D.cakulev-ibake]
Cakulev, V. and G. Sundaram, "IBAKE: Identity-Based Authenticated Key Agreement", draft-cakulev-ibake-02 (work in progress), September 2010.

[I-D.cakulev-mikey-ibake]
Cakulev, V. and G. Sundaram, "MIKEY-IBAKE: Identity-Based Mode of Key Distribution in Multimedia Internet KEYing (MIKEY)", draft-cakulev-mikey-ibake-05 (work in progress), December 2010.

[IEEE-802.11]
"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 Standard 802.11-2007, 2007.

[IEEE-802.16]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks: Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems: Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operations in Licensed Bands", IEEE 802.16e, August 2005.

[IEEE-802.1X]
National Institute for Standards and Technology,, "Local and Metropolitan Area Networks: Port-Based Network Access Control", IEEE Standard 802.1X-2004, December 2004.

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.

Authors' Addresses

Violeta Cakulev
Alcatel Lucent
600 Mountain Ave.
3D-517
Murray Hill, NJ 07974
US

Phone: +1 908 582 3207
Email: violeta.cakulev@alcatel-lucent.com

Ioannis Broustis
Alcatel Lucent
600 Mountain Ave.
3D-526
Murray Hill, NJ 07974
US

Phone: +1 908 582 3744
Email: ioannis.broustis@alcatel-lucent.com