M.I.T. Laboratory for Computer Science

Request for Comments No. 240
February 16, 1983

Authentication Server Protocol

by Shawn A. Routhier

1. Introduction

This paper describes the interface to an authentication server based on the type proposed by Needham with two modifications. First there will be multiple servers on the net, none of which are guarenteed to be trusted by everybody or even know everybody. Of these servers some subset will be chosen to communicate the key. And second the initiator of the conversation will decide what the conversation key should be. The server protocol is based on one proposed in [Daniels]. The server is primarily intended to provide a key distribution service for conversations. It will also provide facilities for securing mail and writing digital signatures.

The protocol for establishing a conversation link, using conversation key Ck, between an initiator A and recipient B with keys Ax and Bx on server X will be as follows. Note that $(B)^{Ax}$ indicates encrypting B with key Ax.

Step one — A talks to B in the clear and they decide on using some mutually agreeable set of servers (y). The size of the set is also up to A and B. A then creates and breaks Ck into the correct number of pieces, Cx being the piece being sent via server X.

Step two — A sends an encoding request to each server of the set y. This request contains: {nonce, A, $(B, Cx)^{Ax}$}.

Step three — Each server returns an encoding reply to A. Each of these replys contain: {nonce, $(B, Cx, (A, Cx, \text{Time Stamp})^{Bx})^{Ax}$}.

Step four — A decodes each of the replies he gets and verifies that each has the right name and key part. The verification is to assure the correctness of the information recieved by the servers. He now has packets of the form: $\{(A, Cx, Time Stamp)^{Bx}\}$. He may now cache these for future use as well as sending them to B. Each packet he sends to B will have the name of the server that created that packet so that B knows which key to use to decode them. They contain $\{X, (A, Cx, Time Stamp)^{Bx}\}$.

Step five — B, after receiving and decoding the packets, puts the key together. At this stage B decides whether or not to trust the key. To trust the key B only has to trust one of the servers, not necessarily the same one as A, and the protocol. If he does trust it he can start a conversation using it and some protocol such as [kenta]. If he doesn't trust it he can try again or not as he sees fit.

The protocol for mail is similar to that for establishing a conversation link. The differences arise from the absence of direct interaction between A and B. This lack of interaction causes steps one, four and five to become:

Step one — To determine which servers to use A sends a query request {nonce, A, B} to various servers. Each server sends a query reply $\{nonce, (yes\ or\ no, B)^{Ax}\}$ back. A decodes these replys and is then able to pick a group of servers that know B (though B may or may not trust them).

Step four — A again verifys replys from steps two and three and attaches the servers name to the packet. This time though, instead of sending the packet directly to B, A ties the packets to a mail file encrypted with Ck. A then sends the entire file packet combination to B.

Step five — B gets the piece of mail, decodes the packets and builds the Ck. B may not trust the key as there is no guarentee that A picked servers that B trusted, but B is able to read the file and decide what action to take because of it.

Steps two and three remain unchanged. If A is using cached keys steps one, two, three, and four are changed to reflect that fact.

The protocol for writing signatures is:

Step one — A determines a suitable subset of servers as in mail protocol step one.

Step two — He then determines the characteristic value of the document to be signed and breaks this number up into the correct number of parts (one for each server).

Step three — A sends a Wsig request to each server in the subset containing: {nonce, A, (data, Time Stamp)^{Ax}}.

Step four — Each server sends a Wsig reply to A containg {nonce, (A, Data, Time Stamp)^{X}}

Step five — A takes these packets removes the nonce, adds the name of the server: {X, (A, Data, Time Stamp)^{X}} and puts these at the end of the signed document.

The protocol for reading signatures is:

Step one — B sends each of the servers in the subset a Rsig request with the correct signature packet: {nonce, B, (A, data, Time Stamp)^{X}}.

Step two — Each server returns a Rsig reply containing {nonce, (A, data, Time Stamp)^{Bx}}.

Step Three — B puts the data pieces together and compares the result to the characteristic value.

The time stamp is included to prevent somebody from creating a signature for some document from signature packets from other documents. The document should bear the same time stamp.

2. Encryption

The method of encryption will be to use DES [fips] in cipher block chaining mode [kentb]. The first 8 byte block sent will be a random number. This block will not be decoded but will only be used to start the cipher block chaining cycle. The encoded blocks will be multiples of 8 bytes long with the last 4 bytes of the message being a checksum. If padding is needed it will be placed between the data and the checksum. The checksum will be computed by adding the message together in 4 byte blocks using one's complement addition. Any padding in the message will be included in the checksum.

3. Time Stamp

Needham and Schroeder propose using a cached response from the server to initiate the conversation. Since DES keys can be found using a brute force search one may not want to have a conversation using an older key. For this reason a time stamp has been included in the protocol. Note that as Needham and Schroeder wanted this time stamp is not universal, it is merely to show whether the key is to old for use.

4. Datagrams

The maximum length of datagrams is implementation dependent, but the internet protocol [postel3] that the user datagram protocol [postel1] calls on suggests a limit of 576 bytes. The internet protocol requires 20 to 60 bytes of header and the user datagram protocol requires 8 more bytes of header so the maximum practical is 508 to 548 bytes. If names are in exess of 200 bytes each then the server may not be able to fit a response into one datagram. In this case the server returns an error message. The general style of the datagrams used by the server is a one byte request/reply code, followed by several items. Each item consists of a one byte item code followed by a one or two byte item length followed by the data itself. The item length includes the item code and the bytes used for item length.

4.1. Key Distribution

4.1.1. Request to the Authentication Server

```
+--------+
| Enc    |
| Request|
|        |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |        Initiator's Name           |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Block One                  |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+

Where:
```

Enc Request — is a one byte request code indicating that this is an encoding request (=1).

Name — is a one byte item code indicating that this item is a name (=1).

Length — is a one byte binary number giving the length of the item in bytes. It includes the item code and item length.

Initiator's name — is a string of ASCII characters.

Non — is a one byte item code indicating that this item is a nonce (=8).

Nonce — is a one byte binary number used to help tell messages apart.

Enc Block — is a one byte item code indicating that the following is an encrypted block (=254).

Llength — is a two byte binary number giving the length of the item in bytes. The item length and item code are included.

Block One — is a block of items encrypted with the initiator's key. This block contains:

```
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |          Recipient's Name         |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |                 |                 |
| Key    | Length | No. of Parts    | Keypart         |
|        |        |                 |                 |
+--------+--------+--------+--------+--------+---//---+
```

Where:

Name — is a one byte item code that indicates that this item is a name (=1).

Length — is a one byte binary number giving the length of the item in bytes. It includes the item code and item length.

Recipient's name — is a string of ASCII characters.

Key — is a one byte item code indicating that this item is a keypart (=5).

No. of Parts — is a one byte number giving the number of sections the key was broken into.

Keypart — is one part of the key. It is eight bytes long and when it is xored with the other keyparts it becomes a DES key.

4.1.2. Reply from the Authentication Server

```
+--------+
| Enc    |
| Reply  |
|        |
+--------+--------+--------+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|         Block Two                 |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
```

Where:

Non	is a one byte item code indicating that this item is a nonce (=8).
Length	is a one byte binary number giving the length of the item in bytes. The item code and item length are included.
Nonce	is a one byte binary number used to help tell messages apart.
Enc reply	is a one byte reply code indicating that this is a encoding reply (=2).
Enc Block	is a one byte item code indicating that the following is an encoded item (=254).
Llength	is a two byte binary number giving the length of the item in bytes. The item length and item code are included.
Block Two	is a block of items encrypted with the initiator's secret key. This block contains:

```
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |        Recipient's Name           |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |                 |                 |
| Key    | Length | No. of Parts    | Keypart         |
|        |        |                 |                 |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Block Three                |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
```

Where:

Name is a one byte item code indicating that this item is a name (=1).

Length is a one byte binary number giving the length of the item in bytes. It includes the item code and item length.

Recipient's Name is a string of ASCII characters.

Key is a one byte item code indicating that this item is a keypart (=5).

No. of parts is a one byte binary number telling how many keyparts there are.

Keypart is one part of the key. It is eight bytes long and when combined with the other keyparts it becomes a DES key. It is the same number as in the request.

Enc Block is a one byte item code indicating that the following item is encoded (=254).

Llength is a two byte binary number giving the length of the item in bytes. It includes the item code and item length.

Block Three is a block of items encrypted with the key that the authentication server had stored under the recipients name. This block contains:

```
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |        Initiator's Name           |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |                 |                 |
| Key    | Length | No. of Parts    | Keypart         |
|        |        |                 |                 |
+--------+--------+--------+--------+--------+---//---+
| Time   |        |                                   |
| Stamp  | Length |        The Time Stamp             |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
```

Where:

Name is a one byte item code indicating that this item is a name (=1).

Length is a one byte binary number giving the length of the item in bytes. This includes the item code and item length.

Initiator's Name is a string of ASCII characters.

Key — is a one byte itme code indicating that this item is a keypart (=5).

No. of Parts — is a one byte binary number telling how many keyparts there are.

Keypart — is an eight byte number that when combined with the other keyparts yields a DES key. This is identical to the one given by the initiator.

Time Stamp — is a one byte item code indicating that this item is a time stamp.

The Time Stamp — is a four byte binary number.

4.2. Querying the Authentication Server

4.2.1. Query Requests

To ask the server if it knows some name the following query request is sent. Where recipient's name is the one being checked.

```
+--------+
| Query  |
| Request|
|        |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |          Initiator's Name         |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |          Recipient's Name         |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+

Where:
```

Query Request — is a one byte request code indicating that this is a query request (=8).

Name — is a one byte item code indicating that this item is a name (=1).

Length — is a one byte binary number giving the length of the item in bytes. The item code and item length are included.

Initiator's Name — is a string of ASCII characters.

Recipient's Name is a string of ASCII characters.

Non is a one byte item code indicating that this item is a nonce (=8).

Nonce is a one byte binary number used to help tell messages apart.

4.2.2. Query Replys

```
+--------|
| Query  |
| Reply  |
|        |
+--------+--------+--------+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Block One                  |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+

Where:
```

Query Reply is a one byte reply code indicating that this is a reply to a query (=9).

Non is a one byte item code indicating that this item is a nonce (=8).

Length is a one byte binary number giving the length of the item in bytes. The item code and item length are included.

Nonce is a one byte binary number used to help tell messages apart.

Enc Block is a one byte item code indicating that the following block is an encrypted block (=254).

Llength is a two byte binary number giving the length of the item in bytes. The item length and item code are included.

Block One is a block of items encrypted with the key the server had stored under the initiator's name. This block contains:

```
+--------+--------+--------+
|        |        |        |
| Ans    | Length | Answer |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |         Recipient's Name          |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
```

Where:

Ans	is a one byte item code indicating that this item is an answer (=7).
Length	is a one byte binary number giving the length of the item in bytes. The item code and item length are included.
Answer	is a one byte binary number telling wheter the server knows the recipient. A yes is 255 and a no is 0.
Name	is a one byte item code indicating that this item is a name (=1).
Recipient's Name	is a string of ASCII characters.

4.3. Digital Signatures

4.3.1. Requesting a Digital Signature

The request for a digital signature is as follows.

```
+--------+
| WSig   |
| Request|
|        |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |        Requestor's Name           |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Request Block              |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
```

Where:

WSig Request — is a one byte request code indicating that this is a request write a signature block (=3).

Name — is a one byte item code indicating that this item is a name (=1).

Length — is a one byte binary number giving the length of the item in bytes. The item code and item length are included.

Requestor's Name — is a string of ASCII characters.

Non — is a one byte item code indicating that this item is a nonce (=8).

Nonce — is a one byte binary number used to help tell messages apart.

Enc Block — is a one byte item code indicating that this item is an encrypted block (=254).

Llength — is a two byte binary number giving the length of the item in bytes. The item code and item llength are included.

Request Block — is a block containing a single item, padded to length 0 mod 8, and encrypted with the key which the server has stored under the requestor's name. This block contains:

```
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Bytes  | Llength|        Request Data               |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
| Padding to length 0 mod 8|
| variable in length       |
|                          |
+--------+--------+--------+

Where:
```

Bytes — is a one byte item code indicating that this item is an uniterpreted byte string (=253).

Llength — is a two byte binary number giving the length of the item in bytes. The item code and item llength are included.

Request Data — is a block of data, variable in length which the server does not interpret. The data should be the "characteristic" value of the digital signature.

4.3.2. Signature Reply

```
+--------+
| WSig   |
| Reply  |
|        |
+--------+--------+--------+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Sig Block                  |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+

Where:
```

WSig Reply — is a one byte reply code indicating that this is a reply to a request for a signature block (=4).

Non — is a one byte item code indicating that this item is a nonce (=8).

Length — is a one byte binary number giving the length of the item in bytes. The item code and item length are included.

Nonce — is a one byte binary number used to help tell messages apart.

Enc Block — is a one byte item code indicating that this is an encrypted block (=254).

Llength — is a two byte binary number giving the length of the item in bytes. The item code and item length are included.

Sig Block — is a block of item, padded to length 0 mod 8, and encrypted with the secret key of the server. This block contains:

```
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Name   | Length |        Requestor's Name           |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                   |
| Bytes  | Llength|        Request Data               |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+
| Padding to length 0 mod 8|
| variable in length       |
|                          |
+--------+--------+--------+
```

Where:

Name — is a one byte item code indicating that this item is a name (=1).

Length — is a one byte binary number giving the length of the item in bytes. The item code and item length are included.

Requestor's Name — is the string of ASCII characters from the requestor's name field of the request to the server.

Bytes — is a one byte item code indicating that this item is an uninterpreted byte string (=253).

Request Data — is the request data from the request to the server.

4.3.3. Request to Read a Signature

To read a digital signature the following datagram should be sent to the server:

```
+--------|
| RSig   |
| Request|
|        |
+--------+--------+--------+--------+--------+---//---+
|        |        |                                    |
| Name   | Length |        Reader's Name               |
|        |        |                                    |
+--------+--------+--------+--------+--------+---//---+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                    |
| Block  | Llength|        Sig Block                   |
|        |        |                                    |
+--------+--------+--------+--------+--------+---//---+

Where:
```

RSig Request	is a one byte request type code indicating that this is a request to read a digital signature (=5).
Name	is a one byte item code indicating that this item is a name (=1).
Length	is a one byte binary number giving the length of the item in bytes. The item code and item length are included.
Reader's Name	is a string of ASCII characters.
Non	is a one byte item code indicating that this item is a nonce (=8).
Nonce	is a one byte binary number used to help tell messages apart.
Enc Block	is a one byte item code indicating that this item is an encrypted block (=254).
Llength	is a two byte binary number giving the item length in bytes. The item code and item length are included.
Sig Block	is the signature block, encrypted with the key of the server as defined above.

4.3.4. Authentication Server Reply

```
+--------+
| RSig   |
| Reply  |
|        |
+--------+--------+--------+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
| Enc    |        |                                   |
| Block  | Llength|        Sig Block                  |
|        |        |                                   |
+--------+--------+--------+--------+--------+---//---+

Where:
```

RSig Reply	is a one byte reply code indicating that this is a reply to a request to read a signature block (=6).
Non	is a one byte item code indicating that this item is a nonce (=8).
Length	is a one byte binary number giving the length of the item in bytes. The item code and item length are included.
Nonce	is a one byte binary number used to help tell messages apart.
Enc Block	is a one byte item code indicating that the following item is encrypted (=254).
Llength	is a two byte binary number giving the length of the item in bytes. The item code and item length are included.
Sig Block	is a block of items, padded to 0 mod 8, encrypted with the key the server had associated with reader's name. the contents are defined above.

4.4. Error Responses

```
+--------+
| Error  |
| Reply  |
|        |
+--------+--------+--------+
|        |        |        |
| Non    | Length | Nonce  |
|        |        |        |
+--------+--------+--------+--------+--------+---//---+
|        |        | Error  |                          |
| Error  | Length | Code   |       Error String       |
|        |        |        |                          |
+--------+--------+--------+--------+--------+---//---+
|                                                     |
| Additional items if required to explain error.      |
| (see error code descriptions below)                 |
+--------+--------+--------+--------+--------+---//---+

Where:
```

Error Reply — is a one byte reply code indicating that this is an error response (=7).

Non — is a one byte item code indicating that this item is a nonce (=8).

Length — is a one byte binary number giving the length of the item in bytes. It includes the item code and item length.

Nonce — is a one byte binary number used to help tell messages apart.

Error — is a one byte item code indicating that this is an error item (=3).

Error Code — is a one byte code stating the error. The codes are defined as:

Code	Meaning
0	Undetermined or undefined error.
1	Initiator's name not found (the unrecognized name follows).
2	Recipient's name not found (the unrecognized name follows).
3	Response larger than one datagram.
4	Key found under initiator's name doesn't work.

Error String — is a string of ASCII characters explaining the error.

5. Code Summary

Request/Reply Type Codes:

Type	Value
Enc Request	1
Enc reply	2
WSig Req	3
WSig Reply	4
RSig Req	5
RSig Reply	6
Error Rep	7
Query Req	8
Query Rep	9

Item Type Codes:

Type	Value	Length or Llength
Name	1	variable
Error	3	variable
Key	5	10
Time stamp	6	6
Ans	7	3
Non	8	3
Bytes	253	variable
Enc Block	254	variable

Error Codes:

Code	Meaning
0	Undetermined or undefined.
1	Initiator's name not found.
2	Recipient's name not found.
3	Response would not fit in one datagram.
4	Key stored under initiator's name doesn't work.

References

[Daniels 81]

Daniels, D., Lucassen, J., and Rubin, W.
Authentication Server Protocol.
Request for Comments 207, MIT Lab for Computer Science, May, 1981.

[FIPS ??]

Federal Information Processing Standards, Specifications for the Data Encryption Standard.
National Bureau of Standards, FIPS PUB 46, Jan, 1977.

[Kent 76]

Kent, S. T.
Encryption-Based Protection Protocols for Interactive User-Computer Communication.
Technical Report TR-162, MIT Lab for Computer Science, May, 1976.

[Kent 79]

Kent, S. T.
Protocol Design Considerations for Network Security.
In K. G. Beauchamp, editor, *Interlinking of Computer Networks.* D Reidel, Dordrecht, Holland, 1979.

[Postel 79]

Postel, J.
User Datagram Protocol.
Internet Experiment Note IEN 88, USC-Information Sciences Institute, May, 1979.

[Postel 80]

Postel, J.
DOD Standard Internet Protocol.
Internet Experiment Note IEN 128, USC-Information Sciences Institute, January, 1980.