AAA Working Group
Internet-Draft
Category: Standards Track
<draft-ietf-aaa-diameter-cms-sec-04.txt>

Pat R. Calhoun
Black Storm Networks
Stephen Farrell
Baltimore Technologies
William Bulley
Merit Network, Inc.
March 2002

Diameter CMS Security Application

Status of this Memo

Abstract

The Diameter base protocol leverages either IPsec or TLS for integrity and confidentiality between two Diameter peers, and allows the peers to communicate through relay and proxy agents. Relay agents perform message routing, and other than routing AVPs, do not modify Diameter messages. Proxy agents, on the other hand, implement policy enforcement, and actively modify Diameter messages.

This Diameter application describes how a security association is established by two peers through agents, and how authentication, integrity, confidentiality and data origin authentication are achieved using a mixture of symmetric and asymmetric transforms, by

encapsulating Cryptographic Message Syntax (CMS) data within AVPs. CMS is also used to carry X.509 certificates.

Table of Contents

1.0 Introduction
- 1.1 Requirements language
- 1.2 Establishing Security Relationship through relay agents
- 1.3 Establishing Security Relationship through proxy agents
- 1.4 Using Redirect agents in lieu of DSA
- 1.5 When to use DSAs
- 1.6 Advertising application support
- 1.7 CMS Processing of Grouped AVPs.

2.0 AVP Format

3.0 Key Management
- 3.1 Usage Scenario
- 3.2 Certificate Requirements
- 3.3 Algorithms
- 3.4 Reuse of CMS Content Encryption Keys

4.0 Command-Codes Values
- 4.1 Diameter-Security-Association-Request
- 4.2 Diameter-Security-Association-Answer
- 4.3 Proxy-Diameter-Security-Association-Request
- 4.4 Proxy-Diameter-Security-Association-Answer

5.0 Diameter Security Association Message Flow

6.0 Diameter Security AVPs
- 6.1 CMS-Signed-Data AVP
- 6.2 CMS-Encrypted-Data AVP
- 6.3 Example Encodings
- 6.4 Local-CA-Info AVP
  - 6.4.1 CA-Name AVP
  - 6.4.2 Key-Hash AVP
- 6.5 OCSP-Nonce AVP
- 6.6 AAA-Node-Cert AVP
- 6.7 OCSP-Responses AVP
- 6.8 CA-Chain AVP
- 6.9 OCSP-Request-Flags AVP
- 6.10 DSAR-Target-Realm AVP
- 6.11 DSA-TTL AVP

7.0 Result-Code AVP Values
- 7.1 Transient Failures
- 7.2 Permanent Failures

8.0 AVP Occurrence Tables

9.0 IANA Considerations
- 9.1 Command Codes
- 9.2 AVP Codes
- 9.3 Result-Code AVP Values

9.4 Application Identifier
9.5 OCSP-Request-Flags AVP Values
10.0 Security Considerations
11.0 References
12.0 Acknowledgements
13.0 Authors' Addresses
14.0 Full Copyright Statement
15.0 Expiration Date

1.0 Introduction

The Diameter base protocol [BASE] leverages either IPsec or TLS for integrity and confidentiality between two Diameter peers. However, the Diameter protocol also allows peers to communicate through relay and proxy agents, and in such environments security information is lost at each agent.

Relay agents perform message routing, and other than routing AVPs, do not modify Diameter messages. Proxy agents, on the other hand, implement policy enforcement, and actively modify Diameter messages. See [BASE] for a more comprehensive definition of the role of relay and proxy agents.

There are two main techniques used in this specification. Digital signatures (along with digital certificates) provide authentication, integrity and data origin authentication. Encryption provides confidentiality (using asymmetric techniques to encrypt a content encryption key, which is then used for bulk encryption). Both techniques can be used simultaneously to provide all the specified security services.

This Diameter application makes use of Cryptographic Message Syntax (CMS), which is the method used to secure MIME (S/MIME) messages. This application was designed to allow Diameter implementations to use existing S/MIME toolkits in order to comply with this specification.

This specification contains two different set of messages. The Diameter Security Association (DSA) messages are used to establish a security assocation, while the Proxy Diameter Security (PDS) messages are used to request that a security association be established by a third party.

The following details the necessary support for both types of messages based on the type of Diameter node:

- Diameter servers: MUST support DSA messages; MAY support PDS messages
- Proxy agents: MUST support DSA messages; MUST support PDS messages
- Diameter clients: SHOULD support DSA messages; MUST support PDS messages
- Relay agents: MAY support DSA messages; MAY support PDS messages
- Redirector agents: MAY support DSA messages; MAY support PDS messages

1.1 Requirements language

In this document, the key words "MAY", "MUST", "MUST NOT", "optional", "recommended", "SHOULD", and "SHOULD NOT", are to be

interpreted as described in [MUSTSHOULD].

1.2 Establishing a Security Relationship through relay agents

The AAA Working Group has defined a set of requirements in [AAAREQS] to allow for Diameter peers to communicate securely through Relay agents. This requirement calls for AVP integrity and confidentiality between two peers communicating through agents. The term agent is used in this specification for either a relay or a proxy agent. Figure 1 provides an example of two Diameter peers establishing a Diameter Security Association (DSA) through Relay agents. The participants of a DSA are the peers where the DSA setup messages terminate. In this example, the participants of the DSA would the NAS (access device), and the Home Server.

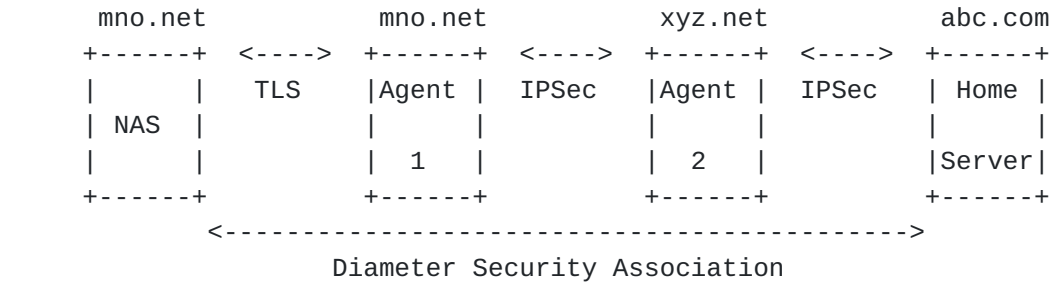

Figure 1: Diameter Security Association

When one or more agents are used between two communicating Diameter peers, the use of hop-by-hop security mechanisms (e.g. TLS, IPSec) is unsuitable, since Diameter messages are processed at the application layer at each agent. Therefore, an alternative mechanism is required to protect portions of the message at the application layer.

Allowing for a security association to be established through Diameter relays allows the participants of the DSA to detect whether protected AVPs have been modified en-route, and hides sensitive data from intermediate agents. Furthermore, the Mobile IP and NASREQ Working Groups have stated in [CDMAREQ, MIPREQ] that data origin authentication of Diameter data, such as Accounting related AVPs, is necessary.

Figure 2 provides an example of a message sent by an access device (NAS), through Diameter relay agents, to its intended destination, the home server. In this example, Proxy 2 modifies the contents of the foo AVP, perhaps due to mis-configuration, or maliciously. This specification would allow the participants of the DSA to detect such a problem, as long as the AVP being modified was protected.

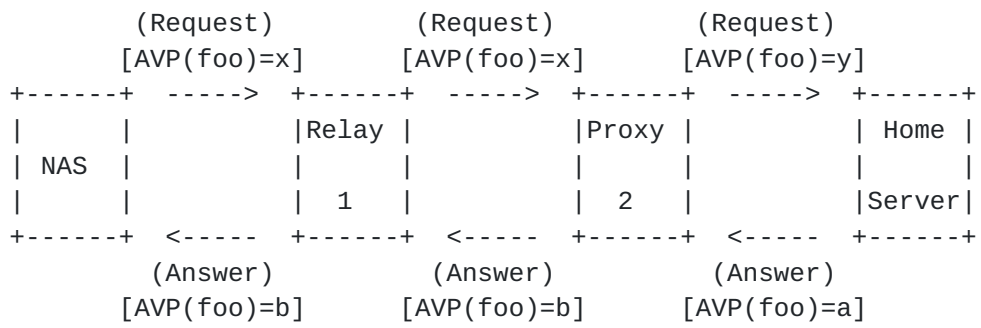

Figure 2: Diameter agent modifying AVP

This document defines the Diameter commands that are used to establish a DSA through Diameter agents, and specifies how encapsulating CMS objects [CMS] in Diameter AVPs can provide authentication, integrity, confidentiality and data origin authentication. CMS objects MAY also be used to transport X.509 certificates and revocation lists.

Establishing a DSA through relay agents requires that the initiator issues a Diameter Security Association Request (DSAR) message. In the example provided in figure 1, NAS would issue the DSAR with the Destination-Realm AVP set to abc.com. The Home Server would process the request, and respond by issuing a Diameter Security Association Answer (DSAA) message. If the DSAA message contains a Result-Code indicating success, the DSA is established between the NAS and the home server.

Once the DSA is established, participants with private keys MAY apply digital signatures to protect one or more AVPs within a message. In the example provided in Figure 2, the Foo AVP would be protected by the digital signature, and any modification of the AVP by the relay agents, would be detected since the signature validation algorithm would fail.

1.3 Establishing Security Relationship through proxy agents

As previously discussed, proxy agents typically modify Diameter messages to implement policy enforcement. An example of a proxy server would be an aggregating server, which typically sits one Diameter hop away from the access device, and enforces policy in order to protect the access device from receiving AVPs that could cause harm (e.g. excessive number of filters, unsupported tunneling protocol). Although in theory such checks could be performed on the access device, these devices are typically embedded systems, and not easily configurable. The proxy agent's behavior, on the other hand, is typically under control of the network operator.

Diameter messages between two participants of a DSA would fail verification if a proxy agent were to modify any protected AVPs.

Therefore proxy agents that modify AVPs MAY prevent the establishment of DSAs based on local configuration.

In this section, we discuss the capabilities provided by this Diameter application which allow proxy agents to secure AVPs on behalf of an access device. The following scenarios are envisioned:

- The access device does not have the cryptographic ability to handle CMS functions locally, and therefore requests such services from the local agent, such an an aggregating relay or proxy agent. The NAS may have been configured to always issue a PDSR to its local agent for CMS services. In such cases, the agent MUST select the values for the DSA-TTL.
- The access device has the cryptographic ability to perform CMS functions, but a proxy agent is in the route towards the home server, and it returned a failure to the DSAR messages stating that it was not willing to allow the DSA to traverse through it. Such agents MAY attempt to re-use the values from the initial DSAR sent by the access device. In such cases,the PDSR initiator SHOULD include the Destination-Host AVP to ensure that the PDSR is received by the same proxy agent.
- The access device may have the cryptographic ability to perform CMS functions locally, but does not request a DSAR to request a DSA. The local agent, however, has been configured to establish DSAs with certain realms automatically, hiding the existence of the DSAs from the access device.

In the above scenarios, the first two occur at the explicit request of the access device, while the last one occurs without any messaging from the access device. In the latter case, the proxy agent acts as an access device of sorts and the rules in section 1.2 should be used instead.

When a local agent receives a DSAR, it has the following options:

- The local agent rejects the DSAR by sending a DSAA message whose Result-Code AVP is set to DIAMETER_NO_CMS_THROUGH_PROXY. The DSAA SHOULD include the CMS-Signed-Data AVP, signed by the proxy agent, and include its certificate to allow the access device to validate the originator of the DSAA. The access device can then determine whether it is willing to provide service, based on its local policy.
- The local agent rejects the DSAR by sending a DSAA message whose Result-Code AVP is set to DIAMETER_CAN_ACT_AS_CMS_PROXY, informing the access device that the agent is willing to

establish the DSA on its behalf. The DSAA MUST include the CMS-Signed-Data AVP, signed by the proxy agent, and include its certificate to allow the access device to validate the originator of the DSAA. If the access device is willing to use the agent's services, it issues a Proxy-Diameter-Security-Association-Request (PDSR) which MUST contain the target realm. The local agent MAY use any of the parameters provided by the access device in the previous DSAR attempt when establishing the DSA. Once the DSA is established, the agent MUST issue a Proxy-Diameter-Security-Association-Answer (PDSA). The PDSA MUST contain the TTL setting agreed by the proxy agent for its DSA. This information will allow the access device to re-issue a PDSR prior to the proxy's DSA expiry if it needs the DSA to remain active.

Note that an access device MAY be configured to always issue a PDSR to its aggregating proxy, reducing the number of round trips. Similarly, an aggregating proxy MAY be configured to initiate an DSAR regardless of whether a PDSR was sent by the access device.

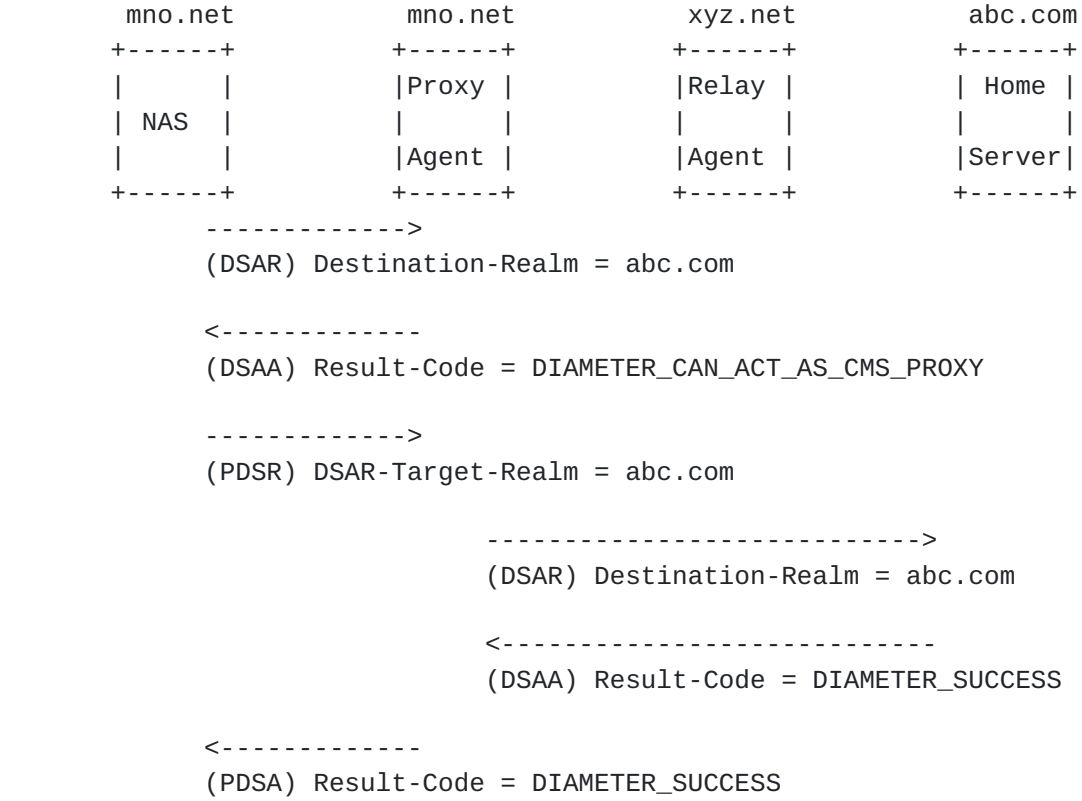

Figure 3: Establishing Security through Proxy Agent

An optimized approach also allows the PDSR to be sent by the access

device without the DSAR-Target-Realm AVP. This message is used to inform the proxy that it MUST establish a DSA for all realms it will be communicating with on behalf of the access device. DSAs are typically established once the first request for a given realm has been recelived by the proxy agent, but it MAY establish certain DSAs with known realms in advance.

If a DSA for a given realm cannot be established, the proxy agent MUST reject the access device's request, and set the Result-Code AVP to DIAMETER_NO_DSA_ESTABLISHED. Although the proxy agent MAY receive many PDSRs from access devices, only one DSA per realm need be established. Furthermore, the proxy is responsible for re-establishing the DSA prior to expiration without any involvement by the access device.

It is important to note that proxy agents establishing DSA's on behalf of a client will most likely need to reorder AVPs during the encryption process, in order to fit the encrypted AVPs within a CMS-Encrypted-Data AVP. This is contrary to the rule established in the Diameter base protocol [BASE], which states that proxy agents SHOULD NOT reorder AVPs.

Allowing the first hop agent to be used to establish the DSA with the home server may reduce the current concerns that the cryptographic operations resulting from this specification MAY overburden embedded access devices.

1.4 Using Redirect agents in lieu of DSA

When a redirect agent is used, allowing an access device, relay or proxy agent to communicate directly with the home server, the hop-by-hop security mechanisms specified in the base protocol may be sufficient.

However, there are certain business models where signing of selected Diameter AVPs (e.g. accounting) MAY be desired, even when redirect agents are used. Figure 4 shows an example where the relay agent contacts the redirect agent to retrieve the necessary information for it to communicate directly with the home server, which MAY include the home server's certificates.

The relay agent MAY then initiate a DSA with the home server, using the certificates provided by the redirect agents.

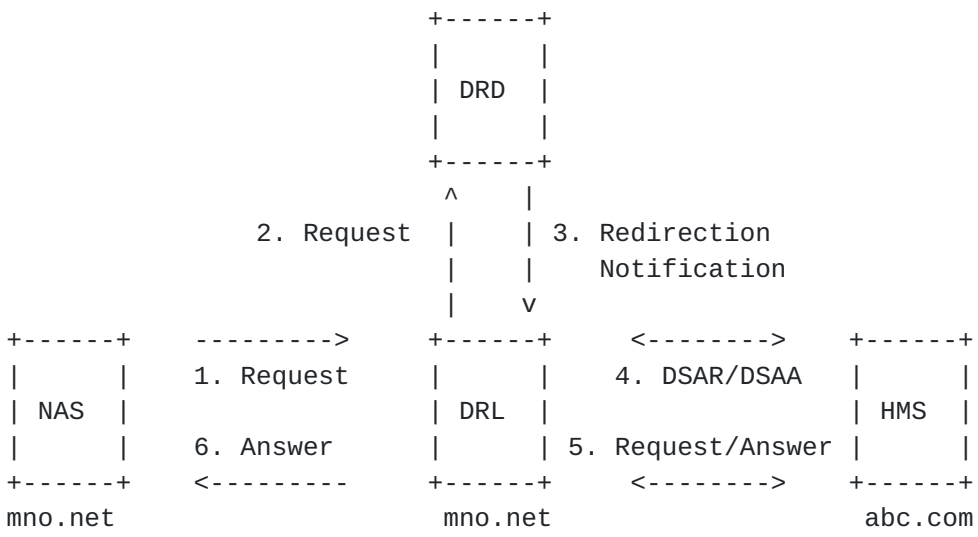

Figure 4: DSA Setup following redirect

The CMS specification allows for Diameter AVPs to be multiply-signed (see section 6.1), which may prove useful in business models that require both parties to sign accounting data in parallel. This scheme provides some assurance that both parties agreed to the accounting data, which MAY be used for settlement purposes.

1.5 When to use DSAs

Given that asymmetric transform operations are expensive, access devices and/or Diameter agents MAY wish to restrict establishment of a DSA to cases where the participants belong to a different administrative domain.

1.6 Advertising application support

Diameter nodes conforming to this specification MAY advertise support by including the value of two (2) in the Auth-Application-Id or the Acct-Application-Id AVP of the Capabilities-Exchange-Request and Capabilities-Exchange-Answer command [BASE].

1.7 CMS Processing of Grouped AVPs.

Grouped AVPs are processed as a whole, there is no partial signing or encryption mechanism defined.

Only the P flag (resp. MAY Encr) setting for a Grouped AVP need be set to Y to allow the Grouped AVP to be signed (resp. encrypted). That is, the AVPs within the Grouped AVP need not have their "P" bit

set (resp. MAY Encr) in order to be part of a signed (resp. encrypted) Grouped AVP.

Where a Grouped AVP is to be signed, implementations MAY set the "P" bit for each of the AVPs within the Grouped AVP. When verifying signatures over a Grouped AVP, implementations MUST NOT insist that the "P" bit has been set for AVPs within the group.

Where a Grouped AVP is to be encrypted, implementations MUST NOT fail encryption due to one of the members of the group being defined so as to prevent encryption ("MAY Encr" set to "N"). Similarly following decryption, implementations MUST NOT produce an error if one of the group members is defined so as to prevent encryption. The Grouped AVP itself, of course, MUST be defined to allow encryption.

Where a Grouped AVP is defined to disallow encryption then that Grouped AVP MUST NOT include any AVPs which are defined so as to allow encryption (since the member of the group might be erroneously sent in clear, if included in such a Grouped AVP).

2.0 AVP Format

The Diameter base protocol [BASE] details the AVP header, which includes the 'P' bit, but does not specify how the 'P' bit is used. The 'P' bit, known as the protected AVP bit, is used to indicate whether the AVP is protected by a digital signature. When set, the AVP is protected and the contents cannot be changed by agents without detection.

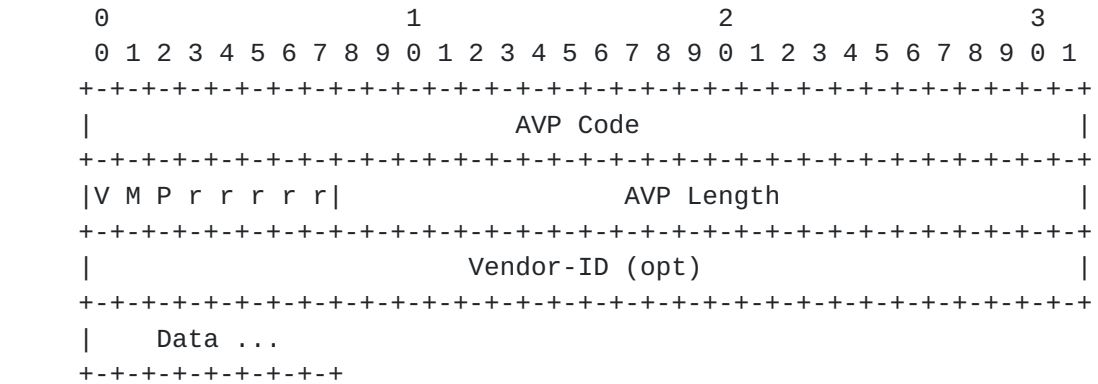

Figure 5: Diameter AVP Header

All Diameter specifications MUST specify whether the 'P' bit can be set or not, as is done in section 4.6 of [BASE] and section 6 below. For AVPs that are designed to be changed at each hop (such as the Proxy-Info AVP) Diameter nodes MUST NOT allow the 'P' bit to be set.

Diameter implementations MUST check whether AVPs with their 'P' bit

set are allowed to have that setting. If not, an appropriate error message MUST be issued containing DIAMETER_INVALID_AVP_BIT_COMBO result code.

3.0 Key Management

For origin authentication, CMS itself already provides sufficient key management without the need for additional specification. Basically, the originating Diameter node signs and includes whatever certificates are necessary for validation of the digital signature. Section 3.1 provides an example of how the Diameter CMS Security application is used.

In order to encrypt AVPs for a recipient, the originating Diameter node must have a copy of the recipient's public key. There are many well-known key retrieval schemes (e.g. LDAP [CERTLDAP]), but this specification also allows for the transportation of certificates within Diameter AVPs, which is expected to simplify implementations. Section 3.2 describes how Diameter node names are encoded within such certificates.

Finally, it is anticipated that the overhead of asymmetric encryption for each Diameter message sent to a given peer could be significant. Section 3.4 specifies how CMS encryption keys MAY be reused for multiple Diameter messages.

3.1 Usage Scenario

When a Diameter node is about to send a message, it must determine whether a DSA should be established or not. We assume the Diameter node knows the user's realm, perhaps through the User-Name AVP.

Implementations MAY cache the information required to establish a DSA. However, they MUST honor time-to-live settings so that certificates MUST re-validated (possibly including revocation checks) once the DSA has expired.

Revalidations SHOULD also occur before the DSA expires according to PKI policies. During the process of certificate path validation some implementations will calculate a duration for which the certificate path may be considered "safe". For example, if an implementation did not support certificate revocation checks, then the "safe" period would be from the time of initial validation until the earliest notAfter time in the set of certificates in the path. An implementation which does support certificate revocation checks will typically be able to calculate a "safe" period based additionally on

the earliest nextUpdate field in a CRL or OCSP response. Basically, the safe period is from the time of certificate validation to the earliest value from the set of notAfter and nextUpdate fields encountered during certificate validation.

However, implemetors should note that CAs MAY issue additional CRLs before the nextUpdate period. Generally it is a matter of local PKI policy as to whether an implementation will make additional checks even during the calculated "safe" period. For the purposes of this specification implementations are not required to make such checks and MAY assume that no re-validation of certificates is required during the "safe" period defined above.

We use Diameter Security Association Request (DSAR) and Diameter Security Association Answer (DSAA) messages to establish a DSA, which specifies which AVPs should be encrypted, signed or both, as well as which public key(s) to use.

The originating node sends the DSAR message to a server in the destination realm. The DSAR message contains:

- TTL for this DSA (seconds)
- the realm part of the user's NAI
- the list of direct trust CA's that the originating Diameter node has configured into it for certificate validation. A "direct trust" CA is one that the node is willing to use as the "top" of a certificate chain, sometimes confusingly known as a "root CA."
- a flag indicating whether the originating Diameter node wishes to receive certificate status information using OCSP messages. If this flag requires a fresh OCSP response, a nonce to be used by the destination Diameter node in OCSP requests MUST also be supplied. See [OCSP] for more details on the certificate status protocol and messages.

The destination node MUST check that the provided elements of the DSAR are valid. It MUST check, at least, that:

- Its local policy allows the given TTL, realm, AVP protection expectations, certification status, and other parameters.
- A common "top" of the certificate chain can be found between the home and foreign domains.

If these conditions can not be verified, the destination node MUST return a DSAA with the Result-Code AVP set to DIAMETER_NO_DSA_ESTABLISHED.

In the event the DSAR requested OCSP validation, via the OCSP-

Request-Flags AVP, and OCSP is not locally supported, the DSAA MUST be returned with the Result-Code AVP set to DIAMETER_OCSP_NOT_SUPPORTED. Otherwise, the destination node returns the DSAA message which contains:

- TTL for this DSA (seconds)
- a chain of CA certificates (possibly empty)
- public key certificates for the Diameter servers in the realm, all of which MUST validate up to one of the CA's contained in the DSAR message, via the chain of CA certificates above;
- (optionally, if OCSP an response was requested in the DSAR and OCSP is supported) a list of OCSP responses for the certificates in question. If a fresh response was required and a nonce value was included, each response will contain the nonce from the DSAR message

The originating Diameter node now has to check the response. Any failure results in error messages, auditing and not sending the Diameter message.

DSAA Checks:

- the certificate chain provided in the DSAA is cryptographically correct, passes the (relevant parts of the) path validation algorithm specified in [CERTPROF] and terminates at a CA mentioned in the DSAR message
- the name in the certificate is consistent with the rules detailed in section 3.2.
- the DSAA message MUST include the CMS-Signed-Data AVP, the signature MUST be validated and the signer's certificate chain MUST terminate as a CA mentioned in the DSAR message
- the expiration of the TTL MUST be less or equal to the earliest expiration of all certificates in the message, encoded in the notAfter field.

If the initiator's policy is such that certificate status is required, it MAY indicate that it requires an OCSP response from the DSA peer in the DSAA message, via the OCSP-Request-Flags AVP. Further, the initiator MAY request that the OCSP response be fresh (non-cached) via the OCSP-Request-Flags and OCSP-Nonce AVPs. Upon receipt of a DSAR message requesting an OCSP response, the receiver issues an OCSP request and returns the response within the DSAA message's OCSP-Responses AVP. The sender of the DSAA MAY included a cached OCSP response, unless the requestor specifically requested a fresh response.

The nonce value is to be (the beginning of) the nonce in the OCSP response. The reason for this is that the responder MAY add

additional bits to the nonce, but the nonce provided in the OSCP-Nonce MUST be present at the beginning of the nonce of the OSCP response.

The DSAR message MAY include the CMS-Signed-Data AVP. If the originating node has a private key, and it includes AVPs whose 'P' bit are set, the CMS-Signed-Data AVP MUST be present.

The DSAA MUST include the CMS-Signed-Data, signed by a Diameter agent or server within the user's realm, to prevent an intermediate node from modifying the protection expectations for AVPs.

Depending upon the security technique required (digital signature, encryption or both), then the originating node prepares the CMS related AVPs as required.

If certificate revocation is enabled, anytime a certificate is used from the local certificate cache, a revocation check MUST be performed.

Once the DSA is in place, any Diameter messages created by a DSA peer that has a private key MUST contain a signature over all AVPs whose definition states that their 'P' bit MAY be set.

Furthermore, these peers MUST encrypt any AVPs whose definition states that they MAY be encrypted.

Note: [BASE] includes the "MAY encr" column when describing AVPs. For the originator "MAY encr" as used in [BASE] means that if a message containing that AVP is to be sent via a proxy/agent (as opposed to directly) then the message MUST NOT be sent unless there is a DSA between the originator and the recipient OR the originator has locally trusted configuration that indicates that CMS need not be used.

3.2 Certificate Requirements

Certificates used for the purposes of Diameter MUST conform to the PKIX profile [CERTPROF], and MUST also include a Diameter node's FQDN, which is typically added in the Origin-Host AVP [BASE], as one of the values of the subjectAltName extension of the Certificate. The FQDN is to be encoded as a dNSName within the subjectAltName.

For Diameter nodes (capable of acting as recipients for confidentiality), the FQDN MUST be of the form "Diameter-<xxx>.<realm>". Other Diameter nodes MAY use this naming scheme. Note that this naming constraint is for PKI purposes only,

and in no way restricts a Diameter's host name.

The naming scheme presented here is intended to:

- make it simple to use existing certification authorities (CAs) for Diameter CMS
- allow CAs to ensure that only "proper" Diameter certificates are accepted by Diameter nodes
- allow Diameter certificates to be used for other purposes where that meets a CA's practices.

These names are used for two purposes:

1. Where a Diameter node is verifying a signature it needs to be able to compare the identity of the signer against the identity in the Origin-Host AVP.

2. Where a Diameter node is encrypting AVPs, it needs to be able to ensure that it uses a public key for the intended recipient. This requires comparing the identity in a Certificate against the FQDN of the intended recipient (which is assumed to be known).

In either case, the presence of the required FQDN as a dNSName value in the subjectAltName extension of a verified public key certificate satisfies the matching requirement.

Note that there MAY also be other values in the subjectAltName extension, (either using dNSName or other elements of the CHOICE), these can be safely ignored, but implementations MUST be able to handle their presence.

Note also that the PKIX profile [CERTPROF], section 4.1.2.6, specifies the rules for the relationship between the subjectAltName extension and the subject field of public key certificates.

For multiple Diameter servers within a realm that share a public key, each server's identity is encoded in the subjectAltName extension. This allows any server within a realm to decrypt AVPs intended for that realm.

Note that once operational experience has been gained, a future document may specify a restricted profile of [CERTPROF] in order to simplify implementation.

3.3 Algorithms

For all uses of CMS in this specification the mandatory to implement

algorithms are as follows:

- Hashing:
  sha-1 (see [CMS] section 12.1.1)
- Signature (the hash algorithm is specified separately):
  rsaEncryption (see [CMS] section 12.2.2)
- Content Encryption:
  des-ede3-cbc (see [CMS] section 12.4.1)
- Asymmetric key transport:
  rsaEncryption (see [CMS] 12.3.2.1)
- Symmetric key encryption (only needed in conjunction with [RCEK]):
  id-alg-CMS3DESwrap (see [CMS] section 12.3.3.1)

At some point in future, AES will replace 3DES.

3.4 Reuse of CMS Content Encryption Keys

This section describes an efficiency improvement which MAY be supported by Diameter nodes. If a node doesn't support this feature, then it MUST (and naturally will), treat all packets with re-used content encryption keys as a cryptographic failure. The originating node MAY then attempt to re-send the packets using asymmetric key transport. If a node does support this feature, then the MUST/SHOULD statements in this section apply, otherwise not.

Once a CMS-Encrypted-Data AVP has been exchanged between two Diameter peers, then they share a symmetric cryptographic key (the content encryption key) which can be used to encrypt further Diameter AVPs between the peers by using the scheme specified in [RCEK]. The peers MUST first take part in an DSAR/DSAA exchange in order to distribute the required asymmetric keys.

Although the use of symmetric encryption might be used to provide integrity or confidentiality, it does not provide data origin authentication with proof of origin.

[RCEK] leaves open some issues, namely how to handle loss of a shared secret (say following a peer re-boot) and for how long to continue to use a shared secret (the maximum number of decryptions required).

Where a Diameter node receives a CMS-Encrypted-Data AVP, but doesn't have the required shared secret, that node SHOULD return the DIAMETER_KEY_UNKNOWN error message. The peer MAY then use the DSAR/DSAA exchange to rebuild their Diameter security association.

In [RCEK], the default value for the maximum number of decryptions

allowed (CEKMaxDecrypts) when re-using a content encryption key is 1. In general this default SHOULD be used, but if a Diameter node "knows" that more than one CMS-Encrypted-Data AVP will be exchanged between the nodes, then the CEKMaxDecrypts setting MAY be set higher. Diameter nodes MUST be able to support a maxDecrypts setting of 1000.

Note that the CEKMaxDecrypts value used does not affect that DSA-TTL. The DSA-TTL dictates the lifetime of the DSA, while the CEKMaxDecrypts dictates how often rekeying will occur within the CMS protocol. A content encryption key MUST NOT be reused once the DSA has expired.

Implementations MUST be able to support a DSA-TTL of one day, and nodes which support certificate checking (e.g. CRLs, OCSP) that are re-establishing a DSA due to expiration of the TTL MUST re-validate the certificate.

4.0 Command-Codes Values

This section defines new Command-Code [BASE] values that MUST be supported by all Diameter implementations that conform to this specification. The following Command Codes are currently defined in this document:

Command-Name	Abbrev.	Code	Reference
Diameter-Security-Association-Request	DSAR	304	4.1
Diameter-Security-Association-Answer	DSAA	304	4.2
Proxy-Diameter-Security-Association-Request	PDSR	305	4.3
Proxy-Diameter-Security-Association-Answer	PDSA	305	4.4

4.1 Diameter-Security-Association-Request

The Diameter-Security-Association-Request command, indicated by the Command-Code field set to 304 and the 'R' bit set in the message flags field, is sent by a Diameter node to establish a Diameter Security Association. The DSAR message MUST NOT be used simply as a convenient certificate distribution protocol without establishing a DSA. The CMS-Signed-Data AVP MUST be present if any AVP in the message has the 'P' bit set.

Message Format

```
<DSAR> ::= < Diameter-Header: 304, REQ, PXY >
           { Origin-Host }
           { Origin-Realm }
           { Destination-Realm }
           { Auth-Application-Id }
           { OCSP-Request-Flags }
           { DSA-TTL }
         * { Local-CA-info }
        0*1[ CA-Chain ]
         * [ AAA-Node-Cert ]
        0*1[ OCSP-Nonce ]
        0*1[ Origin-State-Id ]
        0*1[ Destination-Host ]
        0*1[ CMS-Signed-Data ]
         * [ AVP ]
         * [ Proxy-Info ]
         * [ Route-Record ]
```

4.2 Diameter-Security-Association-Answer

The Diameter-Security-Association-Answer command, indicated by the Command-Code field set to 304, with the 'R' bit in the Command Flags cleared, in response to a DSAR. The CMS-Signed-Data AVP MUST be present if any AVP in the message has the 'P' bit set. If the Result-Code AVP indicates success, the CA-Chain, AAA-Node-Cert, DSA-TTL and CMS-Signed-Data AVPs MUST be present.

Message Format

```
<DSAA> ::= < Diameter-Header: 304, PXY >
           { Result-Code }
           { Origin-Host }
           { Auth-Application-Id }
           { DSA-TTL }
         * { Local-CA-info }
        0*1[ CA-Chain ]
         * [ AAA-Node-Cert ]
        0*1[ Origin-Realm ]
        0*1[ Error-Message ]
        0*1[ Error-Reporting-Host ]
         * [ OCSP-Responses ]
        0*1[ CMS-Signed-Data ]
        0*1[ Origin-State-Id ]
         * [ AVP ]
         * [ Proxy-Info ]
```

4.3 Proxy-Diameter-Security-Association-Request

The Proxy-Diameter-Security-Association-Request command, indicated by the Command-Code field set to 305 and the 'R' bit set in the Command Flags field, is sent by a Diameter node to request that a downstream proxy establishes a Diameter Security Association with a server in a given realm on its behalf.

Message Format

```
<PDSR> ::= < Diameter-Header: 305, REQ, PXY >
           { Origin-Host }
           { Origin-Realm }
           { Destination-Realm }
           { Auth-Application-Id }
         0*1[ DSAR-Target-Realm ]
         0*1[ Origin-State-Id ]
         0*1[ Destination-Host ]
          * [ AVP ]
          * [ Proxy-Info ]
          * [ Route-Record ]
```

4.4 Proxy-Diameter-Security-Association-Answer

The Proxy-Diameter-Security-Association-Answer command, indicated by the Command-Code field set to 305 and the 'R' bit cleared in the Command Flags field, is sent by a Diameter node in response to an PDSR message.

Message Format

```
<PDSA> ::= < Diameter-Header: 305, PXY >
           { Result-Code }
           { Origin-Host }
           { Origin-Realm }
           { Auth-Application-Id }
           { DSA-TTL }
         0*1[ Error-Message ]
         0*1[ Error-Reporting-Host ]
         0*1[ Origin-State-Id ]
          * [ Redirect-Host ]
         0*1[ Redirect-Host-Usage ]
         0*1[ Redirect-Max-Cache-Time ]
          * [ AVP ]
          * [ Proxy-Info ]
```

5.0 Diameter Security Association Message Flow

This section contains an example of a NAS in realm xyz.com, communicating with its local relay agent, which in turn communicates with a server in ABC.COM's network. In the following example, once the initial capabilities exchange is complete, the NAS receives a request for access from alice@abc.com, which causes the DSA setup to be initiated, followed by the application-specific authentication request.

Although the example doesn't specifically use bi-directional digital signature and encryption, this feature is supported.

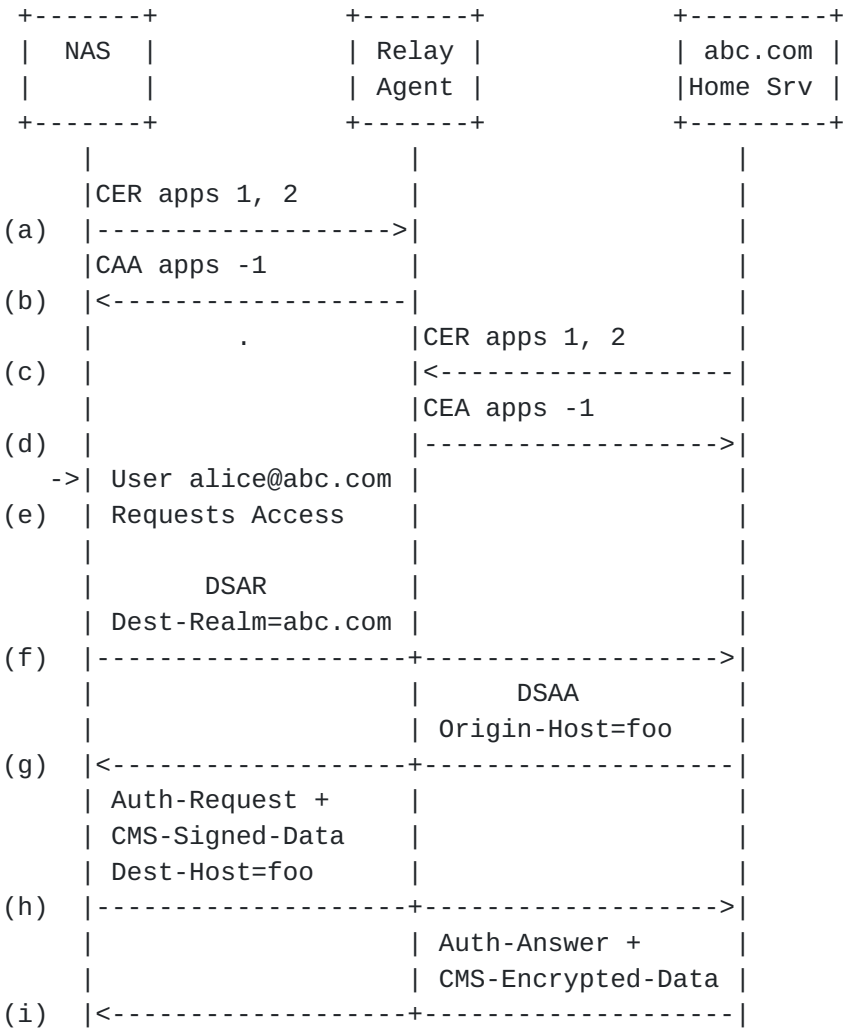

Figure 6: Example of a DSA Setup

(a) NAS sends a CER message to its relay agent indicating that it supports applications 1 (NASREQ) and 2 (CMS Security).

(b) The relay agent sends a CEA message to the NAS indicating that

it is a relay supporting all Diameter applications.

(c) ABC.COM's Home Server sends a CER message to a relay agent indicating that it supports applications 1 (NASREQ) and 2 (CMS Security).

(d) The relay agent sends a CEA message to ABC.COM's Home Server indicating that it is a relay supporting all Diameter applications.

(e) The NAS receives a request for access from a user (alice@abc.com).

(f) The NAS issues an DSAR message, with the Destination-Realm AVP set to abc.com.

(g) ABC.COM's Home Server processes the DSAR message, and replies with the DSAA message.

(h) The NAS issues an authentication request with the Destination-Host AVP set to the value of the Origin-Host AVP in the DSAA. The message includes the CMS-Signed-AVP, which authenticates the AVPs that were requested by the Home Server in the DSAA.

(i) The Home Server successfully authenticates the user, and returns a reply, which includes the CMS-Encrypted-Data AVP, whose contents include the AVPs that require encryption.

6.0 CMS Security AVPs

This section contains AVPs that are used to establish a Diameter Security Association, and to transport CMS objects. Except as specifically constrained, the profile of CMS algorithm and structure usage is as specified in the S/MIME v3 message specification [MSG].

Attribute Name	AVP Code	Section Defined	Value Type	AVP Flag rules: MUST	MAY	SHLD NOT	MUST NOT	MAY Encr
AAA-Node-Cert	351	6.6	OctetString	M,P			V	N
CA-Chain	353	6.8	OctetString	M	P		V	N
CA-Name	349	6.4.1	UTF8String	M	P		V	N
CMS-Encrypted-Data	355	6.2	OctetString	M	P		V	N
CMS-Signed-Data	310	6.1	OctetString	M			P,V	N
DSA-TTL	362	6.11	Unsigned32	M	P		V	N
DSAR-Target-Realm	360	6.10	UTF8String	M	P		V	N
Key-Hash	350	6.4.2	OctetString	M	P		V	N
Local-CA-Info	348	6.4	Grouped	M	P		V	N
OCSP-Nonce	358	6.5	OctetString	M	P		V	N
OCSP-Request-Flags	361	6.9	Enumerated	M	P		V	N
OCSP-Responses	359	6.7	OctetString	M	P		V	N

The profile of CMS algorithm and structure usage is conformant to that specified in the S/MIME v3 message specification [MSG]. This makes is simpler to base an implementation of this specification upon an existing S/MIME toolkit.

No MIME encoding of binary data is required for this specification. This is different from the use of CMS in S/MIME, but is acceptable since Diameter is a binary protocol and investigation has not shown this to cause problems when using existing CMS & S/MIME toolkits.

6.1 CMS-Signed-Data AVP

The CMS-Signed-Data AVP (AVP Code 310) is of type OctetString and contains the Basic Encoding Rules (BER) encoding of a CMS object [CMS] of type ContentInfo. This means that where a set of AVPs is to be signed, the set of AVPs with the 'P' bit set MUST first be catenated together in the order in which they occur in the Diameter message. The result of this encoding is used as input into the signing process.

Note that the AVPs themselves are not encapsulated within the CMS-Signed-Data AVP. Instead, the digest value of the AVPs produced in the signature process MUST be included in the CMS-Signed-Data AVP, as a message-digest attribute (defined in section 11.2 of [CMS]) in the

SignerInfo value.

Multiple Diameter entities MAY add their signatures to an existing CMS-Signed-Data AVP. Multiple signatures are added within the countersignature attribute (defined in section 11.4 of [CMS]) and not as additional SignerInfo values. The countersignature attribute requires that the signatures occur sequentially, meaning that each signature covers the existing signatures in the CMS object.

The initial signature, and any additional countersignatures, MUST cover the exact same set of AVPs, in the order they are present in the message.

Note that the CMS-Signed-Data AVP itself MUST NOT be used in the generation of the signature, and therefore MUST NOT have its 'P' bit set.

The eContent field of the EncapsulatedContentInfo structure MUST be absent since the digital signature covers data outside of the object.

If a receiver cannot verify correctly the signature carried by the CMS-Signed-Data AVP, it SHOULD return the DIAMETER_INVALID_AUTH Result-Code AVP value defined in section 7.1.

When AVPs are to be both encrypted and signed, the CMS-Encrypted-Data AVP MUST be created first. This AVP MUST then have the 'P' bit set and be one of the inputs to the signing process as described above. (Any other processing resulting in the same output can be used.) This means that signing is "outside" encryption.

No more than one CMS-Signed-Data AVP MUST be present in any given Diameter message.

6.2 CMS-Encrypted-Data AVP

The CMS-Encrypted-Data AVP (AVP Code 355) is of type OctetString with the OctetString containing the Basic Encoding Rules (BER) encoding of a CMS object [CMS] of type ContentInfo.

All AVPs to be encrypted are concatenated. This value is then:

- encrypted according to normal CMS rules,
- used as the value of the EncryptedContent field within EnvelopedData.

  The contentType of the EncryptedContentInfo value MUST be id-data [MSG].

A CMS-Encrypted-Data AVP contains exactly one EnvelopedData. Where one or more AVP would be encrypted within separate EnvelopedData structures, then separate CMS-Encrypted-Data AVPs MUST be used.

Thus, implementations MUST be able to support the presence of multiple CMS-Encrypted-Data AVPs and MUST be able to decrypt any EnvelopedData for which it is a recipient, as indicated in the EnvelopedData's RecipientInfos field [CMS].

If the recipient is not specified in a RecipientInfo, it MAY choose to process the message or return an answer with the Result-Code AVP set to DIAMETER_NO_DSA_RECIPIENT. If the recipient is in the RecipientInfos and an error occurs during decryption, then the recipient MUST answer with the Result-Code set to DIAMETER_INVALID_AVP_VALUE.

Diameter nodes SHOULD implement content encryption key reuse (see section 3.4 above).

If a receiver detects that the contents of the CMS-Encrypted-Data AVP are invalid, it SHOULD answer with the Result-Code set to DIAMETER_INVALID_CMS_DATA.

Zero or more CMS-Encrypted-Data AVP MAY be present in any Diameter message.

6.3 Example Encodings

In order to clarify the contents of and the relationships between CMS-Signed-Data and CMS-Encrypted-Data AVPs we present the following example of how these AVPs are calculated.

First, some short-hand:

- The "|" character represents concatenation
- EnvelopedData-fnc(x,y) represents the EnvelopedData produced as output of a function with x as the to-be-encrypted-data and y as the parameters (e.g. recipient information).
- SignedData (y) represents the SignedData produced as output of a function with the catenation of the 'P is set' AVPs as the to-be-digested-data (which is not part of the output!) and y as the parameters (e.g. signer information).

The scenario calls for a message containing 7 AVPs s,t,e,p,h,e' and n to meet the following:

AVPs s, t and e are to be encrypted for recipient P.

```
   AVPs e, p and h are to be encrypted for recipient A.
   AVPs s, and e' are to be signed by originator T.
   AVP  s is to be sent to recipient A.
   AVP  n needs neither signing nor encryption.
```

Note that though there is no explicit requirement that AVP s be encrypted for A, since it will be encrypted for P, we also have to encrypt it for A. Implementations SHOULD NOT send the same AVP both encrypted and in clear.

The resulting message will look like:

```
   AVP1='P is set',   EnvelopedData-fnc(s|t|e,P)
   AVP2='P is set',   EnvelopedData-fnc(s|e|p|h,A)
   AVP3='P is set',   e'
   AVP4='P is clear', n
   AVP5='P is clear', SignedData(T)
```

The result of this is that all AVPs except n are actually signed even though signing of t and e wasn't explicitly required. However, this is no harm.

6.4 Local-CA-Info AVP

The Local-CA-Info AVP (AVP Code 348) is of type Grouped. The Grouped Data field has the following ABNF grammar:

```
   Local-CA-Info ::= < AVP Header: 348 >
                     { CA-Name }
                     { Key-Hash }
```

6.4.1 CA-Name AVP

The CA-Name AVP (AVP Code 349) is of type UTF8String. The AVP contains the DN (in LDAP string syntax [LDAPSTR]) of the Certificate Authority, e.g. "CN=CA;O=Baltimore Technologies;C=IE".

6.4.2 Key-Hash AVP

The Key-Hash AVP (AVP Code 350) is of type OctetString, and contains the SHA-1 hash of a public key.

The hash MUST be calculated over the representation of the CA public key which would be present in an X.509 public key certificate, specifically, the input for the hash algorithm MUST be the DER encoding of a SubjectPublicKeyInfo representation of the key. Note: This includes the AlgorithmIdentifier as well as the BIT STRING. The

rules given in [CERTPROF] for encoding keys MUST be followed.

Since this AVP is used for indexing and not for security (since Diameter nodes SHOULD validate certificates), there is no need to support more than one hash algorithm here.

6.5 OCSP-Nonce AVP

The OCSP-Nonce AVP (AVP Code 358) is of type OctetString, and contains a random value (RECOMMENDED to be at least 128 bits) generated by the Diameter node.

6.6 AAA-Node-Cert AVP

The AAA-Node-Cert AVP (AVP Code 351) is of type OctetString and contains a public key certificate for the AAA node. Note: this AVP contains no CA certificates, just the end-entity certificate. Certificates MUST follow the naming conventions described in section 3.2.

6.7 OCSP-Responses AVP

The OCSP-Responses AVP (AVP Code 359) is of type OctetString, and contains an OCSP response message from an OCSP responder. If the OCSP-Request-Flags AVP indicating a response was required in the corresponding request message, the OCSP-Responses AVP MUST be present. Furthermore, the OCSP-Request-Flags AVP MAY request a fresh OCSP response message, which MUST also include the OCSP-Nonce AVP.

6.8 CA-Chain AVP

The CA-Chain AVP (AVP Code 353) is of type OctetString, and contains a certificate chain, from one of the nominated locally trusted CAs down to the (one and only) CA which has issued the end entity certificates in the AAA-Node-Cert AVP. The OctetString contains a CMS "certs-only" message.

To produce this AVP in an DSAA message, one (and only one) of the Local-CA-info values from the corresponding DSAR message is selected (call this the "top" CA for the purposes of this description). This AVP then contains a certificate path (in order) from the "top" CA down to the (one and only) CA which has issued the end entity certificate in the AAA-Node-Cert AVP. The (typically self-signed), certificate of the "top" CA MUST NOT be included.

6.9 OCSP-Request-Flags AVP

The OCSP-Request-Flags AVP (AVP Code 361) is of type Enumerated, and specifies whether the sender wishes to receive an OCSP response. The following values are defined:

NO_OCSP_RESPONSE 0
The sender does not wish to receive an OCSP Response.

OCSP_RESPONSE 1
The sender wishes to receive an OCSP Response, and is willing to accept a stale response.

OCSP_FRESH_RESPONSE 2
The sender wishes to receive a fresh OCSP Response. When this value is set, the OCSP-Nonce AVP MUST be present.

6.10 DSAR-Target-Realm AVP

The DSAR-Target-Realm AVP (AVP Code 360) is of type UTF8String, and contains the Destination-Realm of the resulting DSAR sent by a non-transparent proxy.

6.11 DSA-TTL AVP

The DSA-TTL AVP (AVP Code 362) is of type Unsigned32, and contains the time to live (in seconds) of the Diameter Security Association. The expiration time (now+TTL) MUST NOT be greater than the earliest expiration time (NotAfter field) of all certificates included in this message accompanying this AVP. The DSA-TTL AVP in the DSAA MUST NOT be greater than the DSA-TTL AVP in the DSAR. A DSA-TTL AVP MUST also be included in the PDSA in order to provide information about the lenght of the DSA established by the proxy on behalf of the access device. Implementations MUST be able to support a DSL-TTL of one day.

7.0 Result-Code AVP Values

This section defines new Result-Code [BASE] values that MUST be supported by all Diameter implementations that conform to this specification.

7.1 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 4008
This error code is returned when a CMS-Signed-Data or CMS-Encrypted-Data AVP is received that was generated using a key that is not locally recognized. This error could be caused if one of the participants of a DSA lost a previously agreed upon key, perhaps as a result of a reboot.

DIAMETER_NO_CMS_THROUGH_PROXY 4009
This error code is returned when a non-transparent proxy receives an DSAR message to state that it doesn't allow a DSA through it since it plans to modify AVPs.

DIAMETER_CAN_ACT_AS_CMS_PROXY 4010
This error code is returned when a non-transparent proxy receives an DSAR message, and although it doesn't allow a DSA through it, it is willing to initiate a DSA on behalf of the access device.

DIAMETER_OCSP_NOT_SUPPORTED 4011
This error code is returned when a DSAR message is received requesting OCSP validation, and the receiver does not support OCSP.

DIAMETER_INVALID_AUTH 4012
The signature in the CMS-Signed-Data AVP is invalid.

DIAMETER_MISSING_SIGNED_AVPS 4013
Some AVPs within a Diameter message were expected to be signed.

7.2 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_INVALID_CMS_DATA 5019
This error code is returned when a CMS-Data AVP is received with an invalid ContentInfo object.

DIAMETER_NO_COMMON_TRUST 5020
This error code is returned when a receiver receives a DSAR for which it has no common trust with the sender, which is required to establish the DSA.

DIAMETER_NO_DSA_ESTABLISHED 5021
A Diameter message refers to a Diameter Security Association which does not exist.

DIAMETER_DSA_EXPIRED 5022
A Diameter message refers to a Diameter Security Association which has expired.

DIAMETER_NO_DSA_RECIPIENT 5023
A Diameter message was received with encrypted data, and the local Diameter node is not a potential recipient of the EnvelopedData.

8.0 AVP Occurrence Tables

The table in this section presents the AVPs defined in this document, and specifies in which Diameter messages they MAY, or MAY NOT be present. Note that AVPs that can only be present within a Grouped AVP are not represented in this table.

The table uses the following symbols:

0	The AVP MUST NOT be present in the message.
0+	Zero or more instances of the AVP MAY be present in the message.
0-1	Zero or one instance of the AVP MAY be present in the message.
1	One instance of the AVP MUST be present in the message.
1+	At least one instance of the AVP MUST be present in the message.

Attribute Name	Command-Code			
	DSAR	DSAA	PDSR	PDSA
Auth-Application-Id	1	1	1	1
Destination-Host	0-1	0	0	0
Destination-Realm	1	0	1	0
Error-Message	0	0-1	0	0-1
Error-Reporting-Host	0	0-1	0	0-1
AAA-Node-Cert	0-1	0-1	0	0
CA-Chain	0-1	0-1	0	0
CA-Name	0	0	0	0
CMS-Encrypted-Data	0	0	0	0
CMS-Signed-Data	0-1	0-1	0	0
DSA-TTL	1	1	0	0
DSAR-Target-Realm	0	0	0-1	0
Local-CA-Info	0+	0+	0	0
OCSP-Nonce	0-1	0	0	0
OCSP-Request-Flags	1	0	0	0
OCSP-Responses	0	1+	0	0
Origin-Host	1	1	1	1
Origin-Realm	1	1	1	1
Original-State-Id	0-1	0-1	0-1	0-1
Proxy-Info	0+	0+	0+	0+
Redirect-Host	0	0+	0	0+
Redirect-Host-Usage	0	0-1	0	0-1
Redirect-Max-Cache-Time	0	0-1	0	0-1
Result-Code	0	1	0	1
Route-Record	0+	0	0+	0

9.0 IANA Considerations

This section contains the namespaces that have either been created in this specification, or the values assigned to existing namespaces managed by IANA.

9.1 Command Codes

This specification assigns the value 304 and 305 from the Command Code namespace defined in [BASE]. See section 4.0 for the assignment of the namespace in this specification.

9.2 AVP Codes

This specification assigns the values 348-351, 353, 355, 358-362 from the AVP Code namespace defined in [BASE]. See section 6.0 for the assignment of the namespace in this specification.

9.3 Result-Code AVP Values

This specification assigns the values 4008-4011, 5019-5023 from the Result-Code AVP (AVP Code 268) value namespace defined in [BASE]. See section 7.0 for the assignment of the namespace in this specification.

9.4 Application Identifier

This specification assigns the value two (2) to the Application Identifier namespace defined in [BASE]. See section 1.6 for more information.

9.5 OCSP-Request-Flags AVP Values

As defined in Section 6.9, the OCSP-Request-Flags AVP (AVP Code 361) defines the values 0-2. All remaining values are available for assignment via IETF Consensus [HUH???????].

10.0 Security Considerations

This document describes how CMS security can be achieved in the Diameter protocol by allowing S/MIME Cryptographic Message Syntax [CMS] objects to be carried as a Diameter AVP.

This specification does not mandate certificate revocation, however not verifying whether a given certificate has been revoked has serious implications, and MAY create a security hole.

The PDSR and PDSA messages (sections 4.3 and 4.4) allow a third party proxy to establish a Diameter security association with a Diameter server in a target realm. An access device MUST ensure that the server establishing the SA on its behalf is a trusted entity, since the proxy in question could modify Diameter messages, which would be very difficult to trace.

11.0 References

Normative References

[BASE]
P. Calhoun, J. Arkko, E. Guttman, G. Zorn, J. Loughney, "Diameter Base Protocol", draft-ietf-aaa-diameter-09.txt, IETF work in progress, March, 2002.

[CERTLDAP]
Boyen, Howes, Richard, "Internet X.509 Public Key Infrastructure Operational Protocols - LDAPv2", RFC 2559, April 1999.

[CERTPROF]
Housley, Ford, Polk, Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", RFC 2459, January 1999.

[CMS]
R. Housley, "Cryptographic Message Syntax", RFC 2630, June 1999.

[LDAPSTR]
M. Wahl, S. Kille, T. Howes, "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[MSG]
B. Ramsdell, "S/MIME Version 3 Message Specification", RFC 2633, June 1999.

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

[OCSP]
Myers, Ankney, Malpani, Galperin, Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol (OCSP)", RFC 2560, June 1999.

[RCEK]
Farrell, Turner, "Reuse of CMS Content Encryption Keys", RFC 3185, October 2001.

Informative References

[AAAREQS]
Aboba et al., "Criteria for Evaluating AAA Protocols for Network Access", RFC 2989, November 2000.

[CDMAREQ]
T. Hiller et al., "Cdma2000 Wireless Data Requirements for AAA", RFC 3141, June 2001.

[MIPREQ]
S. Glass, S. Jacobs, C. Perkins, "Mobile IP Authentication, Authorization, and Accounting Requirements". RFC 2977. October 2000.

12.0 Acknowledgements

The authors would also like to acknowledge the following people for their contribution in the development of this specification:

Bernard Aboba, Jari Arkko, Steven Bellovin and Miguel A. Monjas

Finally, Pat Calhoun would like to thank Sun Microsystems since most of the effort put into this document was done while he was in their employ.

13.0 Authors' Addresses

Questions about this memo can be directed to:

Pat R. Calhoun
Black Storm Networks
250 Cambridge Avenue, Suite 200
Palo Alto, California, 94306
USA

Phone: +1 650-617-2932
Fax: +1 650-786-6445
E-mail: pcalhoun@diameter.org

Stephen Farrell
Baltimore Technologies
39 Parkgate Street,
Dublin 8,
IRELAND

Phone: +353-1-881-6000
Fax: +353-1-881-7000
E-Mail: stephen.farrell@baltimore.ie

William Bulley

Merit Network, Inc.
Building One, Suite 2000
4251 Plymouth Road
Ann Arbor, Michigan, 48105-2785
USA

Phone: +1 734-764-9993
Fax: +1 734-647-5185
E-mail: web@merit.edu

14.0 Full Copyright Statement

15.0 Expiration Date

This memo is filed as <draft-ietf-aaa-diameter-cms-sec-04.txt> and expires in September 2002.