

DIME
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2016

J. Korhonen
Broadcom Ltd.
H. Tschofenig
ARM Ltd.
February 29, 2016

Diameter AVP Level Security: Keyed Message Digests, Digital Signatures,
and Encryption
[draft-korhonen-dime-e2e-security-03.txt](#)

Abstract

This document defines an extension for end to end authentication, integrity and confidentiality protection of Diameter Attribute Value Pairs. The solution focuses on protecting Diameter Attribute Value Pairs and leaves the key distribution solution to a separate specification. The integrity protection can be introduced in a backward compatible manner to existing application. The confidentiality protection requires an explicit support from an application, thus is applicable only for newly defined applications.

Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC 2119](#) [[RFC2119](#)].

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 September 1, 2016.

Internet-Draft

Diameter E2E Security

February 2016

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	3
2.	Solution description	4
2.1.	Integrity protection of AVPs	4
2.2.	Confidentiality protection of AVPs	7
2.3.	Definition of the 'End Point'	8
3.	AVP Encoding	8
3.1.	Signed-Data AVP	8
3.2.	JWS-Header AVP	8
3.3.	Header-Parameters AVP	8
3.4.	JWS-AVP-Payload AVP	9
3.5.	JWS-Signature AVP	9
3.6.	Encrypted-Data AVP	9
3.7.	JWE-Header AVP	9
3.8.	JWE-Enc-Key AVP	10
3.9.	JWE-Init-Vec AVP	10
3.10.	JWE-AVP-Ciphertext AVP	10
4.	Result-Code AVP Values	10
4.1.	Transient Failures	10
4.2.	Permanent Failures	11
5.	IANA Considerations	11
6.	Security Considerations	11
7.	Acknowledgements	12
8.	References	12
8.1.	Normative References	12
8.2.	Informational References	12
	Authors' Addresses	13

Internet-Draft

Diameter E2E Security

February 2016

1. Introduction

The Diameter base protocol [[RFC6733](#)] leverages IPsec and TLS for mutual authentication between neighboring Diameter nodes and for channel security offering data origin authentication, integrity and confidentiality protection. The Diameter base protocol, however, also defines Diameter agents, namely Relay Agents, Proxy Agents, Redirect Agents, and Translation Agents.

Relay Agents are Diameter agents that accept requests and route messages to other Diameter nodes based on information found in the messages. Since Relays do not perform any application level processing, they provide relaying services for all Diameter applications.

Similarly to Relays, Proxy Agents route Diameter messages using the Diameter routing table. However, they differ since they modify messages to implement policy enforcement.

Redirect Agents do not relay messages, and only return an answer with the information necessary for Diameter agents to communicate directly, they do not modify messages. Redirect Agents do not have negative impacts on end-to-end security and are therefore not considered in this document.

A Translation Agent is a device that provides translation between two protocols. To offer end-to-end security across different protocol requires the ability to convey and process the AVPs defined in this document by both end points. Since such support is very likely not available this document does not cover this functionality.

The Diameter extension defined in this document specifies how AVP authentication, integrity and confidentiality protection can be offered using either symmetric or asymmetric cryptography. As a solution mechanism is derived from Javascript Object Signing and Encryption (JOSE). JOSE offers a simple encoding with small set of

features ideal for the purpose of Diameter. This document further defines a binary efficient coding of JOSE objects.

This document focuses on protecting Diameter AVP and leaves the key distribution solution to a separate specification, which most likely is going to be a specific key exchange application. To offer the functionality two grouped AVPs are defined: Signed-Data and Encrypted-Data. The respective JOSE objects are transported within these two AVPs.

[2.](#) Solution description

[2.1.](#) Integrity protection of AVPs

JWS represents digitally signed or HMACed content using JSON data structures. The representation in [[I-D.ietf-jose-json-web-signature](#)] consists of three parts: the JWS Header, the JWS Payload, and the JWS Signature. The three parts are represented as the concatenation of the encoded strings in that order, with the three strings being separated by period ('.') characters. For the JWS Payload one would define a new JSON object that contains an array of AVP code number and a hash of AVP pairs. The JWS Signature then covers the all APVs to be signed or HMACed. Both JWS Payload and signature MUST use the same hash algorithm of the cryptographic algorithm indicated in the JWS Header.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of of the JWS Header, JWS Payload or JWS Signature are not BASE64 encoded but are processed in their plaintext or binary representation formats. For example, the JWS Header is encoded in its plaintext format into the Header-Parameters AVP:

```
{  "typ":"JWT",
  "alg":"HS256",
  "kid":"abc123"
}
```

The JWS Payload and the JWS Signature hashes and AVP Code values are

encoded in their binary format as octets, not in textual or BASE64 encoded formats. Sections [3.4](#) and [3.5](#) describe the encodings of the needed AVPs.

To package a set of AVPs for signing, each AVP octet representation to be protected are first individually hashed and encoded into the "JSON object" with its four octets AVP code number. The entire AVP MUST be input to the hash calculation, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/flags, and optional vendor IDs, and padding. The AVP MUST be input to the hash calculation in network byte order.

The JWS Signature is calculated over the entire JWS Payloads and then the all three JWS parts are placed in the Signed-Data AVP. There can be multiple Signed-Data AVPs in a Diameter message. The AVP code in the JWS Payload is to indicate which AVP this hash possibly refers to. If there are multiple instances of the same AVP in the Diameter message, there is no other way than make the verification against all of those. It is possible that the message sender only hashed one AVP

of the same type and, therefore, the receiver MUST verify the hash against all occurrences of the AVP of the same code number. Such flexibility is added there to allow reordering of the AVPs and addition or deletion of new AVPs by intermediating agents.

If a receiver detects errors with the processing of the Signed-Data AVP it MAY return one of the errors defined in [Section 4](#). If a receiver does not find any AVP the Signed-Data AVP has a signature for, it MAY also return one of the errors defined in [Section 4](#).

When AVPs are to be both encrypted and signed, the Encrypted-Data AVP MUST be created first. This means that signing is "outside" encryption.

Here is an example: Imagine the following AVPs from the QoS-Resources AVP in the QoS-Install Request (defined in [RFC 5866](#) [[RFC5866](#)] message shall be signed. The resulting example message has the following structure:

```
<QoS-Install-Request> ::= < Diameter Header: 327, REQ, PXY >
                           < Session-Id >
                           { Auth-Application-Id }
```

```

    { Origin-Host }
    { Origin-Realm }
    { Destination-Realm }
    { Auth-Request-Type }
    [ Signed-Data ]
    * [ QoS-Resources ]
    ...

```

Example Diameter Message with Signed-Data AVP

The Signed-Data AVP in this example may contain a JWS Header that indicates the use of the HMAC SHA-256 algorithm with the key id 'abc123'. The protected AVPs are Session-Id, Origin-Host and Origin-Realm. The calculated HMAC SHA-256 values are for example purposes only (i.e., are not real):

JWS Header encoded as such in JWS-Header AVP:

```

{"typ":"JWT",
 "alg":"HS256",
 "kid":"abc123"
}

```

```

0x000000xxx // JWS-Header code 'xxx'
0x000000034 // Flags=0, Length=52
'{"typ":"JWT","alg":"HS256","kid":"abc123"}' // 41
0x00,0x00,0x00 // 3 octets padding

```

JWS Payload encoded into three JWS-AVP-Payload AVPs:

```

0x000000zzz // JWS-AVP-Payload code 'zzz' <--+

```

```

0x0000001c // Flags=0, Length=28 |
0x00000107 // 263, Session-Id, 4 octets s
0x9d0e0495 // hash of Session-Id, 128 bits i
0xba8c0312 g
0xb6274c52 n
0x7d51a048 a
t
0x00000zzz // JWS-AVP-Payload code 'zzz' u
0x0000001c // Flags=0, Length=28 r
0x00000108 // 264, Origin-Host, 4 octets e
0x39ca88ff // hash of Origin-Host, 128 bits |
0xaa5a6ff9 c
0x029ed95b o
0xa534e028 v
e
0x00000zzz // JWS-AVP-Payload code 'zzz' r
0x0000001c // Flags=0, Length=28 a
0x00000128 // 296, Origin-Realm, 4 octets g
0x202730ac // hash of Origin-Realm, 128 bits e
0xa6e3a180 |
0x2f44a633 |
0xf250f6fe <--+

```

JWS Signature encoded into the JWS-Signature AVP:

```

0x00000yyy // JWS-Signature code 'yyy'
0x00000018 // Flags=0, Length=24
0xaabbccdd,0xddeeff00,0x11223344,0x55667788

```

Example JWS Header, Payload and Signature

2.2. Confidentiality protection of AVPs

The Encrypted-Data AVP (AVP Code TBD) is of type OctetString and contains the JSON Web Encryption (JWE)

[[I-D.ietf-jose-json-web-encryption](#)] data structure and consists of four parts: the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext. The four parts are represented as the concatenation of the encoded strings in that

order, with the three strings being separated by period ('.') characters. JWE does not add a content integrity check if not provided by the underlying encryption algorithm.

Although the solution relies on the JSON, the encoding into Diameter AVPs differ from the text based encoding of the JSON objects. Specifically, none of the the JWE Header, the JWE Encrypted Key, the JWE Initialization Vector and the JWE Ciphertext are not BASE64 encoded but are processed in their plaintext or binary representation formats. The concept follows what was already described in [Section 2.1](#).

A single AVP or an entire list of AVPs MUST be input to the encryption process, from the first byte of the AVP code to the last byte of the AVP data, including all other fields, length, reserved/flags, and optional vendor IDs, and padding. The AVP MUST be input to the encryption process in network byte order, and the encryptor is free to order AVPs whatever way it chooses. When AVPs are to be both encrypted and authenticated, the Encrypted-Data AVP MUST be created first.

Note that the usage of the Encrypted-Data AVP requires explicit support by the Diameter application since a receiving Diameter node must first decrypt the content of the Encrypted-Data AVP in order to evaluate the AVPs carried in the message. In case that a Diameter node is unable to understand the Encrypted-Data AVP and ignores the AVP then two possible outcomes are possible: First, if the encrypted AVPs are optional then their content is not considered by the receiving Diameter server without any indication to the sender that they have not been processed. Worse, in the second case when the encrypted AVPs are mandatory to be processed then the receiving Diameter node will return an error that may not inform the sender about the failure to decrypt the Encrypted-Data AVP. Consequently, the usage of the Encrypted-Data AVP may require changes to the ABNF definition of a Diameter application.

If a receiver detects that the contents of the Encrypted-Data AVP is invalid, it SHOULD return the new Result-Code AVP value defined in [Section 4](#).

Although this specification claims to introduce the end-to-end security into Diameter, the definition who actually is the 'end point' is not obvious. The 'end point' does not need to be the original Diameter request or answer originator but the Diameter node that inserts the Signed-Data or the Encrypted-Data AVPs into the Diameter message. The node can be the request or answer originator or a proxy agent. Use of proxy agents doing the 'end-to-end' security on behalf of other nodes mimics the deployments where site-to-site VPNs are used.

[3.](#) AVP Encoding

[3.1.](#) Signed-Data AVP

The Signed-Data AVP (AVP Code TBD1) is of type Grouped and utilizes the JSON Web signature (JWS) mechanism defined in [\[I-D.ietf-jose-json-web-signature\]](#). The JWS payload is then encoded into the Signed-Data AVP:

```
Signed-Data ::= < AVP Header: TBD1 >
               { JWS-Header }
               * { JWS-AVP-Payload }
               { JWS-Signature }
               * [ AVP ]
```

[3.2.](#) JWS-Header AVP

The JWS-Header AVP (AVP Code TBD2) is of type UTF8String and contains the JSON Web Signature Header. The contents of the AVP follow the rules for the header found in [\[I-D.ietf-jose-json-web-signature\]](#), which implies the required IANA registries are also defined by JSON documents.

```
JWS-Header ::= < AVP Header: TBD2 >
               { Header-Parameters }
               * [ AVP ]
```

The "alg" is the only REQUIRED Header Parameter for the signature purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

[3.3.](#) Header-Parameters AVP

The Header-Parameters AVP (AVP Code TBD3) is of type UTF8String and contains the JSON Header Parameter Name and its value as described in [\[I-D.ietf-jose-json-web-signature\]](#). The encoding (textual) also

follows [[I-D.ietf-jose-json-web-signature](#)]. Differing from the JSON specifications the parameter names and values are not BASE64 encoded but in their original UTF-8 representation format.

[3.4.](#) JWS-AVP-Payload AVP

The JWS-AVP-Payload AVP (AVP Code TBD4) is of type OctetString and contains both an AVP Code and a hash of the entire AVP identified by the AVP Code. The first four octets contain the AVP Code in a network byte order followed by the hash octets. The length of the hash octets depends on the used hash algorithm.

[3.5.](#) JWS-Signature AVP

The JWS-Signature AVP (AVP Code TBD5) is of type OctetString and contains the signature calculated over the array of complete JWS-AVP-Payload AVPs (including AVP header fields etc) in the order they appear in the Signed-Data AVP. The length of the signature octets depends on the used signature algorithm.

[3.6.](#) Encrypted-Data AVP

The Encrypted-Data AVP (AVP Code TBD6) is of type Grouped and utilizes the JSON Web Encryption (JWE) mechanism defined in [[I-D.ietf-jose-json-web-encryption](#)]. The JWE payload is then encoded into the Encrypted-Data AVP:

```
Encrypted-Data ::= < AVP Header: TBD1 >
                { JWE-Header }
                { JWE-Enc-Key }
                [ JWE-Init-Vec ]
                { JWE-AVP-Ciphertext }
                * [ AVP ]
```

[3.7.](#) JWE-Header AVP

The JWE-Header AVP (AVP Code TBD7) is of type UTF8String and contains the JSON Web Encryption Header. The contents of the AVP follow the rules for the header found in [[I-D.ietf-jose-json-web-encryption](#)], which implies the required IANA registries are also defined by JSON documents.

```
JWE-Header ::= < AVP Header: TBD7 >
              { Header-Parameters }
              * [ AVP ]
```

Internet-Draft

Diameter E2E Security

February 2016

The "alg" and "enc" are the REQUIRED Header Parameter for the encryption purposes. The "typ" and "kid" Header Parameters are also RECOMMENDED.

[3.8.](#) JWE-Enc-Key AVP

The JWE-Enc-Key AVP (AVP Code TBD8) is of type OctetString and contains the JWE Encrypted Key in its binary format.

[3.9.](#) JWE-Init-Vec AVP

The JWE-Init-Vec AVP (AVP Code TBD9) is of type OctetString and contains the JWE Initialization Vector in its binary format.

[3.10.](#) JWE-AVP-Ciphertext AVP

The JWE-AVP-Ciphertext AVP (AVP Code TBD10) is of type OctetString and contains the encrypted AVPs. The encrypted AVPs are first concatenated into one large plaintext octet blob and then encrypted as a whole. The length of the ciphertext depends on the used algorithm and encrypted AVPs. The plaintext to be encrypted is never BASE64 encoded but MAY be compressed if a "zip" parameter was included in the JWE Header.

[4.](#) Result-Code AVP Values

This section defines new Diameter result code values for usage with Diameter applications.

[4.1.](#) Transient Failures

Errors that fall within the transient failures category are used to inform a peer that the request could not be satisfied at the time it was received, but MAY be able to satisfy the request in the future.

DIAMETER_KEY_UNKNOWN (TBD11)

This error code is returned when a Signed-Data or an Encrypted-Data AVP is received that was generated using a key that cannot be

found in the key store. This error may, for example, be caused if one of the endpoints of an end-to-end security association lost a previously agreed upon key, perhaps as a result of a reboot. To recover a new end-to-end key establishment procedure may need to be invoked.

DIAMETER_HEADER_NAME_ERROR (TBD12)

This error code is returned when a Header Parameter Name is not understood in the JWS-Header AVP or in the JWE-Header AVP.

[4.2.](#) Permanent Failures

Errors that fall within the permanent failures category are used to inform the peer that the request failed, and should not be attempted again.

DIAMETER_DECRYPTION_ERROR (TBD13)

This error code is returned when an Encrypted-Data AVP is received and the decryption fails for an unknown reason.

DIAMETER_SIGNATURE_ERROR (TBD14)

This error code is returned when a Signed-Data AVP is received and the verification fails for an unknown reason.

[5.](#) IANA Considerations

IANA is requested to allocate AVP codes for the following AVPs:

AVP Name	AVP Code	Section Defined	Data Type
Signed-Data	TBD1	3.1	Grouped
JWS-Header	TBD2	3.x	Grouped
JWS-AVP-Payload	TBD3	3.x	OctetString
JSW-Signature	TBD4	3.x	OctetString
Header-Parameters	TBD5	3.x	UTF8String

Requirement Levels", [BCP 14](#), [RFC 2119](#),
DOI 10.17487/RFC2119, March 1997,
<<http://www.rfc-editor.org/info/rfc2119>>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn,
Ed., "Diameter Base Protocol", [RFC 6733](#),
DOI 10.17487/RFC6733, October 2012,
<<http://www.rfc-editor.org/info/rfc6733>>.

[8.2](#). Informational References

[I-D.ietf-aaa-diameter-e2e-sec]
Calhoun, P., "Diameter End-2-End Security Extension",
2001.

[RFC5866] Sun, D., Ed., McCann, P., Tschofenig, H., Tsou, T., Doria,
A., and G. Zorn, Ed., "Diameter Quality-of-Service
Application", [RFC 5866](#), DOI 10.17487/RFC5866, May 2010,
<<http://www.rfc-editor.org/info/rfc5866>>.

Authors' Addresses

Jouni Korhonen
Broadcom Ltd.
3151 Zanker Road
CA 95134
US

Email: jouni.nospam@gmail.com

Hannes Tschofenig
ARM Ltd.

Email: Hannes.Tschofenig@gmx.de
URI: <http://www.tschofenig.priv.at>

