Network Working Group
Internet-Draft
Updates: 2510 (if approved)
Expires: August 9, 2004

A. Kapoor
R. Tschalar
Certicom
T. Kause
SSH

February 9, 2004

Internet X.509 Public Key Infrastructure -- Transport Protocols for CMP
draft-ietf-pkix-cmp-transport-protocols-05.txt

Abstract

This document describes how to layer Certificate Management Protocols over various transport protocols.

Table of Contents

1. Introduction

Well defined transport mechanisms are required for Certificate Management Protocol [CMP] in order to allow end entities, RAs and CAs to pass PKI messages between them.

2. Requirements

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

3. TCP-Based Management Protocol

While this section is called TCP-Based and the messages are called TCP-message's, the same protocol can be used over any reliable, connection oriented transport protocol (e.g. SNA, DECnet, etc.). This protocol is suitable for cases where an end entity (or an RA) initiates a transaction and can poll to pick up the results.

The client sends a TCP-message to the server, and the server responds with another TCP-message. Note that a response MUST be sent for every request, even if the encapsulated CMP message in the request does not have a corresponding response.

The protocol basically assumes a listener process on an RA or CA which can accept TCP-messages on a well-defined port (default port number is 829). Typically a client initiates connection to the server and submits a PKI message. The server replies with a PKI message or with a reference number to be used later when polling for the actual PKI message response.

If a polling-reference was supplied then the client will send a polling request using this polling-reference after waiting for at least the specified time. The server may again reply with a polling-reference or with the actual PKI message response.

When the final PKI response message has been picked up by the client then no new polling reference is supplied.

If a transaction is initiated by a PKI entity (RA or CA) then an end entity must either supply a listener process or be supplied with a polling reference (see below) in order to allow it to pick up the PKI message from the PKI management component.

3.1 General Form

A TCP-message consists of:

length (32-bits)

version (8-bits)

flags (variable length)

message-type (8-bits)

value (defined below)

The length field contains the number of octets of the remainder of

the TCP-message (i.e., number of octets of <value> plus <flags-length> plus 2). All bit values in this protocol are specified to be in network byte order

The version field indicates the version of the TCP-message. It MUST be incremented for each specification which changes the flags field in a way that is not fully backwards compatible with the previous version (e.g. when the length of the flags field is changed).

The flags field is for transporting TCP-message specific data. The length of this field is version dependent and is fixed for a given version.

The message-type field is used to indicate the type of TCP-message.

The value field contains message-type dependent data.

3.2 Version Negotiation

If a client knows the protocol version(s) supported by the server (e.g. from a previous TCP-message exchange or via some out-of-band means) then it SHOULD send a TCP-message with the highest version supported both by it and the server. If a client does not know what version(s) the server supports then it SHOULD send a TCP-message using the highest version it supports.

If a server receives a TCP-message version that it supports, then it MUST reply with a TCP-message of the same version. If the version received is higher than what the server supports, it MUST send back a VersionNotSupported errorMsgRep (defined below) containing the highest version it supports.

3.3 TCP-message Version 10

The TCP-message version will be 10 for this document. The number has deliberately been chosen to prevent [RFC2510] compliant applications from treating it as a valid message type. Applications receiving a version less than 10 SHOULD interpret the message as being an [RFC2510] style message.

The length of the flags field for this version is 1 octet. The LSB is used to indicate a connection close; all other bits in the flags octet MUST be ignored by receivers, and MUST be set to zero by senders.

By default connections are kept open after the receipt of a response. Either party (client or server) MAY set the connection close bit at any time. If the connection close bit is set on a request, then the

server MUST set the bit in the response and close the connection after sending the response. If the bit is set on a response from the server, the client MUST NOT send any further requests on that connection. Applications MAY decide to close an idle connection (one on which no response is outstanding) after some time-out. Because of the problem where a client sends a request and the server closes the connection while the request is still in flight, clients SHOULD automatically retry a request for which no part of the response could be read due to a connection close or reset.

If the connection is kept open, it MUST only be used for subsequent request/response transactions started by the client - the server MUST NOT use it to send requests to the client. Different transactions may be freely interwoven on the same connection. E.g. a CR/CP need not immediately be followed by the Confirm, but may be followed by any other request from a different transaction.

3.4 Detecting and Interoperating with RFC-2510 Conformant Implementations

Servers wishing to interoperate with clients conforming to [RFC2510] can do so by treating any received message with a version less than 10 as an [RFC2510] message and responding in that format. Servers not wishing to support [RFC2510] messages MUST respond with a [RFC2510] errorMsgRep.

Clients wishing to interoperate with [RFC2510] compliant servers SHOULD treat a response with a version less than 10 as an [RFC2510] style message. If this message is an errorMsgRep (message-type 06) then the client MAY automatically retry the request using the [RFC2510] format; if the message is not an errorMsgRep or the implementation does not wish to support [RFC2510] then it MUST abort the corresponding CMP transaction.

3.5 Message Types

message-types 0-127 are reserved and will be issued under IANA auspices. message-types 128-255 are reserved for application use.

The message-type's currently defined are:

Message name	Message-type
pkiReq	'00'H
pollRep	'01'H
pollReq	'02'H

finRep	'03'H
pkiRep	'05'H
errorMsgRep	'06'H

If server receives an unknown message-type then it MUST reply with an InvalidMessageType errorMsgRep. If a client receives an unknown message-type then it MUST abort the CMP transaction.

The different TCP-messages are discussed in the following sections:

3.5.1 pkiReq

The pkiReq is to be used to carry a PKIMessage from the client to the server. The <value> portion of this TCP-message will contain: DER-encoded PKIMessage.

The type of PKIMessages that can be carried by this TCP-message are:

CRL Announcement

Certificate Confirmation

Poll Request

Subscription Request

CA Key Update Announcement

Certificate Announcement

Certification Request

Cross-Certification Request

Error Message

General Message

Initialization Request

Key Recovery Request

Key Update Request

Nested Message

PKCS-10 Request

POP Response

Revocation Request

3.5.2 pkiRep

This TCP-message is to be used to send back the response to the request. The <value> portion of the pkiRep will contain: DER encoded PKI message

The type of PKIMessages that can be carried by this TCP-message are:

Confirmation

Poll Response

Subscription Response

Certification Response

Error Message

General Response

Initialization Response

Key Recovery Response

Key Update Response

POP Challenge

Revocation Response

3.5.3 pollReq

The pollReq will be the used by the client to check the status of a pending TCP-message. The <value> portion of the pollReq will contain:

polling-reference (32 bits)

The <polling-reference> MUST be the one returned via the pollRep TCP-message.

3.5.4 pollRep

The pollRep will be the response sent by the server to the client when there are no TCP-message response ready. The <value> portion of the pollRep will contain:

polling-reference (32 bits)

time-to-check-back (32 bits)

The <polling-reference> is a unique 32-bit number sent by the server. The <time-to-check-back> is the time in seconds indicating the minimum interval after which the client SHOULD check the status again. The duration for which the server keeps the <polling-reference> unique is left to the implementation.

3.5.5 finRep

finRep is sent by the server whenever no other response applies (such as after receiving a CMP pkiConf), and usually indicates the end of the CMP transaction. The <value> portion of the finRep SHALL contain:

'00'H (8 bits)

3.5.6 errorMsgRep

This TCP-message is sent when a TCP-message level protocol error is detected. Please note that PKIError messages MUST NOT be sent using this. Examples of TCP-message level errors are:

1. Invalid protocol version

2. Invalid TCP message-type

3. Invalid polling reference number

The %lt;value> field of the TCP-message SHALL contain:

error-type (16-bits)

data-length (16-bits)

data (<data-length> octets)

UTF8 String (SHOULD include an [RFC3066] language tag, as described in [RFC2482])

The <error-type> is of the form MMNN where M and N are hex digits (0-F) and MM represents the major category and NN the minor. The major categories defined by this specification are:

'01'H	TCP-message version negotiation
'02'H	client errors
'03'H	server errors

The major categories '80'H-'FF'H are reserved for application use.

The <data-length> and <data> are additional information about the error to be used by programs for further processing and recovery. <data-length> contains the length of the <data> field in number of octets. Error messages not needing additional information to be conveyed MUST set the <data-length> to 0.

The UTF8 text string is for user readable error messages. Note that it does not contain a terminating null character at the end.

3.5.7 VersionNotSupported errorMsgRep

The VersionNotSupported errorMsgRep is defined as follows:

error-type:	'0101'H
data-length:	1
data:	<version>
UTF8-text String:	implementation defined

where <version> is the highest version the server supports.

3.5.8 GeneralClientError errorMsgRep

The GeneralClientError errorMsgRep is defined as follows:

error-type:	'0200'H
data-length:	0
data:	<empty>
UTF8-text String:	implementation defined

3.5.9 InvalidMessageType errorMsgRep

The InvalidMessageType errorMsgRep is defined as follows:

error-type:	'0201'H
data-length:	1
data:	<message-type>
UTF8-text String:	implementation defined

where <message-type> is the message-type received by the server.

3.5.10 InvalidPollID errorMsgRep

The InvalidPollID errorMsgRep is defined as follows:

error-type:	'0202'H
data-length:	4
data:	<polling-reference>
UTF8-text String:	implementation defined

where <polling-reference> is the polling-reference received by the server.

3.5.11 GeneralServerError errorMsgRep

The GeneralServerError errorMsgRep is defined as follows:

error-type:	'0300'H
data-length:	0
data:	<empty>
UTF8-text String:	implementation defined

4. HTTP-Based Management Protocol

A client creates a TCP-message, as specified in section 2.0 XXX . The message is then sent as the entity-body of an HTTP POST request as specified in [RFC2616]. If the HTTP request is successful then the server returns a similar message in the body of the response. The response status code in this case MUST be 200; other 2xx codes MUST NOT be used. The content type of the request and response MUST be "application/pkixcmp". Applications MAY wish to also recognized and use the "application/pkixcmp-poll" MIME type (specified in earlier versions of this document) in order to support backward compatibility wherever applicable. Content codings may be applied.

Note that a server may return any 1xx, 3xx, 4xx, or 5xx code if the HTTP request needs further handling or is otherwise not acceptable.

Because in general CMP messages are not cacheable, requests and responses should include a "Cache-Control: no-cache" (and, if either side uses HTTP/1.0, a "Pragma: no-cache") to prevent the client from getting cached responses. This is especially important for polling requests and responses.

Connection management SHOULD be based on the HTTP provided mechanisms (Connection and Proxy-Connection header fields) and not on the connection flag carried in the TCP-message.

5. File based protocol

A file containing a PKI message MUST contain only the DER encoding of one PKI message, i.e., there MUST be no extraneous header or trailer information in the file.

Such files can be used to transport PKI messages using, e.g., FTP.

6. Mail based protocol

This subsection specifies a means for conveying ASN.1-encoded messages for the protocol exchanges via Internet mail ([RFC821]. A simple MIME object is specified as follows.

```
Content-Type: application/pkixcmp
Content-Transfer-Encoding: base64

<<the ASN.1 DER-encoded PKIX-CMP message, base64-encoded>>
```

This MIME object can be sent and received using common MIME processing engines and provides a simple Internet mail transport for PKIX-CMP messages. Implementations MAY wish to also recognize and use the "application/x-pkixcmp" MIME type (specified in earlier versions of this document) in order to support backward compatibility wherever applicable.

7. Security Considerations

Three aspects need to be considered by server side implementors:

1. There is no security at the TCP and HTTP protocol level (unless tunneled via SSL/TLS) and thus TCP-message should not be used to change state of the transaction. Change of state should be done on the signed PKIMessage being carried within the TCP-message.

2. If the server is going to be sending messages with sensitive information (not meant for public consumption) in the clear, it is RECOMMENDED that the server send back the message directly and not use the pollRep.

3. The polling request/response mechanism can be used for all kinds of denial of service attacks. It is RECOMMENDED that the server not change the polling-reference between polling requests.

Normative References

[CMP] Adams, C., Farrell, S., Kause, T. and T. Mononen, "Internet X.509 Public Key Infrastructure -- Certificate Management Protocol (CMP)", RFC 2510bis, January 0000.

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

[RFC2482] Whistler, K. and G. Adams, "Language Tagging in Unicode Plain Text", RFC 2482, January 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

Informative References

[RFC821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

Authors' Addresses

Amit Kapoor
Certicom
25801 Industrial Blvd
Hayward, CA
US

EMail: amit@trustpoint.com

Ronald Tschalar
Certicom
25801 Industrial Blvd
Hayward, CA
US

EMail: ronald@trustpoint.com

Tomi Kause
SSH Communications Security Corp.
Fredrikinkatu 42
Helsinki 00100
Finland

EMail: toka@ssh.com

Appendix A. Acknowledgments

The authors gratefully acknowledge the contributions of various members of the IETF PKIX Working Group and the ICSA CA-talk mailing list (a list solely devoted to discussing CMP interoperability efforts).

Appendix B. Registration of MIME Type for E-Mail or HTTP Use

To: ietf-types@iana.org
Subject: Registration of MIME media type application/pkixcmp

MIME media type name: application

MIME subtype name: pkixcmp

Required parameters: -

Optional parameters: -

Encoding considerations:

Content may contain arbitrary octet values (the ASN.1 DER encoding of a PKI message, as defined in the IETF PKIX Working Group specifications). base64 encoding is required for MIME e-mail; no encoding is necessary for HTTP.

Security considerations:

This MIME type may be used to transport Public-Key Infrastructure (PKI) messages between PKI entities. These messages are defined by the IETF PKIX Working Group and are used to establish and maintain an Internet X.509 PKI. There is no requirement for specific security mechanisms to be applied at this level if the PKI messages themselves are protected as defined in the PKIX specifications.

Interoperability considerations: -

Published specification: this document

Applications which use this media type: Applications using certificate management, operational, or ancillary protocols (as defined by the IETF PKIX Working Group) to send PKI messages via E-Mail or HTTP.

Additional information:

  Magic number (s): -
  File extension (s): ".PKI"
  Macintosh File Type Code (s): -

Person and email address to contact for further information:
Tomi Kause, toka@ssh.com

Intended usage: COMMON

Author/Change controller: Tomi Kause

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.