

Internet Draft
Expiration: September 2003
File: [draft-jwalker-eap-archie-01.txt](#)
Expires: December 23, 2003

Jesse Walker
Intel Corporation
Russ Housley
Vigil Security
June 2003

The EAP Archie Protocol

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC2026](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This memo defines an EAP authentication protocol based on pre-shared keys, called EAP-Archie. This protocol performs mutual authentication, session establishment, and session key derivation. EAP-Archie is based on a static pre-shared key. EAP-Archie is designed as a native EAP method.

INTERNET-DRAFT

EAP-Archie

23 June 2003

Table Of Contents

1.	Introduction	3
1.1.	Specification of Requirements	3
1.2.	Terminology	3
2.	Protocol Overview	4
2.1.	Overview of EAP-Archie	4
2.2.	Retry Behavior	6
2.3.	Fragmentation	7
2.4.	Identity Verification	7
2.5.	Key Derivation	7
3.	Cryptographic Algorithms	9
3.1.	AES-CBC-MAC-128 and AES-CBC-MAC-96	9
3.2.	The Archie-PRF	9
4.	Detailed Description of EAP-Archie	10
4.1.	Archie Message Format Summary	10
4.2.	Archie-Request Message	11
4.3.	Archie-Response Message	15
4.4.	Archie-Confirm Message	22
4.5.	Archie-Finish Message	27
5.	IANA Considerations	30
6.	Security Considerations	30
6.1.	Intended Use	30
6.2.	Mechanism	30
6.3.	Security Claims	31
6.4.	Key Strength	35
6.5.	Key Hierarchy	35
6.6.	Vulnerabilities	35
7.	Acknowledgements	36
8.	References	36
9.	Author Addresses	37
10.	Full Copyright Statement	37
11.	Expiration Date.....	38

INTERNET-DRAFT

EAP-Archie

23 June 2003

[1.](#) Introduction

The Extensible Authentication Protocol (EAP), described in [\[4\]](#), provides a standard mechanism for support of additional authentication methods within PPP. EAP is also used within IEEE 802 networks through the IEEE 802.1X [\[8\]](#) framework.

EAP supports many authentication schemes, including smart cards, Kerberos, Public Key, One Time Passwords, TLS, and others. This memo defines a new authentication scheme, called the EAP-Archie. EAP-Archie performs mutual authentication, session establishment, and fresh session key derivation. Eap-Archie is based on a static pre-shared key.

[1.1.](#) Specification of requirements

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 [\[7\]](#).

[1.2](#) Terminology

This document frequently uses the following terms:

Backend Authentication Server

A backend authentication server is an entity that provides an authentication service to an authenticator. When used, this server typically executes EAP Methods for the Authenticator. [This terminology is also used in [\[IEEE.802-1X.2001\]](#).]

EAP Authenticator

The end of the EAP link initiating the EAP authentication methods. [Note: This terminology is also used in [\[IEEE.802-1X.2001\]](#), and has the same meaning in this

document].

EAP Peer

The end of the EAP Link that responds to the authenticator. [Note: In [IEEE.802-1X.2001], this end is known as the Supplicant.]

EAP Server

The entity that terminates the EAP authentication with the peer. In the case where there is no Backend Authentication Server, this term refers to the EAP Authenticator. Where the EAP Authenticator operates in pass-through, it refers to the Backend Authentication Server.

[2.](#) Protocol Overview

[2.1.](#) Overview of EAP-Archie

EAP-Archie is a native EAP authentication method. That is, EAP-Archie is an authentication protocol, and a stand-alone version of EAP-Archie outside of EAP is not defined. Its design follows the Bellare-Rogaway authentication paradigm [[11](#)], adapted somewhat to fit EAP.

EAP-Archie assumes a long-lived 512-bit secret shared between the EAP Peer and the EAP Server. This long-lived secret is called the Archie Key. The Archie Key has an internal structure. It consists of two 128-bit subkeys and a 256-bit subkey, respectively called the key-confirmation key (KCK), the key-encryption key (KEK), and the key-derivation key (KDK). The protocol uses the KCK to mutually authenticate the EAP Peer and the EAP Server. The protocol uses the KEK to distribute secret nonces used for session key derivation. The protocol uses the KDK to derive a session key between the EAP Peer and the EAP Server. EAP-Archie assumes that the Archie Key is known only to the EAP Peer and EAP Server, and the security properties of the protocol may be compromised if the pre-shared secret has wider distribution. The protocol also assumes the EAP Server and EAP Peer identify the correct Archie Key to use with the other by their respective NAIs [[9](#)].

A successful instance of the Archie protocol establishes a session, sometimes also called a security association.

EAP-Archie uses four messages consisting of two EAP request/response transactions. As with all EAP methods, the EAP Server initiates the exchange. Figure 1 depicts the EAP-Archie message flow:

```
EAP Peer                                EAP Server
=====                                =====
                                <--- EAP-Request/EAP-Type=Archie
                                (AuthID | SessionID)
EAP-Response/EAP-Type=Archie --->
(SessionID | PeerID | NonceP |
 Binding | MAC1)
                                <--- EAP-Request/EAP-Type=Archie
                                (SessionID | NonceA | Binding |
                                MAC2)
EAP-Response/EAP-Type=Archie --->
(SessionID | MAC3)
```

Figure 1. The EAP-Archie Message Flow

The first message is called an Archie-Start message. The Archie-Start message initiates the EAP-Archie exchange. The EAP Server sends the Archie-Start message to the EAP Peer in an EAP-Request message of type Archie. The Archie-Start message requests that the EAP Peer authenticate itself using EAP-Archie. The Archie-Start message is unauthenticated but does suggest the identity of the EAP Server, denoted by AuthID, and a challenge value, denoted SessionID. The AuthID is an NAI [9] identifying the EAP Server, and SessionID is a 256-bit random value. The AuthID allows the EAP Peer to identify the pre-shared secret for this context. SessionID serves several purposes. First, since it is unpredictable, it allows the EAP Server to detect replay attacks. Second, it provides the EAP server with a session identifier for the entire exchange.

The response to the Archie-Start is called the Archie-Response message. The EAP Peer sends the Archie-Response message to the EAP Server in an EAP-Response message of type Archie. The Archie-Response message authenticates the EAP Peer to the EAP Server, and requests the EAP Server to authenticate itself to the EAP Peer. This message conveys five values, called SessionID, PeerID, NonceP, Binding, and MAC1. SessionID is the value of the same name from the Archie-Request

to which this Archie-Response replies. PeerID is an NAI identifying the EAP Peer to the EAP Server. NonceP is a 256-bit random value selected by the EAP Peer, encrypted under the KEK using the AES Key Wrap [6]; the EAP Peer and EAP Server use the decrypted nonce value to derive a session key, as explained in [Section 2.5](#). MAC1 is a message authentication code computed over the Archie-Request's AuthID field and the prior Archie-Response message fields. The Message Authentication Code (MAC) algorithm is AES-CBC-MAC-96, defined in [Section 3.1](#), using the KCK as the key. SessionID identifies the session to which this Archie-Response belongs. PeerID allows the EAP Server to identify the pre-shared secret to use in this protocol instance. If the unencrypted nonce value is random, the encrypted value NonceP will be unpredictable as well, and thus NonceP can serve as a challenge from the EAP Peer to the EAP Server. Binding is the addressing information the EAP Peer believes it is using during this session. It indicates the identifier the EAP Peer uses for itself, as well as that of its intended communication partner, which is typically some sort of Network Access Server (NAS). Finally, a correct MAC1 value protects the Archie-Response undetected alteration, so authenticates the Archie-Response message and hence the EAP Peer to the EAP Server. Inclusion of the AuthID field from the Archie-Request in the MAC1 calculation binds the identities of the EAP Server and EAP Peer, thus helping to defeat invalid binding attacks and as a first step toward establishing a consistent view of the session.

The response to the Archie-Response message is called the Archie-Confirm message. The EAP Server sends the Archie-Confirm message to the EAP Peer in an EAP-Request message of type Archie. The Archie-Confirm message authenticates the EAP Server to the EAP Peer. The Archie-Confirm message conveys four values, SessionID, NonceA, Binding, and MAC2. SessionID is the value from the Archie-Request and Archie-Response for this instance of the Archie Protocol. NonceA is a 256-bit random value selected by the EAP Server and encrypted under the KEK using the AES Key Wrap; the EAP Peer and EAP Server use the decrypted nonce value to derive a session key, as explained in [Section 2.5](#). Binding is a copy of field of the same name from the Archie-Response message, and confirms the target addressing. MAC2 is a message authentication code computed over the Archie Request's AuthID field, the Archie-Response's NonceP field,

followed by the prior Archie-Confirm fields, using the KCK as the key and AES-CBC-MAC-96 as the MAC algorithm. Including the Archie-Response's AuthID field binds the EAP Server's name to this instance of the protocol, allowing the EAP Server and EAP Peer to form a consistent view of the protocol instance participants. Including the NonceP value in the MAC2 calculation also proves that the Archie-Confirm is live if the EAP Peer randomly selects its nonce value encrypted in the Archie-Response NonceP field. MAC2, if valid, authenticates the Archie-Confirm message and hence the EAP Server to the EAP Peer.

The final message responds to the Archie-Confirm, and is called the Archie-Finish message. The EAP Peer sends the Archie-Finish message to the EAP Server in an EAP-Response message of type Archie. This message consists of SessionID and MAC3. SessionID is the same value as in the Archie-Request, Archie-Response, and Archie-Confirm messages for this instance of the EAP-Archie protocol. MAC3 is the AES-CBC-MAC-96 digest of the fields of the Archie-Finish prior to the MAC3 field. This message is defined since every EAP-Request requires an EAP-Response. The Archie-Finish message includes MAC3 to prevent undetected forgery attacks, and allows the EAP Server to avoid some denial-of-service attacks.

[2.2](#). Retry Behavior

As with other EAP protocols, the EAP Server is responsible for retry behavior. This means that if the EAP Server does not receive a reply from the peer, it MUST resend the EAP-Request for which it has not yet received an EAP-Response. However, the EAP Peer MUST NOT resend EAP-Response messages without first being prompted by the EAP Server.

For example, if the initial Archie-Start message sent by the EAP

Server were to be lost, then the EAP Peer would not receive this message, so would not respond to it. As a result, the EAP Server would resend the Archie-Start message. Once the EAP Peer received the Archie-Start message, it would send an EAP-Response conveying the Archie-Response message. If the EAP-Response were to be lost, then the EAP Server would resend the initial Archie-Start, triggering the EAP Peer to resend the EAP-Response.

As a result, it is possible that an EAP Peer will receive duplicate EAP-Request messages, and may send duplicate EAP-Responses. Both the EAP Peer and the EAP Server should be engineered to handle this possibility.

[2.3.](#) Fragmentation

EAP-Archie does not require fragmentation.

[2.4.](#) Identity Verification

EAP-Archie always performs mutual authentication. EAP-Archie uses the Archie Key, a 512-bit long-lived pre-shared secret, identified to the EAP Peer by the EAP Server's NAI, and to the EAP Server by the EAP Peer's NAI. EAP-Archie uses the first 16 octets (first 128 bits) of the pre-shared secret serves as an authentication key, called the KCK.

The EAP Server considers the EAP Peer successfully authenticated if MAC1 from the Archie-Response message is valid. An EAP Server implementation of EAP-Archie MUST silently discard any EAP-Response messages of type Archie it receives with invalid MAC1 field.

Similarly, the EAP Peer considers the EAP Server authenticated if the MAC2 value from the Archie-Confirm message is valid. An EAP Peer implementation of EAP-Archie MUST silently discard any EAP-Request messages of type Archie it receives with invalid MAC2 field.

The EAP Server MAY require that the Binding field from the Archie-Response message be consistent with the verified identity and the NAS in use. For instance, the Binding might identify the MAC address of the EAP Peer. If it knows the proper value for this, the EAP Server MAY declare the message a forgery if this MAC address does not match the EAP Peer's reported NAI.

[2.5.](#) Key Derivation

Each instance of EAP-Archie constitutes a session, and derives an EAP Master Key (EMK) from the KDK portion of the long-lived pre-shared

secret. The derived keying material consists of 32 octets (256 bits). This section describes how the EMK is derived. It also describes how to derive a Transient EAP Key (TSK), used as a session oriented authorization token to protect a channel between the EAP Peer and NAS.

EAP-Archie relies on the Archie Key. EAP-Archie uses the second 16 octets (second 128 bits) of the pre-shared secret as a key encryption key, or KEK, and it uses the final 32 octets (512 bits) of the pre-shared key as a key derivation key, or KDK.

When using EAP-Archie, the EAP Peer selects a random value called PeerNonce, wraps it using the AES Key Wrap algorithm under the KEK to produce a field called NonceP, and sends NonceP to the EAP Server in the Archie-Response message. PeerNonce is 256-bits. The EAP Server uses KEK and the AES Key Wrap algorithm to recover PeerNonce.

Similarly, the EAP Server selects a random value called AuthNonce, wraps it using the AES Key Wrap algorithm under the KEK to produce a field called NonceA, and sends NonceA to the EAP Peer in the Archie-Confirm message. AuthNonce is 256 bits. The EAP Peer uses the KEK and the AES Key Wrap algorithm to recover AuthNonce.

Once both the EAP Server and Peer have both AuthNonce and PeerNonce, they derive the EMK using the KDK, AuthNonce, PeerNonce, and the text string "Archie session key", using a function called the Archie-PRF:

```
EMK = Archie-PRF(KDK,  
                  AuthNonce | PeerNonce | "Archie session key",  
                  32)
```

Here, the "|" symbol denotes string concatenation. [Section 3.2](#) defines the Archie-PRF. The parameter "32" indicates that the EMK consists of [32](#) octets.

Note that the AES Key Wrap has its own internal integrity check. Potential evidence of compromise of the KCK is the AES Key Wrap integrity check of either NonceP or NonceS fails but the MAC of the Archie message delivering the nonce is valid.

The derived EMK is uniquely identified by the 5-tuple <AuthID, PeerID, SessionID, AuthNonce, PeerNonce>, where AuthID is the NAI the EAP Server presents in the Archie-Request message, PeerID is the NAI the EAP Peer presents in the Archie-Response message, SessionID is the random challenge value the EAP Server presents in the Archie-Request message, and where AuthNonce and PeerNonce are as above.

The Archie EMK can be used to derive new cryptographic keys for use in other contexts, e.g., PPP encryption or to protect an 802 link. Following [10], the way to accomplish this is as follows:

$$\text{TSK} = \text{Archie-PRF}(\text{SK}, \text{AddrS} \mid \text{AddrP} \mid \text{"Archie transient EAP key"}, 128)$$

The derived key TSK is the first 128 octets of the output of the Archie_PRF. In the language of [10], the first 32 octets of the TEK is called the PMK.

3. Cryptographic Algorithms

3.1. AES-CBC-MAC-128 and AES-CBC-MAC-96

This section reviews the definition of the AES-CBC-MAC.

Let K denote an AES key, and let AES-Encrypt denote the AES encrypt primitive. The AES-CBC-MAC-128 of a string S is defined using the following steps:

- a. Let L denote the length of S in octets. Let $L' = L + p$, where p is the unique integer $0 \leq p < 16$ needed so that L' is a multiple of 16.
- b. Let $n = L'/16$, and partition S into substrings, such that $S = S[1] S[2] \dots S[n-1] S'[n]$, with $S[1], S[2], \dots, S[n-1]$ consisting of 16 octets each and $S'[n]$ consisting of 16 octets if p is 0 and $16-p$ octets otherwise. Let $S[n] = S'[n] \theta^p$, where θ^p denotes p octets of zero pad.
- c. Set IV to 16 zero octets.
- d. For $i = 1$ to n do
 $IV = \text{AES-Encrypt}(K, S[i] \text{ XOR } IV)$
- e. Return IV .

The EAP-Archie protocol uses a variant of AES-CBC-MAC-128 called AES-CBC-MAC-96. This variant differs from the AES-CBC-MAC-128 described above in only step e, which it replaces by:

- e'. Return the first 12 octets of IV .

Note that any padding added in step b is used to compute the MAC value only, and is never sent as part of an EAP-Archie message.

3.2. The Archie-PRF

This section defines the Archie-PRF function.

Let K denote a key, and let Length denote the number of octets desired. The Archie-PRF of a string S is defined by the following steps:

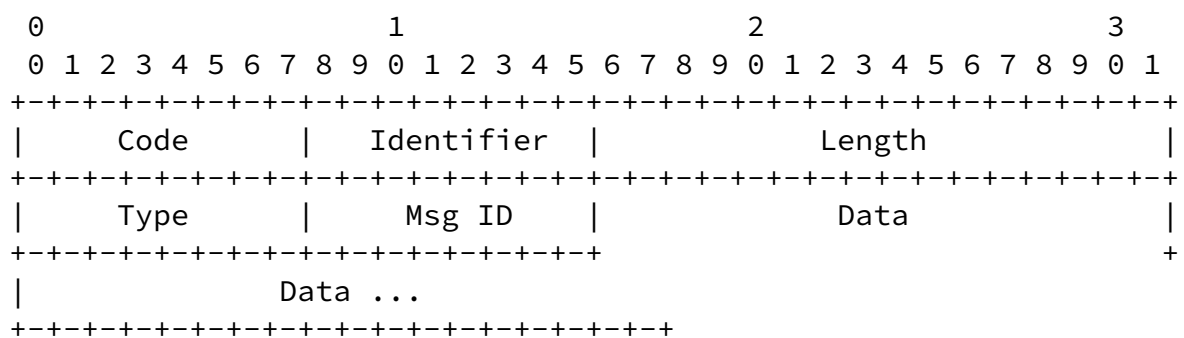
- a. Output is initialized to the null string.
- b. For $i = 1$ to $(\text{Length}+15)/16$ do
 Output = Output | AES-CBC-MAC-128(K, i | S | Length)
- c. Return first Length octets of Output

In step b, the loop index i and the Length are encoded as 32-bit big-Endian unsigned integers.

[4.](#) Detailed Description of EAP-Archie

[4.1.](#) Archie Message Format Summary

A summary of the Archie Request/Response packet format is shown below. The fields are transmitted from left to right.



Code

- 1 - Request
- 2 - Response

Identifier

The identifier field is one octet and aids in matching responses with requests.

Length

The Length field is two octets and indicates the length of the EAP message including the Code, Identifier, Length, Type, and Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception.

Type

TBS - Archie

MsgID

- 1 - Archie-Request
- 2 - Archie-Response
- 3 - Archie-Confirm
- 4 - Archie-Finish

Data

The format of the Data field is determined by the Code and MsgID fields.

This specification refers to the Code, Identifier, and Length fields as the EAP Header. The EAP Header is a standard construction starting every EAP message. It is implemented by EAP proper and not necessarily by EAP-Archie.

This specification uses the notation of the message name, followed by a parenthesized range list to indicate a substring of message included in some cryptographic computation. For example,

Archie-Request(Type...AuthID)

denotes the octet string formed by taking the Type through the AuthID fields of an Archie Request message. This includes the Type, MsgID, Reserved, NaiLength, SessionID, and AuthID fields of the Archie Request message. To refer to a single field in an EAP-Archie message we use the same notation but with a single argument instead. For

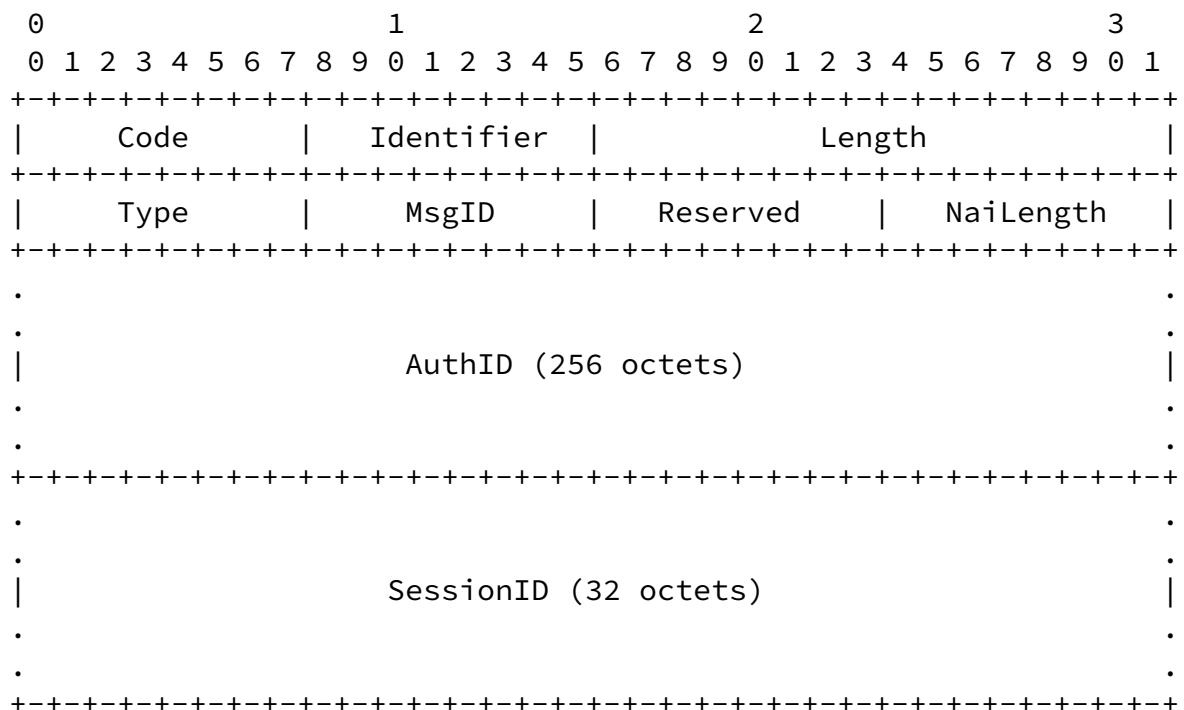
instance

Archie-Response(NonceP)

denotes the NonceP field of an Archie-Response message.

[4.2.](#) Archie-Request Message

A summary of the Archie-Request message format is shown below. The fields are transmitted from left to right.



Code

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field MUST be changed on each Request message.

Length

The Length field is two octets and indicates the length of the EAP message including the Code, Identifier, Length, Type, MsgID, Reserved, NailLength, AuthID, and SessionID fields. Its value is 296.

Type

TBS - Archie

MsgID

1 - Archie-Request

Reserved

Set to 0 on send, ignored on receive

NailLength

The number of octets used in the AuthID field. The value 0 means the AuthID is the full 256 octets in length.

AuthID

The AuthID field gives the NAI the EAP Server uses to identify itself to the EAP Peer. The first NailLength octets of this field are non-zero, while any remaining octets of the AuthID field MUST be zero on transmit and ignored on receive.

SessionID

The SessionID field is 32 octets. This value represents the

session id for this instance of the EAP-Archie protocol.

[4.2.1](#). Archie-Request Message Transmission

The EAP Server creates a new session to initiate a new instance of EAP-Archie. The Archie-Request is the first message of the new session. To construct and send an Archie-Request, the EAP Server performs the following steps:

- o Sets the Code field to 1.
- o Generates a new EAP Identifier value for the Identifier field.
- o Sets the Length field to 296, encoded in network byte order.
- o Sets the Type field to Archie.
- o Sets the MsgID field to 1 (Archie-Request).
- o Sets the Reserved field to 0.
- o Sets the NaiLength field value to the number of actual octets of the AuthID field, excluding any zero pad. If the AuthID is 256 octets, then the EAP Server sets the NaiLength to 0 instead.
- o Inserts the NAI the EAP Server uses to identify itself into the first NaiLength octets of the AuthID field and zeros any unused octets remaining in the field.

- o Generates a 32 octet value and inserts this into the SessionID field. The SessionID value MUST be random, as it the security of the protocol requires it be unpredictable. This also means that with very high probability the SessionID of every Archie Request will be different.

The EAP Server sends the Archie-Request message to the EAP Peer. The EAP Server SHOULD set a timer and a retry count to detect lost messages. The EAP Server MUST save whatever state it needs to verify whether an Archie-Response message received later is a valid response

to this Archie-Request. In particular, the saved state SHOULD include the SessionID value, to identify other messages that are a part of this session.

4.2.2. Archie-Request Message Reception

If the EAP Peer does not expect to receive an Archie-Request message, it MUST silently discard the Request.

An Archie-Request message is an EAP-Request of Type Archie, Length 296 octets, and MsgID 1. An EAP Peer MUST silently discard EAP-Requests of Type Archie and MsgID 1 of a different length. An Archie-Request responds to no other message, so always initiates a new session. Note that it is not a protocol error for multiple Archie-Request messages with different SessionID to be outstanding simultaneously.

The EAP Peer can identify a retried Archie-Request by matching the Archie-Request AuthID and SessionID fields against those received in a prior Archie-Request whose Archie-Response is still unacknowledged by an Archie-Confirm message. When interpreting the AuthID field, the EAP Peer MUST ignore any AuthID characters beyond the first Nailength octets in the field. If the EAP Peer receives an Archie-Request matching an outstanding Archie-Response, it MUST retransmit its Archie-Response if it wants to authenticate via Archie.

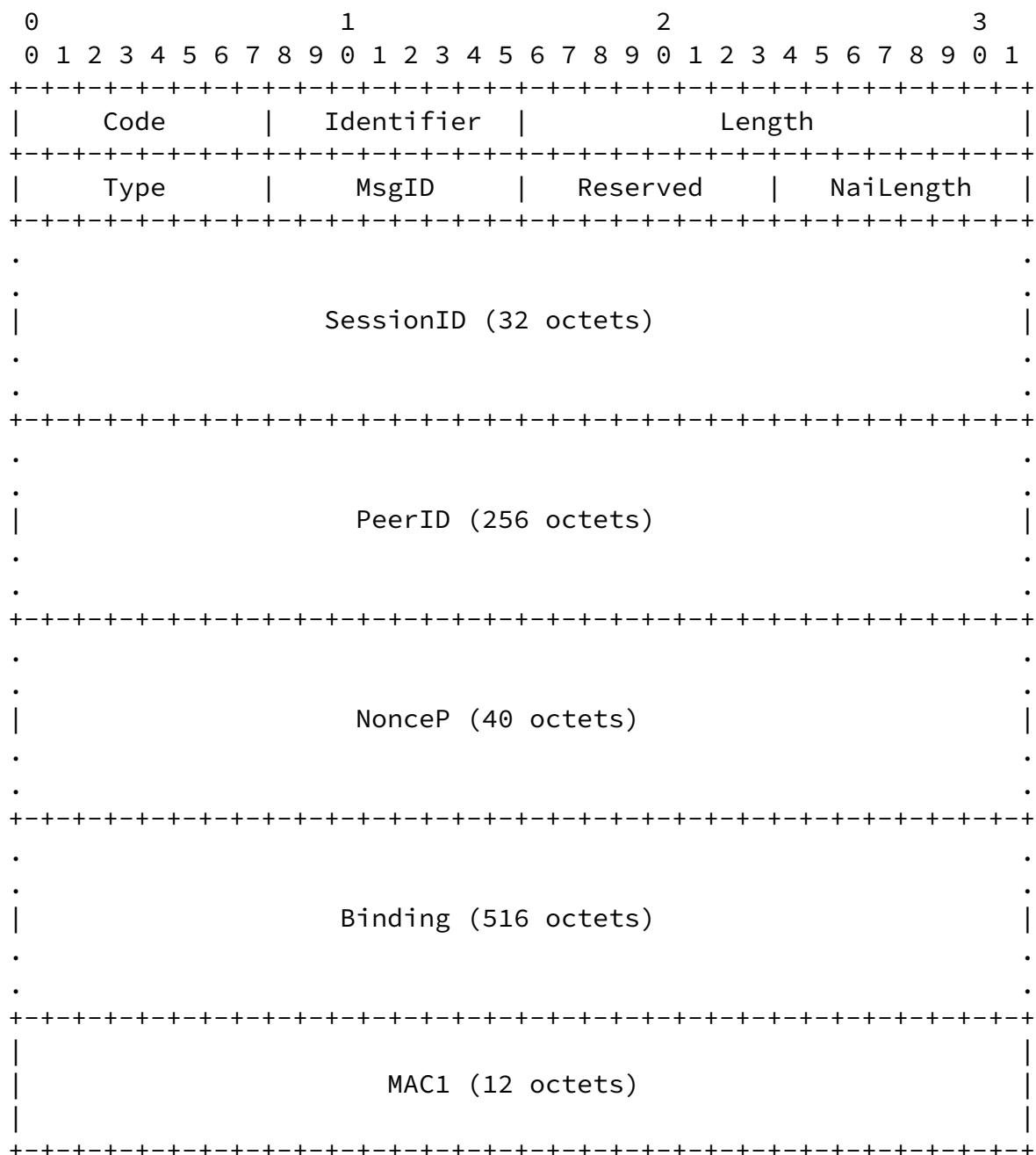
If the EAP Peer does not have an Archie-Response outstanding, i.e., if this represents a new protocol instance, then the Peer MUST first attempt to match the AuthID field value against the name of some EAP Server in its database of authorized EAP Servers. If the EAP Peer does not recognize the name, then it MUST NOT respond to the message with an Archie-Response message. In this case, the EAP Peer MAY wait for another Archie-Request message, or it may terminate the authentication. In either case, if a database entry for the EAP Server's NAI cannot be found, the EAP Peer silently discards the alleged Archie-Request message.

If the EAP Peer locates a database entry for the EAP Server's NAI, then it MAY choose to authenticate using EAP-Archie. In this case, it MUST create a new session and respond with an Archie-Response message,

and for this session the EAP Peer MUST select the Archie Key it associates with this NAI.

4.3. Archie-Response Message

A summary of the Archie-Response message format is shown below. The fields are transmitted from left to right.



Code

2

Identifier

The Identifier field is one octet and MUST match the Identifier field from the corresponding request.

Length

The Length field is two octets and indicates the length of the EAP message including the Code, Identifier, Length, Type, MsgID, Reserved, NaiLength, PeerID, NonceP, Binding, and MAC1 fields. Its value is 864.

Type

TBS - Archie

MsgID

2 - Archie Response

Reserved

Set to 0 on send, ignored on receive.

NaiLength

The number of octets used in the PeerID field. The value 0 means the PeerID is the full 256 octets in length.

SessionID

The SessionID field is 32 octets.

PeerID

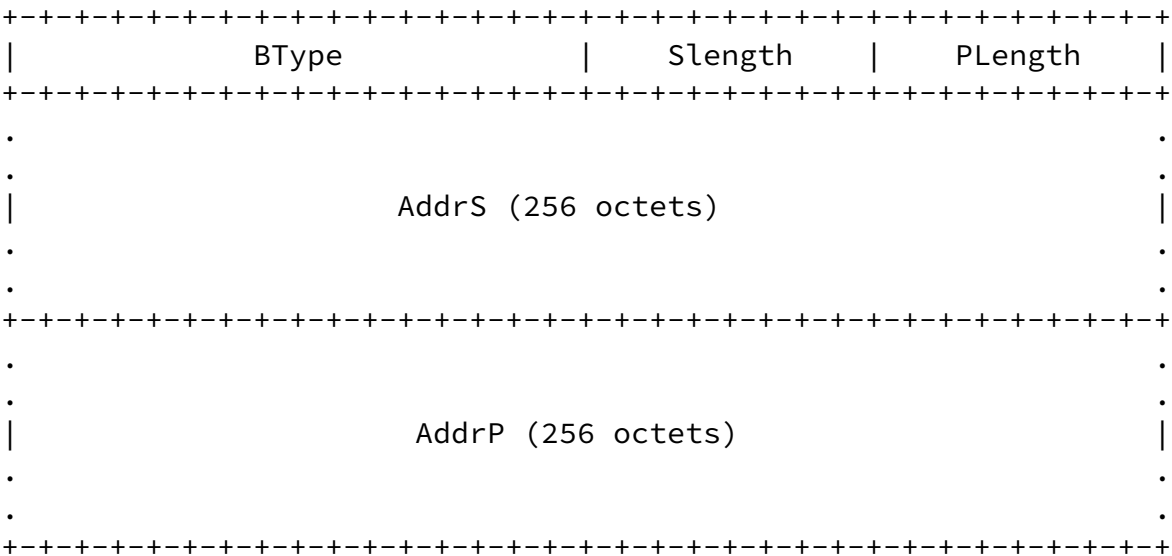
The PeerID field gives the NAI the EAP Peer uses to identify itself to the EAP Server. The first NaiLength octets of this field are non-zero, while any remaining octets of the PeerID field MUST be zero on transmit and ignored on receive.

NonceP

The NonceP field is 40 octets, conveying the value of a 32-octet nonce selected by the EAP Peer and encrypted using AES key wrap under the KEK portion of the Archie Key.

Binding

The Binding field is 516 octets. It has its own internal structure:



BType

Btype identifies the type of address in the binding structure. The value of this subfield is an Address Family Number as defined by the IANA Assigned Numbers database.

SLength

The number of octets of addressing information the AddrS Subfield contains. The value 0 is reserved for addressing Information which is 256 octets in length.

PLength

The number of octets of addressing information the AddrP Subfield contains. The value 0 is reserved for addressing Information which is 256 octets in length.

AddrS

AddrS identifies address of the party with whom the EAP Peer wants to communicate, typically a NAS device. This field assumes a different format depending of the value of the BType field. The AddrS subfield uses the standard encoding for the address of its BType. The sender SHALL zero pad any unused AddrS subfield octets.

AddrP

AddrP identifies address the EAP Peer wants to use when communicating with the targeted system, typically a NAS device. This field assumes a different format depending of the value of the BType field. The formatting rules for the AddrS field apply to this field verbatim. The sender SHALL zero pad any unused AddrS subfield octets.

The Binding field allows the EAP Peer to bind the derived TSK to a particular pair of addresses. This helps identify when the TSK is being used in an unauthorized context. In particular, the Bindings field are the EAP Peer's request for a TSK for use between the set of addresses the Bindings field specifies.

MAC1

The MAC1 field is 12 octets. The value of this field MUST be the AES-CBC-MAC-96 of the AuthID field from the Archie-Request message, followed by the Archie-Response message, less the EAP Header and MAC1 field:

$$\text{MAC1} = \text{AES-CBC-MAC-96}(\text{KCK}, \text{Archie-Request}(\text{Type} \dots \text{AuthID}) \mid \text{Archie-Response}(\text{Type} \dots \text{Binding}))$$

where "|" denotes concatenation.

[4.3.1](#). Archie-Response Message Transmission

The EAP Peer performs the following steps to construct and send an Archie-Response message:

- o Sets the Code field to 2.
- o Sets the EAP Identifier value to that in the EAP-Request message to which this message responds
- o Sets the Length field to 864, encoded in network byte order.

- o Sets the Type field to Archie.
- o Sets the MsgID field to 2 (Archie-Response).
- o Sets the Reserved field to 0.
- o Sets the NailLength field value to the number of actual octets of the PeerID field, excluding any zero pad. If the PeerID is 256 octets, then the EAP Server sets the NailLength to 0 instead.
- o Inserts the NAI the EAP Peer uses to identify itself into the first NailLength octets of the PeerID field, zeroing any unused octets in the field.
- o Copies the SessionID field of the Archie-Request message into the SessionID field of the Archie-Response.
- o Creates the NonceP field as follows:
 - Generates a 32 octet (256-bit) random or pseudo-random value called PeerNonce, to be used as the EAP Peer's nonce for this session.
 - Uses the AES key wrap algorithm [\[6\]](#) under the KEK portion of

the Archie Key for this protocol instance, to encrypt PeerNonce.

- Copies the encrypted PeerNonce into the NonceP field.
- o Creates the Bindings field as follows:
 - Encodes the Address Family identifier used as the Bindings field BType value. The EAP Peer encodes this in network byte order.
 - Sets the SLength subfield to the number of octets used in the AddrS subfield, with the convention that 0 designates that the AddrS is 256 octets long.
 - Sets the PLength subfield to the number of octets used in the AddrP subfield, with the convention that 0 designates that the AddrP is 256 octets long.
 - Encodes the address of BType its NAS will use with the derived TEK in the first SLength octets of the AddrS

subfield, zeroing the remaining unused octets of the subfield.

- Encodes the address of BType it will use for itself with the derived TEK in the first PLength octets of the AddrP subfield, zeroing the remaining unused octets of the subfield.
- o Creates the MAC1 field by using the KCK portion of the selected Archie key to compute the AES-CRC-MAC-96 of the concatenation of (1) the Type, MsgID, Reserved, NaiLength, and AuthID field from the Archie-Request to which the EAP Peer is responding and (2) the Archie-Response, less its EAP Header and MAC1 field. This includes the Type, MsgID, Reserved, NaiLength, SessionID, and AuthID fields from the Archie Request, and the Type, MsgID, Reserved, NaiLength, SessionID, PeerID, NonceP, and Bindings fields from the Archie-Response.

The EAP Peer sends the completed Archie-Response message to the EAP Server. The EAP Peer SHOULD set a timer bounding the time it will wait to receive a valid Archie-Confirm message replying to its Archie Response. The EAP Peer MUST save whatever state it needs to verify whether an Archie-Confirm message received later is a valid response to this Archie-Response message.

4.3.2. Archie-Response Message Reception

An Archie-Response message is an EAP-Response of Type Archie, Length [864](#) octets, MsgID 2, and with the EAP Identifier field of an outstanding EAP-Request and SessionID of the corresponding authentication instance. An EAP Server MUST silently discard EAP-Responses of Type Archie and MsgID 2 of a different length, as well as one whose Identifier field does not correspond to any EAP-Request, as well as an Archie-Response whose SessionID fails to match an active authentication instance. If the EAP Server receives an unexpected Archie-Response message, it MUST silently discard the Response.

Since an Archie-Response replies to an Archie-Request message, its reception makes sense only in one of two contexts: either the EAP Server has an Archie-Request outstanding, and this Archie-Response might be a response to that request, or else the Archie-Response might be a response to a retried Archie-Request that was retried before a Archie-Response was received. In the latter case it is possible that the EAP Server might have received a prior Archie-Response and so has an Archie-Confirm outstanding.

If the EAP Server has an Archie-Confirm outstanding, i.e., if it has already received an Archie-Response, it can identify whether this Archie-Response is a retry by verifying the message as being the same octet string as the original Archie-Response, if valid, received earlier. If the SessionID is the same but any other field after the EAP header differs from the earlier Archie-Response, the retried Archie-Response is invalid. The EAP Server MUST silently discard an alleged the retried Archie-Response in this case. Otherwise, the Archie-Response represents a valid reply, and the EAP Server SHOULD retransmit its Archie-Confirm message.

If the Archie-Response is not a retry, i.e., if the EAP Server has not yet received an Archie-Response replying to its Archie-Request, then it MUST first attempt to locate the PeerID field value in its database of authorized EAP Peers. If the EAP Server does not recognize the PeerID, then it MUST silently discard the message. If the EAP Server locates the PeerID in its database, then it tentatively selects the Archie Key corresponding to the PeerID for this session. The EAP Server uses the KCK portion of the Archie Key to verify the MAC1 field of the Response. It does this by computing the AES-CBC-MAC-96 over the Archie-Request, less EAP Header, for this SessionId, followed by the Archie-Response, less EAP Header and MAC1 field.

$$\text{MAC} = \text{AES-CBC-MAC-96}(\text{KCK}, \text{Archie-Request}(\text{Type} \dots \text{AuthID}) \mid \text{Archie-Response}(\text{Type} \dots \text{Bindings}))$$

Specifically, the EAP Server computes the MAC value over concatenation of (1) the Type, MsgID, Reserved, NaiLength, and AuthID fields from the Archie-Request, and (2) the Archie-Response, less its EAP Header and MAC1 field, which includes the Type, MsgID, Reserved, NaiLength, SessionID, PeerID, NonceP, and Bindings fields from the Archie-Response.

If the computed AES-CBC-MAC-96 value MAC is a different octet string than the MAC1 field value, then the EAP Server MUST silently discard the message as a forgery, and the EAP Server dissolves the binding of the selected Archie Key and the authentication instance. If the computed AES-CBC-MAC-96 value is identical to the MAC1 field, then the message is authentic, and the EAP Server makes the selected key a permanent part of this authentication instance.

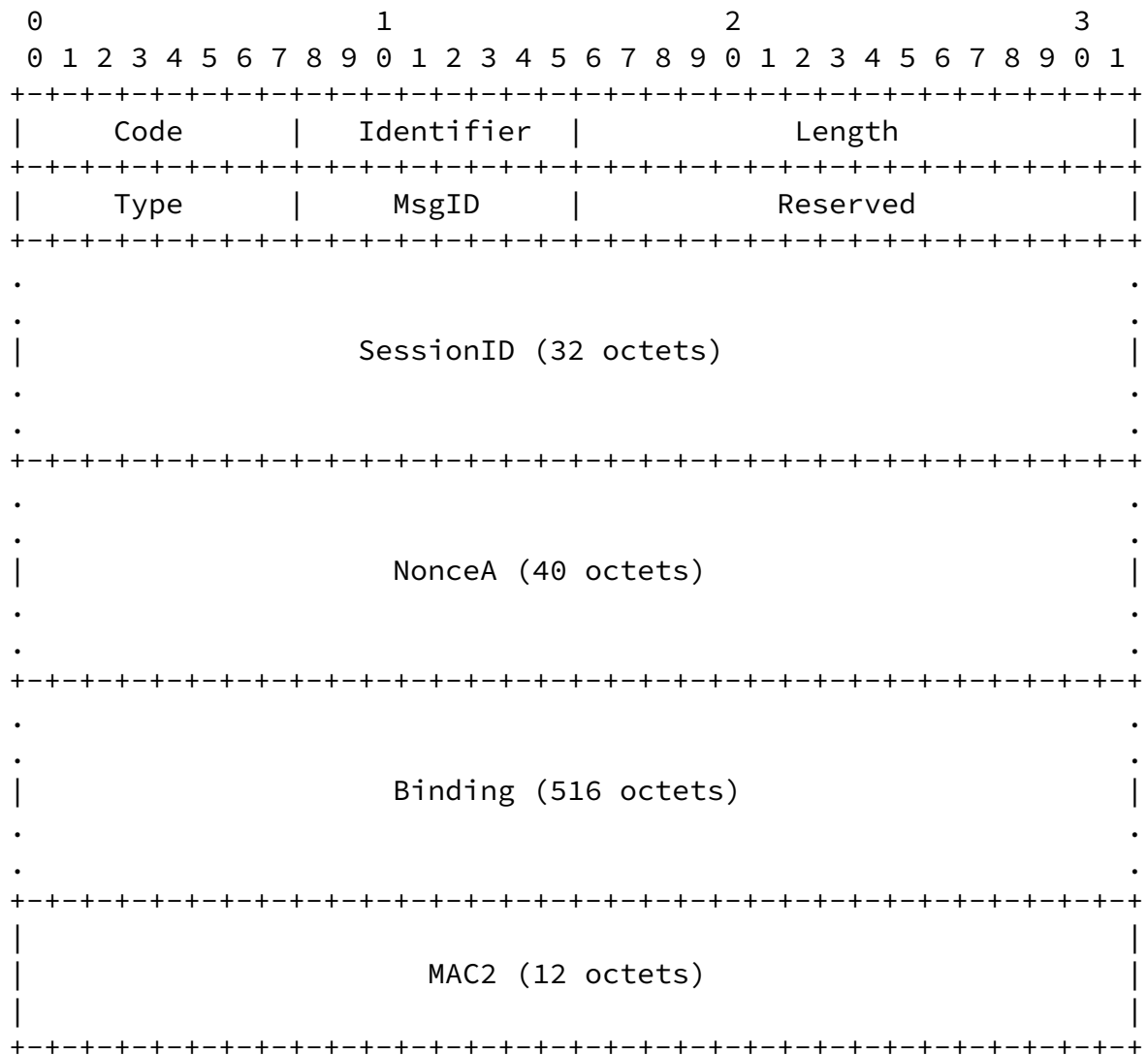
If it tracks the addressing information in the Bindings field, the EAP Server SHOULD verify the validity of the Bindings. In this case, the EAP Server SHOULD verify that the AddrP field is an expected address for the EAP Peer, and that the AddrS field is an expected address for the NAS. If either fails to correspond in the expected way, the EAP

Server MAY suspect a man-in-the-middle attack and deny the authentication. Alternately, it may report the discrepancy in the Archie-Confirm message as described in [Section 4.4](#).

If it verifies that the Archie-Response is valid, the EAP Server obtain the PeerNonce value by unwrapping the NonceP field by using the AES key wrap algorithm [6] under the KEK portion of the Archie Key for this session. If the key unwrap fails, the EAP Server should log an alert and silently discard the message; this is a strong indication that the Archie Key has been compromised and needs to be replaced. Otherwise, the EAP Server MUST reply to the Archie-Response by constructing and returning an Archie-Confirm message.

[4.4.](#) Archie-Confirm Message

A summary of the Archie-Confirm message format is shown below. The fields are transmitted from left to right.



Code

1

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field MUST be changed on each Request message.

Length

The Length field is two octets and indicates the length of the EAP message including the Code, Identifier, Length, Type, MsgID, Reserved, SessionID, NonceA, Binding, and MAC2 fields. Its value is 608.

INTERNET-DRAFT

EAP-Archie

23 June 2003

Type

TBS - Archie

MsgID

3 - Archie-Confirm

Reserved

Set to 0 on transmit, value ignored on receive

SessionID

The SessionID field is 32 octets.

NonceA

The NonceA field is 40 octets, which is a 32-octet random value encrypted by the AES key wrap under the KEK portion of the Archie Key.

Binding

The Binding field is 516 octets. It is encoded as in the Archie-Response message. The EAP Server SHOULD copy the value of this field from the Binding field of the Archie-Response message for this authentication instance. The EAP Server MAY alter the AddrS or AddrP values to indicate something wrong with the Archie-Response message Binding values (e.g., the AddrS MAC address does not belong to the NAS that forwarded the Archie-Response message), but the Archie-Confirm Binding BType value MUST be identical to that of the Archie-Response Binding BType.

MAC2

The MAC2 field is 12 octets. The value of this field MUST be the AES-CBC-MAC-96 of the (1) the Type, MsgID, Reserved, NaiLength, and AuthID field from the Archie-Request this protocol instance, followed by the NonceP field from the Archie-Response for this

session, followed by this Archie Confirm, less its EAP Header and MAC2 field. Specifically, MAC2 is computed over the Type, MsgID, Reserved, NailLength, and AuthID from the Archie-Request, the NonceP from the Archie-Request, and the Type, MsgID, Reserved, SessionID, NonceA, and Binding fields of this Archie-Confirm,

using the KCK portion of the long-lived pre-shared secret as the authentication key.

[4.4.1](#). Archie-Confirm Message Transmission

The EAP Server performs the following steps to construct and send an Archie-Confirm message:

- o Sets the Code field to 1.
- o Generates a new EAP Identifier value for the Identifier field.
- o Sets the Length field to 608, encoded in network byte order.
- o Sets the Type field to Archie.
- o Sets the MsgID field to 3 (Archie-Confirm).
- o Sets the Reserved field to 0.
- o Copies the SessionID field for this authentication instance into the SessionID field of the Archie-Confirm.
- o Creates the NonceA field, using the following steps:
 - Generates a 32 octet (256-bit) random or pseudo-random value to be used as the EAP Server's nonce for this session.
 - Uses the KEK portion of the selected Archie long-lived key with the AES Key Wrap algorithm to encrypt the EAP Server's nonce value. The resulting string is 40 octets.
 - Copies the encrypted EAP Peer nonce into the NonceP field.

- o Copies the Bindings field from the Archie Response into the Bindings field of the Archie-Confirm. The EAP Server MAY alter the value of either the Bindings AddrS or AddrP subfields if either fails to correspond with the EAP Server's expectation.
- o Creates the MAC2 field by using the KCK portion of the selected long-lived Archie key to compute the AES-CRC-MAC-96 of the concatenation of the (1) Type, MsgID, Reserved, NaiLength, and AuthID fields of the Archie-Request, (2) the NonceP field of the Archie-Response, and (3) this Archie-Confirm, less the EAP header and MAC2 fields, i.e., Type, MsgID, Reserved, SessionID, NonceA, and Bindings fields.

The EAP Server sends the completed Archie-Confirm message to the EAP Peer. The EAP Server SHOULD set a timer and a retry count to detect lost messages.

[4.4.2.](#) Archie-Confirm Message Reception

An Archie-Confirm is an EAP-Request of Type Archie, Length 608 octets, MsgID 3, and SessionID of an active authentication instance. An EAP Peer MUST silently discard EAP-Requests of Type Archie and MsgID 3 of a different length, as well as an Archie-Confirm whose SessionID fails to match the authentication instance. If the EAP Peer receives an unexpected Archie-Confirm message, it MUST silently discard the Confirm.

Since an Archie-Confirm replies to an Archie-Response message, its reception makes sense only in one of two contexts: either the EAP Peer has an Archie-Response outstanding, and the Archie-Confirm might be a reply to that Archie-Response, or else the Archie-Confirm might be a reply to the EAP Peer's own reply to a retried Archie-Request. In the latter case it is possible that the EAP Peer might have received a prior Archie-Confirm and so has complete EAP-Archie authentication. Hence it is necessary for the EAP Peer to continue to be able to respond to retried Archie-Confirm messages received for some period after receiving an initial Archie-Confirm.

If the EAP Peer has already completed EAP-Archie authentication by sending an Archie-Finish, it can identify a retried Archie-Confirm by

matching the Archie Confirm against a prior one received with a valid MAC2 field. If the Archie-Confirm, less EAP header, consists of the same octet string as the earlier valid Archie-Confirm, then the EAP Peer SHOULD retransmit its Archie-Finish message. The EAP Peer MUST silently discard an alleged retried Archie-Confirm if any of the octets differ from the earlier Archie Confirm.

If the EAP Peer locates the SessionID among its active Archie authentication instances, then it uses the KCK portion of the Archie Key to verify the MAC2 field of the Archie-Confirm. It does this by computing the AES-CBC-MAC-96 as follows:

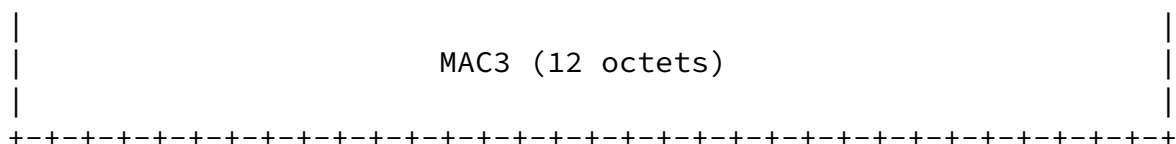
$$\text{MAC} = \text{AES-CBC-MAC-96}(\text{KCK}, \text{Archie-Request}(\text{Type} \dots \text{AuthID}) \mid \text{Archie-Response}(\text{NonceP}) \mid \text{Archie-Confirm}(\text{Type} \dots \text{Bindings}))$$

That is, it recomputes the MAC value over the Archie-Request Type, MsgID, Reserved, NailLength, and AuthID fields from the Archie-Request for this session, followed by the NonceP value it sent in the

Archie-Response for this session, followed by the Type, MsgID, Reserved, SessionID, NonceA, and Bindings values in the Archie-Confirm. If the computed MAC value is a different octet string than the received MAC2 field value, then the EAP Peer MUST silently discard the message as a forgery. If the computed MAC value is identical to the MAC2 field, then the message is authentic.

If it verifies that the Archie Confirm is valid, the EAP Peer uses the KEK portion of the Archie Key for this session with the AES Key Wrap algorithm to obtain the EAP Server's nonce value by unwrapping the NonceA field. If recovery of the AuthNonce from the NonceA field fails due to an AES Key Wrap integrity check failure, the EAP Peer MUST log an error and break off the authentication; this is evidence that the Archie Key has been compromised.

If the recovery of the AuthNonce value succeeds, the EAP Peer MUST implement one final validity check. If the EAP Server has altered the addressing information in the Bindings field from that sent in the Archie Response, then the EAP Peer SHOULD take this as a report of a man-in-the-middle attack and end its association with the NAS.



Code

2

Identifier

The Identifier field is one octet and MUST match the Identifier field from the corresponding request.

Length

The Length field is two octets and indicates the length of the EAP message including the Code, Identifier, Length, Type, MsgID, Reserved, SessionID, and MAC3 fields. Its value is 52.

Type

TBS - Archie

MsgID

4 - Archie-Finish

Reserved

0 on send and ignored on receive

SessionID

The SessionID field is 32 octets.

MAC3

The MAC3 field is 12 octets. The value of this field MUST be the AES-CBC-MAC-96 of the Type, MsgID, Reserved, and SessionID fields of

this Archie-Finish message, using the KCK portion of the Archie Key as the authentication key.

[4.5.1.](#) Archie-Finish Message Transmission

The EAP Peer performs the following steps to construct and send an Archie-Finish message:

- o Sets the Code field to 2.
- o Sets the EAP Identifier value to that in the EAP-Request message to which this message responds
- o Sets the Length field to 52, encoded in network byte order.
- o Sets the Type field to Archie.
- o Sets the MsgID field to 4 (Archie-Finish).
- o Sets the Reserved field to 0.
- o Copies the SessionID field for this Archie authentication instance into the SessionID field of the Archie-Confirm.
- o Creates the MAC3 field by using the KCK portion of the Archie Key for this authentication instance to compute the AES-CRC-MAC-96 of the Type, MsgID, Reserved, and SessionID fields of this Archie-Finish message.

The EAP Peer sends the completed Archie-Finish message to the EAP Server. The EAP Peer SHOULD set a timer bounding the time it will continue wait to receive a retried Archie-Confirm message.

[4.5.2.](#) Archie-Finish Message Reception

An Archie-Finish message is an EAP-Response of Type Archie, Length 52 octets, MsgID 4, with the EAP Identifier field of an outstanding EAP-Request and SessionID of the corresponding authentication instance. An EAP Server MUST silently discard EAP-Responses of Type Archie and MsgID 4 of a different length, as well as one whose

Identifier field does not correspond to any EAP-Request, as well as an Archie-Finish whose SessionID fails to match the authentication instance. If the EAP Server receives an unexpected Archie-Finish message, it MUST silently discard the Finish.

If the EAP Server is waiting for an Archie-Finish, then it uses the KCK portion of the Archie Key to verify the MAC3 field of the Archie-Finish. It does this by computing the AES-CBC-MAC-96 over Archie-Finish message less EAP Header, i.e.,

$$\text{MAC} = \text{AES-CBC-MAC-96}(\text{KCK}, \text{Archie-Finish}(\text{Type} \dots \text{SessionID}))$$

If the computed MAC value is a different octet string than the received MAC3 field value, then the EAP Server MUST silently discard the message as a forgery. If the computed MAC value is identical to the MAC3 field, then the message is authentic. and the EAP Server makes the selected key a permanent part of this authentication instance.

If it verifies that the Archie-Finish is valid, the EAP Server uses the KDK portion of the Archie Key to derive the EMK and TEK as described in 2.5.

[5.](#) IANA Considerations

This document introduces one new IANA considerations.

It requires IANA to allocate a new EAP Type for EAP-Archie.

[6.](#) Security Considerations

[6.1.](#) Intended use

EAP-Archie is designed to be useful over an insecure network. The protocol implements mechanisms to protect an EAP-Archie exchange from both active and passive attack.

[6.2.](#) Mechanism

EAP-Archie relies on a 512-bit pre-shared secret. The protocol does not restrict how the secret is constructed nor address how it comes into being. However, the protocol does not increase the entropy of the shared secret in any way; if the shared secret does not provide sufficient entropy, then any derived keys can have less entropy as

INTERNET-DRAFT

EAP-Archie

23 June 2003

well, and it may be possible to compromise the Archie exchange as well.

[6.3](#). Security Claims

EAP-Archie provides:

- a. Conservative use of cryptography,
- b. Integrity protection,
- c. Replay protection,
- f. Man-in-the-middle resistance,
- d. Mutual authentication,
- e. Session formation,
- f. Consistent view,
- g. Peer liveness,
- h. Fresh key derivation,
- i. Key strength,
- j. Dictionary attack resistance, and
- g. Limited Denial-of-Service protection.

[6.3.1](#). Conservative use of Cryptography

EAP-Archie uses AES as its sole cryptographic primitive. AES is believed to be a conservative choice of a block cipher.

EAP-Archie uses AES in two ways: in the CBC-MAC construction and in the AES key wrap. The CBC-MAC construction results in a pseudorandom function if the underlying block cipher is a pseudorandom permutation [\[12\]](#). Since it is standard to assume that AES effectively models a pseudorandom permutation, this makes AES-CBC-MAC safe to use as a MAC and for key derivation. It is also a standard assumption that the AES key wrap protects its wrapped contents from disclosure or alteration if the KEK has sufficient entropy and is not exposed.

EAP-Archie uses message formats each of whose lengths are fixed and known a priori, meaning CBC-MAC is a safe construction if the underlying block cipher has not been compromised [\[12\]](#).

EAP-Archie's design provides strong key separation by dividing the Archie Key into three non-overlapping subkeys: the KCK, the KEK, and the KDK. EAP-Archie uses the KCK only for message authentication, the KEK for key encryption, and the KDK for message derivation. This

property means that cryptanalytic progress against the KCK does not contribute any useful knowledge to make cryptanalytic progress against the KDK and KEK, and hence against the EMK. Similar, this separation guarantees that cryptographic progress against the EMK, KDK, or KEK

contributes nothing to the analysis of the KCK.

All nonces used by EAP-Archie are 256-bits and are specified as random.

[6.3.2.](#) Integrity Protection

EAP-Archie provides integrity protection of its messages by computing the AES-CBC-MAC-96 of each of the messages exchanged in a protocol instance, using the KCK of the pre-shared Archie Key as the MAC key. If the KCK is well chosen, then its 128-bit size precludes exhaustive search. The 96 bit MAC size protects the protocol from an active birthday attacks attempting to forge a packet in significantly fewer than 2^{48} messages. AES used in CBC-MAC mode provides message integrity against all other forgery attacks provided AES is not broken [12]. Truncation of the AES-CBC-MAC to 96 bits helps protect the KCK from attack by examination of the protocol messages.

[6.3.3.](#) Replay Protection

EAP-Archie provides replay protection—that is, message freshness—through the SessionID and NonceP fields.

If the EAP Server selects the SessionID randomly, then it is unpredictable and an adversary has one chance in 2^{256} to guess its value. Since SessionID is included in each Archie Message, it is protected by the AES-CBC-MAC-96. The EAP Server will ignore Archie messages with SessionIDs that are not part of a current instance, and the AES-CBC-MAC-96 will allow the EAP Server to detect attempts to alter the SessionID in Archie messages. This precludes an attacker from causing the EAP Peer to construct a useful dictionary of pre-computed Archie Response, Confirm, or Finish messages.

EAP-Archie's use of NonceP for replay protection is similar but subtler. NonceP appears explicitly only in the Archie-Response

message. However, the protocol requires the EAP Server to include the Archie-Response message in calculating the MAC2 value of the Archie-Confirm message. If the EAP Peer randomly selects the nonce encrypted as NonceP, an adversary has at most one chance in 2^{256} in selecting the right NonceP value, so the EAP Peer can detect replayed Archie-Confirm messages by invalid MAC2 values.

[6.3.4.](#) Man-in-the-Middle Attack Resistance

The EAP-Archie design defends against man-in-the-middle attack against an instance of the Archie protocol itself if the KCK portion of the

pre-shared key has sufficient entropy to resist cryptanalytic attack. This can be verified using the argument showing how EAP-Archie accomplishes mutual authentication: since the MAC1 value in the Archie Response message is over both SessionID and NonceP, it is not possible that the Archie-Response message could be interwoven from a different Archie protocol instance, so the EAP Server knows there can be no man-in-the-middle. Similarly, the MAC2 value in the Confirm message is computed over both the Archie-Response NonceP value and the body of the Confirm, so the EAP Peer may conclude that the Confirm has not been interleaved from a different Archie protocol instance.

EAP-Archie defends against the man-in-the-middle attack against the TSK by including the addresses of the NAS and of the EAP Peer to the TSK.

[6.3.5.](#) Mutual Authentication

The Archie-Request, Archie-Response, and Archie-Confirm messages and the mapping of NAIs to a common pre-shared Archie Key accomplish mutual authentication. The Archie-Request message includes the EAP Server's NAI, and then includes this again in the MAC2 computation of its Archie-Confirm message. By assumption, the EAP Peer maps the NAI from the Archie-Request to the correct Archie Key. If it receives a valid Archie-Confirm message as part of the same protocol instance, then this authenticates the EAP Server to the EAP Peer, because the Archie-Confirm is fresh and has a correct MAC2, so could only be constructed by another entity that possesses the Archie Key, and because the Archie-Confirm format is different from any other protocol

message and is only constructed by the EAP Server. By similar reasoning, the Archie-Response authenticates the EAP Peer to the EAP Server.

[6.3.6. Session Formation](#)

EAP-Archie accomplishes session formation by binding each particular protocol instance to a distinct <EAP Server, EAP Peer> pair. The 5-tuple <Server NAI, Peer NAI, SessionID, NonceS, NonceP> distinguishes this session from any other sessions if the EAP Server selects SessionID or its AuthNonce value randomly, or if the EAP Peer selects its PeerNonce randomly. The mutual authentication provided by the KCK portion of the pre-shared Archie key confirms this information as belonging to a single session entity. Similarly, the use of the KCK binds each EAP-Archie message to this session. Other session attributes are the derived EMK, the derived TEK, and the binding. The 5-tuple <Server NAI, Peer NAI, SessionID, NonceS, NonceP> can be used to name both the session and its EMK. Use of the EMK demonstrates

authorization to participate in this session.

[6.3.6. Consistent View](#)

When an instance of EAP-Archie succeeds, both the EAP Server and EAP Peer share the same view of the participants in the protocol instance and their respective roles, and of the protocol state. Both parties view the EAP Server as the initiator of the protocol instance, and that it is named by the AuthID from the Archie-Request message. Both parties view the EAP Peer as the responder of the protocol instance, and that it is named by the PeerID from the Archie-Response message.

[6.3.7. Peer Liveness](#)

EAP-Archie's replay protection mechanism also assures both the EAP Server and EAP Peer that the other is live, if the period either is willing to wait for a message is bounded.

[6.3.6. Fresh Key Derivation](#)

EAP-Archie derives both fresh session keys EMK and TEK, as described

in 2.5. The EMK is fresh if either the nonce decrypted from NonceP or NonceS is fresh.

[6.3.7. Key Strength](#)

The key strength of the derived EMK depends on the key derivation algorithm and on the strength of the KEK and KDK portions of the Archie Key.

Since AES-CBC-MAC is a pseudorandom function if AES is a pseudorandom permutation, and since it is standard to assume that AES models a pseudorandom permutation, we can assume the derived keys will be strong if the KDK and KEK contain sufficient entropy.

Since the KDK portion of the Archie Key is 256-bits, the KDK can contribute at most 256-bits of entropy to the EMK.

While the AuthNonce and PeerNonce values used to derive the EMK are not public information, and while they combine to contribute 512 bits to the key derivation, they do not contribute 512 bits of entropy to the EMK. instead they are encrypted under the KEK portion of the Archie Key. Since the KEK is 128-bits, the nonces provide at most 128-bits of entropy of the EMK.

[6.3.8. Dictionary Attack Resistance](#)

EAP-Archie provides resistance to on- and off-line dictionary attack if the KCK and KEK portions of the Archie Key have sufficient entropy to resist dictionary attack.

[6.3.10. Limited Denial-of-Service Resistance](#)

EAP-Archie provides limited denial-of-service resistance for the EAP server through the Archie-Finish message. If the EAP Server does not receive the Archie-Finish in a timely fashion, it can safely deallocate the corresponding session, thus freeing resources for another session.

[6.4. Key Hierarchy](#)

EAP-Archie uses a three-level key hierarchy.

Level 1 is the Archie Key. This is a 512-bit key partitioned into three subkeys: the 128-bit key confirmation key (KCK) used for mutual authentication, the 128-bit key encryption key (KEK) used for nonce hiding, and the 256-bit key derivation key (KDK), which is used to derive the next level of the hierarchy.

The second level of the hierarchy is the session key, called the EMK. The session key is derived from the KCK and the nonces using the Archie-PRF. Because it includes nonces in its derivation, the master key is fresh key if either nonce is random. The session key is known only to the EAP Peer and EAP Server.

The final level of the hierarchy is the derived TEK, which are bound to particular <EAP Peer, NAS> pairs. The TEK is derived from the session key and addressing information of the EAP Peer and NAS. Since the EMK is fresh, the derived TEK is also fresh.

[6.5. Vulnerabilities](#)

The Archie Request message is vulnerable to spoofing. The Archie Request message could be MACed, but this would not eliminate the problem, as the first message of any session establishment protocol such as EAP-Archie is subject to replay.

EAP-Archie cannot provide mutual authentication if the 128 most significant bits of the Archie Key are known to any party other than the EAP Server and the EAP Peer.

EAP-Archie cannot guarantee that the derived session keys EMK and TSK are secure unless the last 256 bits of the Archie Key are known only

to the EAP Server and the EAP Peer.

If the Archie Key is derived from a low-entropy quantity such as a password, then EAP-Archie is subject to both on- and off-line dictionary attacks, and the EMK and TSK are both subject to off-line dictionary attack.

To defend against on-line dictionary attack, implementations SHOULD keep track of the number of EAP-Archie messages received with invalid message authentication codes throughout the Archie Key lifetime, and SHOULD force the Archie Key to be changed if the number exceeds some threshold.

The protocol is not secure unless SessionID is random in the sense of being unpredictable to any computationally bound adversary. If SessionID is not unpredictable, then an attacker can masquerade as the EAP Server to the EAP Peer for the purpose of generating the Archie-Response. The attacker can then replay the Archie-Response to the EAP Server at a later time, when the Server actually would issue SessionID.

7. Acknowledgements

EAP-Archie evolved from an earlier unpublished Archie protocol defined by Bob Moskowitz, Doug Whiting, Greg Chesson, Jesse Walker, Russ Housley, and Thomas Hardjono. Bernard Aboba, Doug Whiting, and Glen Zorn also contributed useful comments that led to the protocol's evolution.

8. References

- [1] Simpson, W., Editor, "The Point-to-Point Protocol (PPP)", STD 51, [RFC 1661](#), July 1994.
- [2] Sklower, K., Lloyd, B., McGregor, G., Carr, D. and T. Coradetti, "The PPP Multilink Protocol (MP)", [RFC 1990](#), August 1996.
- [3] Simpson, W., Editor, "PPP LCP Extensions", [RFC 1570](#), January 1994.
- [4] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", [RFC 2284](#), March 1998.

- [5] National Bureau of Standards, "Secure Hash Standard", FIPS PUB 180-1 (April 1995).
- [6] Housley, R., "Advance Encryption Standard (AES) Key Wrap Algorithm", [RFC 3394](#), September 2002.
- [7] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [8] IEEE STD 802.1X, Standards for Local and Metropolitan Area Networks: Port Based Access Control, June 14, 2001
- [9] Aboba, B., and M. Beadles, "The Network Access Identifier", [RFC 2486](#), January 1999.
- [10] Aboba, B., and D. Simon, "EAP Keying Framework", [draft-aboba-pppext-key-problem-05.txt](#), December 2002
- [11] Bellare, M, and P. Rogaway, "Entity Authentication and Key Distribution," CRYPTO 93, LNCS 773, pp232-249, Springer-Verlag, Berlin, 1994.
- [12] Bellare, M., J Kilian, and P. Rogaway, "The Security of Cipher Block Chaining," Journal of Computer and System Sciences, Vol. 61, No. 3, Dec 2000, pp 362-399.

9. Author Addresses

Jesse R. Walker
Intel Corporation
2111 N.E. 25th Avenue
Hillsboro, OR 97214
USA
jesse.walker@intel.com

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA
housley@vigilsec.com

10. Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved. This

document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English. The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

11. Expiration Date

This memo is filed as <[draft-jwalker-eap-archie-01.txt](#)>, and expires December 23, 2003.

