

Network Working Group
INTERNET-DRAFT
Expires January 2002

J. Carlson
Sun Microsystems
B. Aboba
Microsoft Corporation
H. Haverinen
Nokia Mobile Phones
July 2001

EAP SRP-SHA1 Authentication Protocol
<[draft-ietf-pppext-eap-srp-03.txt](#)>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of [Section 10 of RFC 2026](#).

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

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>

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

This document is the product of the Point-to-Point Protocol Extensions Working Group of the Internet Engineering Task Force (IETF). Comments should be submitted to the ietf-ppp@merit.edu mailing list. Distribution of this memo is unlimited.

Abstract

The Extensible Authentication Protocol (EAP) [[RFC2284](#)] describes a framework that allows the use of multiple authentication mechanisms. This document defines an authentication mechanism for EAP called SRP-SHA1, based on the Secure Remote Password (SRP) [[RFC2945](#)] protocol.

EAP can be used with the Point-to-Point Protocol (PPP) [[RFC1661](#)] for link authentication.

Table of Contents

1.	Introduction	2
2.	Detailed Description of EAP SRP-SHA1 Authentication	3
2.1.	EAP SRP-SHA1 Packet Formats	4
2.2.	EAP SRP-SHA1 Challenge Subtype-Data	6
2.3.	EAP SRP-SHA1 Client Key Subtype-Data	8
2.4.	EAP SRP-SHA1 Server Key Subtype-Data	8
2.5.	EAP SRP-SHA1 Client Validator Subtype-Data	9
2.6.	EAP SRP-SHA1 Server Validator Subtype-Data	10
2.7.	EAP SRP-SHA1 Subtype 3 Response Packet	12
2.8.	EAP SRP-SHA1 Lightweight Challenge Subtype-Data	12
2.9.	EAP SRP-SHA1 Lightweight Response Subtype-Data	13
3.	Identity Privacy Support	13
3.1	Step One	14
3.2	Step Two	15
3.3	Using the Pseudonym	15
4.	Use with ECP	16
4.1.	One-Way Versus Bidirectional Encryption	17
4.2.	One-Way Versus Mutual Authentication	17
4.3.	DESEbis Versus 3DES	18
5.	Use with AAA	18
6.	Examples	19
6.1.	Successful Authentication	19
6.2.	Client Unauthenticated	19
6.3.	Server Unverified	20
7.	Security Considerations	21
8.	Intellectual Property Rights Notices	21
8.1.	Patent Issues	21
8.2.	Full Copyright Statement	22
9.	References	23
9.1.	Normative References	23
9.2.	Informative References	23
10.	Acknowledgements	24
11.	Authors' Addresses	24
12.	Appendix A - Well-Known Prime Modulus	25

[1.](#) Introduction

EAP allows the use of multiple authentication mechanisms within a single negotiation framework. When used with PPP, this protocol overcomes the chief design limitation of the original PPP authentica-

tion mechanisms, PAP [[RFC1334](#)] and CHAP [[RFC1994](#)], which required that the protocol to be used for authentication be chosen before the peer's identity was known. EAP also simplifies the process of adding new authentication mechanisms and linking them into external authentication services.

SRP is an open source protocol that provides strong, password-based authentication based on cryptographic hashing. Unlike PAP, the password never appears on the wire. Unlike CHAP (and variants MS-CHAPv1 [[RFC2433](#)] and MS-CHAPv2 [[RFC2759](#)]), access to a cleartext password is not required for the authenticator. Unlike all of these authentication protocols, SRP is resistant to dictionary attacks against the over-the-wire information. SRP is also resistant to eavesdropping and active attacks. As a side-effect, SRP also creates a session key that is resistant to man-in-the-middle attacks and can be used for data encryption.

SHA-1 [[FIPS180](#)] is a message digest algorithm that can be used as a hashing mechanism for SRP, producing an SRP variant known as SRP-SHA1. For calculation of the shared key in SRP, SHA-1 is used in an interleaved form to produce 40 octet hashes. See [section 3.1 of \[RFC2945\]](#) for details.

This document describes the message exchanges necessary for one peer to authenticate the other using EAP and SRP-SHA1. When used with PPP, the peers are independent and equal, and either or both may demand authentication from the other, and different protocols MAY be used independently in each direction.

When the PPP Encryption Control Protocol (ECP) [[RFC1968](#)] is used along with EAP SRP-SHA1, this document describes methods that SHOULD be used to generate the necessary shared keys from the SRP-SHA1 session key. Because authentication necessarily requires prior arrangement between the peers, pre-configured keys MAY be used in place of the SRP-SHA1 session key, and any such selection is outside the scope of this document.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [BCP 14 \[RFC2119\]](#).

[2.](#) Detailed Description of EAP SRP-SHA1 Authentication

Unlike CHAP, SRP-SHA1 requires more than one exchange to authenticate the peer. For this reason, three primary message subtypes are defined.

With SRP, the authenticator must communicate at least three values to the authenticatee, referred to as 's' (the password salt), 'B' (an ephemeral public key), and 'M2' (a hash value). The authenticatee must communicate at least three values to the authenticator, referred to as 'C' (the client name), 'A' (an ephemeral public key), and 'M1' (a hash value).

(The value 'u' passed from authenticator to authenticatee in the general SRP algorithm is derived from the first 32 bits of a SHA-1 hash of the 'B' value itself in the SRP-SHA1 algorithm. Thus, it does not need to be sent explicitly.)

In order to send its last value (M1), the authenticatee needs A, B, and u. However, in order to guarantee that the authenticatee's value chosen for A isn't a rogue value (see [section 3.2.4](#) of the SRP description [[SRP](#)]), the value of u must not be sent until the authenticatee reveals A. This implies that there are at least three steps -- authenticatee sends A, authenticator sends B, authenticatee sends M1.

The adaptation described in this document recommends the use of the EAP Identity Request/Response mechanism to obtain C from the peer. It then uses a new message type, called EAP SRP-SHA1, with three main subtypes to handle the SRP authentication. The Subtype 1 Request ("Challenge") message contains s, g, and N. The Subtype 1 Response ("Client Key") contains A. The Subtype 2 Request ("Server Key") continues with B and the Subtype 2 Response ("Client Validator") contains M1. Finally, the Subtype 3 Request ("Server Validator") contains the M2 value and the Subtype 3 response is an acknowledgement containing only the Subtype number and no data.

A fourth subtype is used for optional lightweight rechallenges.

[2.1.](#) EAP SRP-SHA1 Packet Formats

All EAP SRP-SHA1 authentication is driven by the authenticator (server). The authenticatee (client) MUST NOT retransmit Response messages, but SHOULD terminate the link if a Request is not received within a reasonable time period. The authenticator SHOULD silently ignore unexpected Response messages. The authenticatee MUST respond using EAP Nak if any unknown Subtype or otherwise unacceptable message is received.

Rationale:

Although the EAP Nak message is intended to signal only that a given EAP Type is unknown to the authenticatee and that a different Type should be used, its use here is still consistent with that meaning. If the authenticator attempts to use a subtype unknown to the authenticatee, then authentication using EAP SRP-SHA1 is itself not possible, and another Type should be tried.

A summary of the EAP SRP-SHA1 Request/Response packet format is shown

below. The fields are transmitted from left to right.

0								1								2								3			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1						
Code								Identifier								Length											
Type								Subtype								Subtype-Data ...											

Code

- 1 - Request
- 2 - Response

Identifier

The Identifier field is one octet and aids in matching responses with requests. The Identifier MUST be changed for each new Request message sent and MUST NOT be changed on retransmission of a given message. The Identifier in the Response message MUST match the corresponding Request. The authenticator MUST discard

non-matching Response messages.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, Subtype, and Subtype-Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception. Truncated packets (with Length greater than the link layer reported length) MUST be silently discarded.

Type

19 - EAP SRP-SHA1

Subtype

- 1 - Challenge / Client Key
- 2 - Server Key / Client Validator
- 3 - Server Validator
- 4 - Lightweight Rechallenge

The Subtype field is one octet and must contain the value 1, 2, 3, or 4. If any other subtype value is encountered in an EAP SRP-SHA1 Request message, then the authenticatee SHOULD return an EAP Response with the Type field set to Nak. In EAP SRP-SHA1

Response messages, the Subtype value must be copied from the corresponding Request message. The authenticator should treat unknown Subtype values in Response messages as malformed packets and silently discard.

Subtype-Data

The format of the Subtype-Data field is determined by the Code and Subtype fields as described in the sections below.

[2.2.](#) EAP SRP-SHA1 Challenge Subtype-Data

The EAP SRP-SHA1 Challenge (Subtype 1 Request) packet MUST be sent after the peer's identity has been obtained; use of the EAP Identity

Request/Response packet as described in [[RFC2284](#)] is RECOMMENDED. Using the peer's unauthenticated identity, the authenticator MUST look up the password salt, verifier ('v'), prime modulus ('N'), and generator ('g') values in an implementation-dependent manner. Use of EAP Identity is not required by this specification, and determination of salt, verifier, prime modulus, and generator MAY be done in any convenient implementation-dependent manner.

The authenticatee MUST validate that the specified generator value is indeed a generator modulo N, as described in the SRP documentation [[SRP](#)]. In many cases, an efficient method to do this is to keep a list of known-good values, and compare against this list. Since the full validation procedure is complex, the authenticator SHOULD use a longer timeout value if the default generator and modulus are not used; at least 30 seconds is RECOMMENDED.

A summary of the EAP SRP-SHA1 Challenge Subtype-Data format is shown below. Code (1), Identifier, Length, Type (19), and Subtype (1) fields are as described above. The fields are transmitted from left to right and are not padded or justified.

```

      0              1              2              3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Name Length | Server Name ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Salt Length | Salt ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Gen Length  | Generator ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Prime Modulus ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Name Length

A single octet giving the length of the Server Name field in octets. The server name identifies the authenticator, and is used by the authenticatee in an implementation-dependent manner. It MAY be used by the authenticatee to select an appropriate secret for authentication or displayed to a user.

Server Name

The authenticator's name. This field is not authenticated. It SHOULD NOT be used by the authenticatee as an authenticated peer name.

Salt Length

A single octet giving the length of the Salt field in octets. This MUST be at least 4 octets and MAY be any length up to 255 octets.

Salt

Password salt string; may contain any values. The contents of this field correspond to 's' in the SRP documentation.

Gen Length

A single octet giving the length of the Generator field in octets. This value MAY be zero to select the default generator value of 2.

Generator

The Generator value, called 'g' in the SRP documentation, is in network byte order. If the Gen Length field is zero, then the Generator field is omitted, and g is set to 2.

Prime Modulus

The Prime Modulus value, called 'N' in the SRP documentation, is in network byte order and fills the rest of the message to the length specified by the Length field in the EAP header.

This value MAY be omitted to select the 2048 bit value for N listed in [Appendix A](#). In this case, the Generator value MUST NOT be present in the message in order to default to 2.

If the Prime Modulus Length field is present, then it SHOULD be

2.3. EAP SRP-SHA1 Client Key Subtype-Data

The EAP SRP-SHA1 Client Key (Subtype 1) Response message MUST be sent in reply to an EAP SRP-SHA1 Subtype 1 Request message.

A summary of the EAP SRP-SHA1 Client Key Subtype-Data format is shown below. Code (2), Identifier, Length, Type (19), and Subtype (1) fields are as described above. The fields are transmitted from left to right.

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Value A ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Value A

The result of $(g^a) \bmod N$, where 'a' is a randomly chosen number in the range 1 .. N (exclusive). The randomly chosen number is the authenticatee's private key, and the Value A field is the corresponding public key. This field is in network byte order and is not padded.

The A value MUST NOT be zero modulo N. If the authenticator receives a bad value for this field, it MUST take action to disconnect or disable the link.

2.4. EAP SRP-SHA1 Server Key Subtype-Data

The EAP SRP-SHA1 Server Key (Subtype 2 Request) message MUST be sent by the authenticator after reception of a valid EAP SRP-SHA1 Client Key (Subtype 1) Response message.

A summary of the EAP SRP-SHA1 Server Key Subtype-Data format is shown below. Code (1), Identifier, Length, Type (19), and Subtype (2) fields are as described above. The fields are transmitted from left to right.

```

      0                               1                               2                               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|   Value B ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
```

Value B

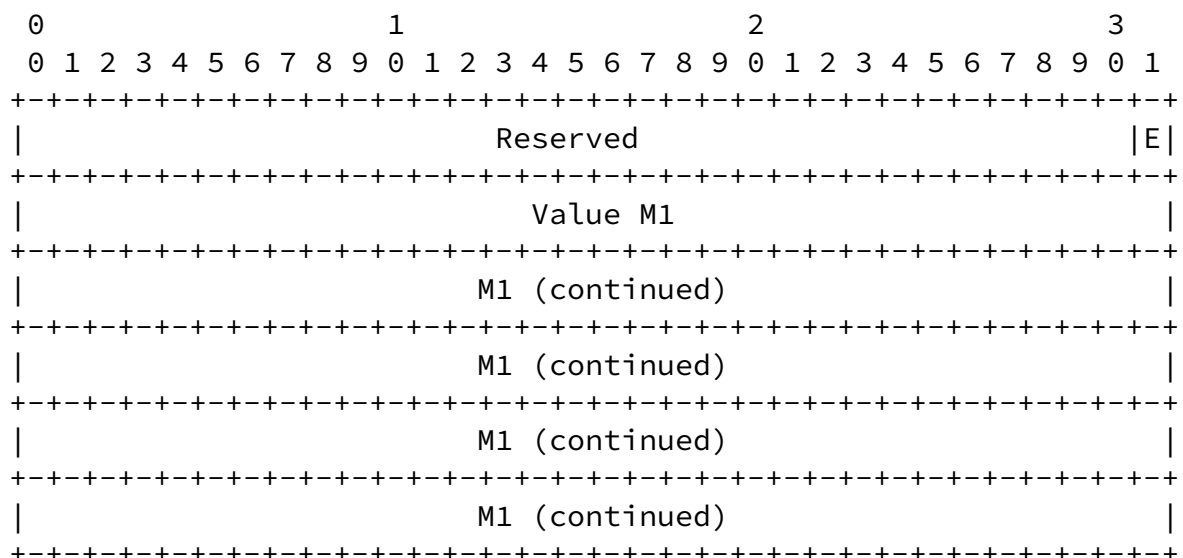
The result of $(v + g^b) \bmod N$, where 'b' is a randomly chosen number in the range 1 .. N (exclusive), and v is the stored verifier from the authentication database. The randomly chosen number is the authenticator's private key, and the Value B field is the corresponding public key. This field is in network byte order and is not padded.

The B value MUST NOT be zero modulo N. If the authenticatee receives a bad value for this field, it MUST take action to disconnect or disable the link.

2.5. EAP SRP-SHA1 Client Validator Subtype-Data

The EAP SRP-SHA1 Client Validator (Subtype 2 Response) message MUST be sent by the authenticatee in response to a valid EAP SRP-SHA1 Subtype 2 Request. The authenticator validates this Response message by calculating the M1 value from its own copies of A, B, and K, and compares the result. If the values match, then the authentication is successful, and EAP SRP-SHA1 Server Validator Request SHOULD be sent. If the values do not match, then EAP Failure SHOULD be returned and the link terminated.

A summary of the EAP SRP-SHA1 Client Validator Subtype-Data format is shown below. Code (2), Identifier, Length (30), Type (19), and Subtype (2) fields are as described above. The fields are transmitted from left to right.



Reserved

Must be zero on transmit by the authenticatee, ignored on reception by the authenticator.

E Bit

This bit is set if the authenticatee intends to use the derived session key (K) for ECP, as described in [section 4](#). If this bit is zero, then K will not be used, and if ECP is negotiated, it must use a different keying mechanism.

Value M1

The 20 octet result of SHA1(SHA1(N) xor SHA1(g), SHA1(clientname), s, A, B, K, id, Type). The single octet "id" value used in this calculation MUST be the EAP Identifier field from the EAP SRP-SHA1 Challenge (Subtype 1) Request message. The single octet "Type" value is a copy of the EAP Type field, which is fixed at 19 (decimal).

As described in SRP [[RFC2945](#)], the authenticatee computes $x = \text{SHA1}(s, \text{SHA1}(\text{clientname}, ":", \text{password}))$, and then computes $K = \text{SHA_Interleave}((B - g^x)^{(a + u \cdot x) \bmod N})$. The authenticator computes $K = \text{SHA_Interleave}((A * v^u)^b \bmod N)$. The calculated K values MAY be used with ECP (see [section 4](#)), lightweight rechallenge ([section 2.8](#)), and identity privacy ([section 3](#)). Finally, M1 is computed as SHA1(SHA1(N) xor SHA1(g), SHA1(clientname), s, A, B, K, id, Type). (Note that reference [[SRP](#)] gives different definitions of these values; the [[RFC2945](#)] document should be treated as the normative reference.)

The SHA1 hash to produce the long-term private key x, described above and in [[RFC2945](#)], SHOULD be used by default in EAP SRP-SHA1. As an implementation option, however, the x value used above MAY instead be derived from any mutually-chosen hashing algorithm, provided that the security of that algorithm is acceptable to both authentication parties. Note that such usage requires prior arrangement between the peers.

2.6. EAP SRP-SHA1 Server Validator Subtype-Data

The EAP SRP-SHA1 Server Validator (Subtype 3 Request) message MUST be sent by the authenticator after reception of a valid EAP SRP-SHA1 Client Validator (Subtype 2) Response. If the authenticator receives a Server Validator message with invalid contents, it MUST terminate the link.

Carlson, Aboba, Haverinen expires January 2002

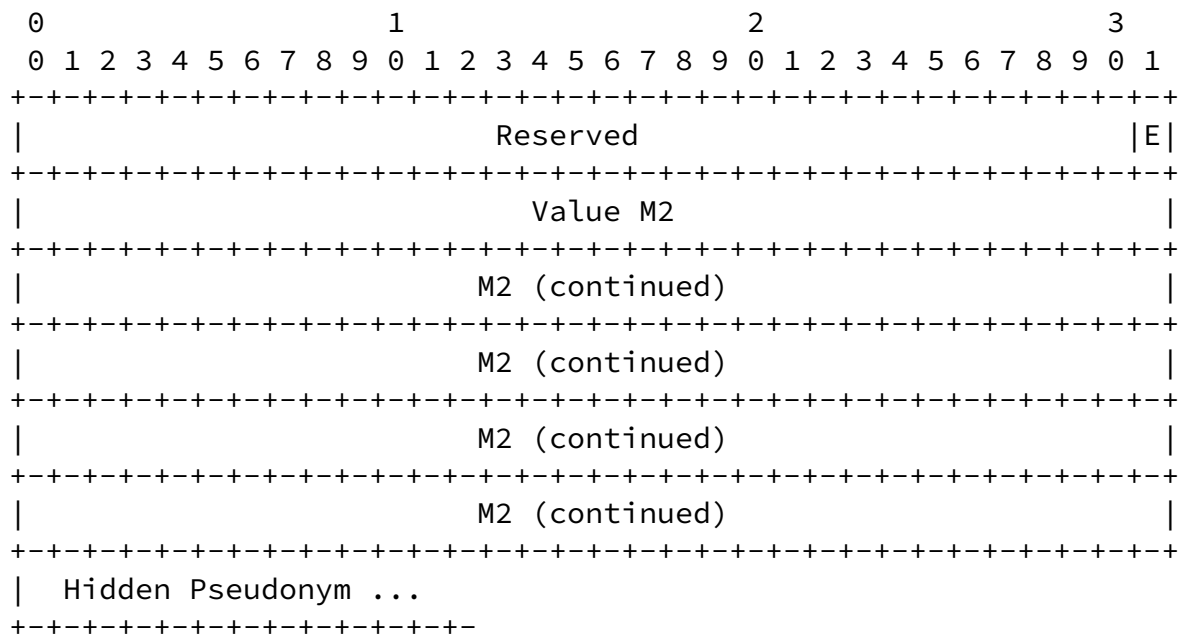
[Page 10]

INTERNET-DRAFT

EAP SRP-SHA1 Authentication Protocol

July 2001

A summary of the EAP SRP-SHA1 Server Validator Subtype-Data format is shown below. Code (1), Identifier, Length, Type (19), and Subtype (3) fields are as described above. The fields are transmitted from left to right.



Reserved

Must be zero on transmit by the authenticator, ignored on reception by the authenticatee.

E Bit

This bit is set if the authenticator intends to use the derived session key (K) for ECP, as described in [section 4](#). If this bit

is zero, then K will not be used, and if ECP is negotiated, it must use a different keying mechanism. This bit MUST be 0 if the authenticator uses a proxy authentication mechanism that does not provide access to the session key. (See [section 5](#).)

Value M2

The 20 octet result of SHA1(A, M1, K, id, Type). The single octet "id" value used in this calculation MUST be the EAP Identifier field from the EAP SRP-SHA1 Challenge (Subtype 1) Request message. The single octet "Type" value is a copy of the EAP Type field, which is fixed at 19 (decimal).

Hidden Pseudonym

This optional field is described in [section 3](#). The Hidden

Pseudonym field extends to the end of the message as indicated by the EAP Length field.

[2.7](#). EAP SRP-SHA1 Subtype 3 Response

The authenticatee MUST transmit a EAP SRP-SHA1 Subtype 3 Response message in reply to a valid Server Validator message from the peer. This Response message has no Subtype-Data field. The Code (2), Identifier, Length (6), Type (19), and Subtype (3) fields are as described above.

[2.8](#). EAP SRP-SHA1 Lightweight Challenge Subtype-Data

The EAP SRP-SHA1 Lightweight Challenge (Subtype 4 Request) message MAY be used by the authenticator for periodic rechallenges at any time after EAP authentication is complete.

After rechallenge with this mechanism, the shared session key remains unchanged. This property may be useful when this key is used with ECP, because regular reauthentication (starting with a new Subtype 1 Request) will change the key. This mechanism may also be useful with external security servers, because the NAS can implement this feature without making additional queries to the server if the server sup-

plies the K value.

A summary of the EAP SRP-SHA1 Lightweight Challenge Subtype-Data format is shown below. Code (1), Identifier, Length, Type (19), and Subtype (4) fields are as described above. The fields are transmitted from left to right.

```

      0               1               2               3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
| Challenge ...
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

Challenge

The Challenge value is a sequence of random data. This field MAY be any length up to the MTU, but MUST be at least four octets. The authenticatee MUST NOT reply if the Challenge field is too short.

[2.9.](#) EAP SRP-SHA1 Lightweight Response Subtype-Data

The EAP SRP-SHA1 Lightweight Response (Subtype 4 Response) message SHOULD be sent by the authenticatee in response to a Lightweight Challenge message. The authenticatee MAY instead use EAP Nak with Type-Data set to 19 (EAP SRP-SHA1) to restart full SRP authentication.

The authenticator MUST take action to disconnect or disable the session if the Lightweight Response value is invalid or if a preset number of Lightweight Challenge messages are sent without a valid response.

A summary of the EAP SRP-SHA1 Lightweight Response Subtype-Data format is shown below. Code (2), Identifier, Length (26), Type (19), and Subtype (4) fields are as described above. The fields are transmitted from left to right.

3.1.1. Method A

The server constructs a padded client name string by prepending the length of the client name in bytes to the client name, and pads to the next 8 octet boundary with random data (the "|" operator represents concatenation):

```
paddedname = length(clientname) | clientname | random{0..7}
```

A 56 bit key is then generated from a locally-stored password and the current date (to the nearest day) by extracting the first 56 bits from the result of SHA1(local-password, date-string). The paddedname is then encrypted using DES [[FIPS46](#)] with this key. The encrypted result is converted to a printable string using the BASE64 method described in [section 4.3.2.4 of \[RFC1421\]](#), but without the '=' padding. This string is the pseudonym used in Step Two below.

If the client name is an NAI and this is known to the server, then the realm SHOULD be removed to limit the string length. The realm will be supplied by the client when the pseudonym is used.

The advantages of this method are that the server need not store the generated pseudonym because it can always decrypt the value provided by the client, the generated pseudonym for a given client changes frequently because of the use of the date in the hash, and previous pseudonyms are always usable because prior dates can be tested as well. The disadvantage is that it requires the use of reversible encryption.

[3.1.2.](#) Method B

The server generates a random value (a nonce) and converts it to a printable string as in method A. The server MUST store a copy of this value in stable storage and relate it to the true client identity.

The advantages of this method are that the pseudonym changes with

every connection and DES is not required. The disadvantage is that stable storage is required.

[3.2.](#) Step Two

The pseudonym is communicated to the client using a hiding mechanism relying on the session key so that the client can undo the hiding. Because this operation must be reversible on the client side, the server **MUST** use the method described here (based on [[KPS](#)]) for this step. The length of the pseudonym from Step One is prepended as a single octet to the front of the pseudonym and random octets are appended to pad the data to the next 20 octet boundary:

```
value = length(pseudonym) | pseudonym | random{0..19}
```

The Hidden Pseudonym for the Server Validator message is then calculated by breaking the value above into a sequence of 20 octet segments ($v[1]$ through $v[n]$):

```
hpn[1] = SHA1(id, K, clientname) ^ v[1]
hpn[2] = SHA1(id, K, hpn[1]) ^ v[2]
hpn[3] = SHA1(id, K, hpn[2]) ^ v[3]
...
```

The "id" number is the EAP Identifier octet for Subtype 3 Request. The $hpn[1]$ through $hpn[n]$ values are then concatenated to form the Hidden Pseudonym. On reception of the Server Validator, the client undoes this hiding. It calculates:

```
v[1] = SHA1(id, K, clientname) ^ hpn[1]
v[2] = SHA1(id, K, hpn[1]) ^ hpn[2]
...
```

The client extracts the decoded pseudonym from the $v[1]$ through $v[n]$ sequence, and saves it in stable storage. The client **MUST** discard the pseudonym if the length octet is invalid; either greater than the remaining length of the unhidden value or 20 or more octets less than that length.

[3.3.](#) Using the Pseudonym

On the next connection to the server, the client **SHOULD** transmit the stored pseudonym in response to the first received EAP Identity request. In order to do this, it **MUST** concatenate three strings, as follows:

identity = "pseudo_" | pseudonym | realm

The first string is the constant 7 character string "pseudo_". This string allows the server to identify this name as a pseudonym. The second string is the stored pseudonym itself. The third string is the NAI separator ("@") and realm, if any, as specified by an administrator. The client's user interface SHOULD allow the user to specify the NAI user name and realm as separate values if the pseudonym feature is supported.

The server, on finding the peer name starting with "pseudo_", attempts to decode it in an implementation-dependent manner. If Method A was chosen in Step One, then this consists of generating DES keys with the current date as well as several prior dates, and attempting to decrypt with each of those keys, only one of which will result in a usable client name. If Method B was chosen, then the server looks up the pseudonym in the local storage to find the corresponding client.

If decoding to the padded client name fails or does not result in a known client name, then the server requests the regular (non-pseudonym) identity by resending the EAP Identity request with a new (changed) ID number. The client MUST distinguish between retransmissions of the EAP Identity request, which will have the same ID number, and for which the pseudonym MAY be sent, and a new EAP Identity request, which will have a different ID number, and for which the client's actual name MUST be sent. The server MUST NOT allow the use of a pseudonym in reply to its resent request, because the client is required to supply its actual name.

As an implementation option, the client may wish to support only obscured connections when possible. In this case, the client SHOULD have a boolean flag for "use private connections when possible." If the server does not offer a pseudonym, then the flag is ignored. If the server does offer a pseudonym, then the client MUST disconnect or disable the link if it has used its actual name for EAP Identity and reconnect after a random interval. This option SHOULD be disabled and an administrator notified if use of a pseudonym fails more than once, in order to prevent loss of functionality.

4. Use with ECP

The 40 octet shared key ('K') calculated by the SRP-SHA1 authentication procedure MUST be used to form encryption keys as described in this section if PPP encryption is in use and the E bits in both the Client Validator and the Server Validator messages were set.

For the DESEbis [[RFC2419](#)] algorithm, a shared 56 bit key is necessary, and for the 3DES [[RFC2420](#)] algorithm, a shared 168 bit key is required. Complicating this, ECP may be negotiated in only one direction or both. In addition, PPP authentication may be performed one-way (the most common case) or mutually, and when mutual authentication is chosen, different authentication schemes may be used to authenticate each peer. The effects of these details are described below.

The "decryptor" is the peer that sends ECP Configure-Request suggesting an algorithm and receives a corresponding ECP Configure-Ack. The "encryptor" is the sender of the ECP Configure-Ack, and will transmit the encrypted data.

Rekeying can be accomplished at any time by restarting EAP authentication. When rekeying, the decryptor SHOULD accept data encrypted with the prior key until the Subtype 3 Response is sent. The encryptor SHOULD suspend data transmission, if possible, when the Subtype 3 Request is sent and resume after Subtype 3 Response.

[4.1.](#) One-Way Versus Bidirectional Encryption

ECP may be negotiated in one direction on the link, or in both directions. For each separate ECP session, the K value that is chosen for the shared key should be the value associated with the EAP SRP-SHA1 authentication in which the decryptor is the authenticator. If the decryptor was not an EAP SRP-SHA1 authenticator, then the K value associated with the other EAP SRP-SHA1 authentication session is used instead as described in the next section.

[4.2.](#) One-Way Versus Mutual Authentication

If only one peer authenticates the other using SRP-SHA1, and the other either does not authenticate its peer at all or uses a method that does not result in encryption keys, then the one K value associated with this authentication MUST be used for both encryption sessions. The first 20 octets of K are used for the encryption of data sent by the authenticatee to the authenticator, and the second 20

octets of K are used for encryption of data in the opposite direction.

If mutual authentication with algorithms that produce encryption keys is done, such as SRP-SHA1, then two K values are produced. As described above, the K values are used so that the encryptor is the authenticatee for the corresponding authentication session, and the decryptor is the authenticator.

[4.3.](#) DESEbis Versus 3DES

For DESEbis, the first 8 octets of the key value are used. DES keys are constructed such that the most significant bit (MSB) of each octet is reserved for parity, and must be discarded.

For 3DES, 24 octets of key value are used by most implementations. As for DESEbis, the MSBs are discarded. If the 40 octet K value has been split into two 20 octet values because one-way authentication is in use, then an extra 4 octets are needed for each key. The SHA-1 algorithm is run again over the 40 octet K value to produce a 20 octet hash. This hash is split into two 10 octet values that are then appended to the two 20 octet values from the split K value. The first 24 octets of each 30 octet value is used as the 3DES key.

[5.](#) Use with AAA

The SRP exchange uses the client name as part of the calculation, and thus depends on leaving this string unmodified between the authenticatee and authenticator. If a mechanism, such as a AAA protocol, is used to perform the SRP authentication on behalf of a Network Access Server (NAS), then the client name MUST be identical on both ends. In particular, if a Network Access Identifier [[RFC2486](#)] is used with a proxy authentication server, then the implementor MUST take steps to preserve the client name string end-to-end.

Use of a AAA protocol also makes the use of the session key (K) for ECP keying problematic, because the contents of this key are available only at the authentication endpoints (where SRP is run), and not the link layer endpoints (where ECP is run).

For RADIUS [[RFC2138](#)], no common method exists to transfer the session

key to the NAS, but some implementations are known to have proprietary extensions for this purpose. If such an extension is available, it SHOULD be used to transfer the key and the Server Validator E bit MUST then be set to 1. Otherwise, if the extension is not available, then the E bit MUST be cleared to 0.

For other AAA protocols, encryption session key transfer SHOULD be used to send K to the NAS.

Use of a directory access protocol for the hash values, rather than a AAA protocol, would also solve this problem. Such usage is outside the scope of this document.

[6.](#) Examples

[6.1.](#) Successful Authentication

In the case where the EAP SRP-SHA1 authentication is successful, the conversation may appear as follows:

Authenticatee	Authenticator

	<- EAP-Request id=100 / Identity
EAP-Response id=100 / Identity ("MyName") ->	
	<- EAP-Request id=101 / SRP-SHA1 Subtype 1 ("TheServer", salt, generator, modulus)
EAP-Response id=101 / SRP-SHA1 Subtype 1 (A) ->	
	<- EAP-Request id=102 / SRP-SHA1 Subtype 2 (B)
EAP-Response id=102 / SRP-SHA1 Subtype 2 (E, M1) ->	
	<- EAP-Request id=103 / SRP-SHA1 Subtype 3 (E, M2, hidden-pseudonym)

```

EAP-Response id=103 /
SRP-SHA1 Subtype 3    ->
                        <- EAP-Success id=104

```

Note that M1 and M2 were calculated with "id" set to 101, the EAP Identifier field for the Subtype 1 Request. The id is shown as an integer that increments by 1 for each EAP Request, but this is not required. Any values MAY be used, provided that repetition is minimized.

6.2. Client Unauthenticated

In the case where the client fails to authenticate to the server, the conversation may appear as follows:

Authenticatee	Authenticator

	<- EAP-Request id=50 / Identity
EAP-Response id=50 / Identity ("MyName")	->

```

                        <- EAP-Request id=51 / SRP-SHA1
                           Subtype 1 ("TheServer", salt, generator,
                                       modulus)
EAP-Response id=51 /
SRP-SHA1 Subtype 1
(A)                ->
                        <- EAP-Request id=52 / SRP-SHA1
                           Subtype 2 (B)
EAP-Response id=52 /
SRP-SHA1 Subtype 2
(E, M1)            ->
                        <- EAP-Failure id=53

```

(At its option, the authenticator MAY choose to send a false Subtype 3 message with a random number in place of M2, followed by EAP Failure. Doing so limits the amount of information that an attacker has available.)

6.3. Server Unverified

In the case where the client's verification of the server is unsuccessful, the conversation will appear as follows:

Authenticatee	Authenticator

	<- EAP-Request id=75 / Identity
EAP-Response id=75 / Identity ("MyName") ->	
	<- EAP-Request id=76 / SRP-SHA1 Subtype 1 ("TheServer", salt, generator, modulus)
EAP-Response id=76 / SRP-SHA1 Subtype 1 (A) ->	
	<- EAP-Request id=77 / SRP-SHA1 Subtype 2 (B)
EAP-Response id=77 / SRP-SHA1 Subtype 2 (E, M1) ->	
	<- EAP-Request id=78 / SRP-SHA1 Subtype 3 (E, M2, hidden-pseudonym)
[Disconnect] ->	

(The "disconnect" operation is medium-dependent. On a PPP link, it consists of sending LCP Terminate-Request followed by sending a Close event to the physical layer.)

7. Security Considerations

EAP SRP-SHA1 is a security protocol.

The security of SRP on the wire relies on having a prime modulus that is large enough to make brute force attack of the key exchange infeasible. A length of at least 1024 bits is recommended. In addition, the SRP document [[RFC2945](#)] describes tests that MUST be performed on the chosen modulus and generator values and random numbers. SRP is a significant improvement over the situation with PAP, which has no on-the-wire security, and with CHAP, which is vulnerable to dictionary attacks against a captured Challenge/Response pair.

The security of the credentials stored in the authenticator's database relies on the entropy in the chosen password, the difficulty of hash inversion of a salted string, and the security of the system containing the database. For this reason, the chosen password SHOULD be restricted to limit the effectiveness of dictionary attacks against a disclosed database. However, since typical passwords have only a few bits of entropy, the database itself MUST be protected against disclosure.

The method used to hide the pseudonym has not been subjected to rigorous analysis, but is believed to be sufficient for the task. Because the outer layer of hiding must be decoded by the authenticatee, and interception of this information would link one session to another, it is not believed that stronger methods for the inner layer are useful. If strong identity privacy is a concern, this mechanism should not be used.

[8.](#) Intellectual Property Rights Notices

[8.1.](#) Patent Issues

The SRP key-exchange protocol described in this document is available worldwide on a royalty-free basis for commercial and non-commercial uses. This specifically includes simultaneous bidirectional use of SRP, which is distinct from SRP-Z. No usage of SRP-Z is described in this document.

"The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such

rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in [BCP-11](#). Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made

to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat."

"The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director."

[8.2.](#) Full Copyright Statement

"Copyright (C) The Internet Society 2001. All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

9. References

9.1. Normative References

- [RFC2284] L. Blunk, J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)," [RFC 2284](#), March 1998
- [RFC2945] T. Wu, "The SRP Authentication and Key Exchange System," [RFC 2945](#), September 2000
- [RFC1661] W. Simpson, "The Point-to-Point Protocol (PPP)," [RFC 1661](#), July 1994
- [FIPS180] National Institute of Standards and Technology (NIST), "Announcing the Secure Hash Standard," FIPS 180-1, U.S. Department of Commerce, April 1995

9.2. Informative References

- [RFC1334] B. Lloyd, W. Simpson, "PPP Authentication Protocols," [RFC 1334](#), October 1992
- [RFC1994] W. Simpson, "PPP Challenge Handshake Authentication Protocol (CHAP)," [RFC 1994](#), August 1996
- [RFC2433] G. Zorn, S. Cobb, "Microsoft PPP CHAP Extensions," [RFC 2433](#), October 1998
- [RFC2759] G. Zorn, "Microsoft PPP CHAP Extensions, Version 2," [RFC 2759](#), January 2000
- [RFC1968] G. Meyer, "The PPP Encryption Control Protocol (ECP)," [RFC 1968](#), June 1996
- [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels," [BCP 14](#) and [RFC 2119](#), March 1997
- [SRP] T. Wu, "The Secure Remote Password Protocol", Proceedings of the 1998 Internet Society Symposium on Network and Distributed Systems Security, San Diego, CA, pp. 97-111
- [RFC2486] B. Aboba, M. Beadles, "The Network Access Identifier," [RFC 2486](#), January 1999
- [FIPS46] National Bureau of Standards, "Data Encryption Standard", FIPS PUB 46, January 1977

- [RFC1421] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures," [RFC 1421](#), February 1993
- [KPS] C. Kaufman, R. Perlman, and M. Speciner, "Network Security: Private Communications in a Public World," Prentice Hall, ISBN 0-13-061466-1, March 1995
- [RFC2419] K. Sklower, G. Meyer, "The PPP DES Encryption Protocol, Version 2 (DESE-bis)," [RFC 2419](#), September 1998
- [RFC2420] H. Kummert, "The PPP Triple-DES Encryption Protocol (3DESE)," [RFC 2420](#), September 1998
- [RFC2138] C. Rigney, A. Rubens, W. Simpson, and S. Willens, "Remote Authentication Dial In User Service (RADIUS)," [RFC 2138](#), April 1997

[10.](#) Acknowledgements

The authors are grateful for the many critiques and ideas offered on the IETF PPP Extensions mailing list and by private mail. In particular, we thank N. Asokan, Jacques Caron, and Vernon Schryver.

The hiding method used for the pseudonym was adapted from RFCs 2661 and 2138, both of which were based on the "Mixing in the Plaintext" section in the book "Network Security" by Kaufman, Perlman and Speciner [[KPS](#)].

[11.](#) Authors' Addresses

James Carlson
Sun Microsystems
1 Network Drive MS UBUR02-212
Burlington MA 01803-2757
Email: james.d.carlson@sun.com
Phone: +1 781 442 2084
Fax: +1 781 442 1677

Bernard Aboba

Microsoft Corporation
One Microsoft Way
Redmond WA 98052
Email: bernarda@microsoft.com
Phone: +1 425 936 6605

Carlson, Aboba, Haverinen expires January 2002

[Page 24]

INTERNET-DRAFT EAP SRP-SHA1 Authentication Protocol

July 2001

Henry Haverinen
Nokia Mobile Phones
P.O. Box 88
FIN-33721 Tampere
Finland
Email: henry.haverinen@nokia.com
Phone: +358 50 594 4899
Fax: +358 3 318 3690

12. Appendix A - Well-Known Prime Modulus

This 2048 bit prime modulus (and corresponding generator value 2) are borrowed from the SRP GSS-API mechanism, an IETF work in progress.

0xAC, 0x6B, 0xDB, 0x41, 0x32, 0x4A, 0x9A, 0x9B,
0xF1, 0x66, 0xDE, 0x5E, 0x13, 0x89, 0x58, 0x2F,
0xAF, 0x72, 0xB6, 0x65, 0x19, 0x87, 0xEE, 0x07,
0xFC, 0x31, 0x92, 0x94, 0x3D, 0xB5, 0x60, 0x50,
0xA3, 0x73, 0x29, 0xCB, 0xB4, 0xA0, 0x99, 0xED,
0x81, 0x93, 0xE0, 0x75, 0x77, 0x67, 0xA1, 0x3D,
0xD5, 0x23, 0x12, 0xAB, 0x4B, 0x03, 0x31, 0x0D,
0xCD, 0x7F, 0x48, 0xA9, 0xDA, 0x04, 0xFD, 0x50,
0xE8, 0x08, 0x39, 0x69, 0xED, 0xB7, 0x67, 0xB0,
0xCF, 0x60, 0x95, 0x17, 0x9A, 0x16, 0x3A, 0xB3,
0x66, 0x1A, 0x05, 0xFB, 0xD5, 0xFA, 0xAA, 0xE8,
0x29, 0x18, 0xA9, 0x96, 0x2F, 0x0B, 0x93, 0xB8,
0x55, 0xF9, 0x79, 0x93, 0xEC, 0x97, 0x5E, 0xEA,
0xA8, 0x0D, 0x74, 0x0A, 0xDB, 0xF4, 0xFF, 0x74,
0x73, 0x59, 0xD0, 0x41, 0xD5, 0xC3, 0x3E, 0xA7,
0x1D, 0x28, 0x1E, 0x44, 0x6B, 0x14, 0x77, 0x3B,
0xCA, 0x97, 0xB4, 0x3A, 0x23, 0xFB, 0x80, 0x16,
0x76, 0xBD, 0x20, 0x7A, 0x43, 0x6C, 0x64, 0x81,
0xF1, 0xD2, 0xB9, 0x07, 0x87, 0x17, 0x46, 0x1A,
0x5B, 0x9D, 0x32, 0xE6, 0x88, 0xF8, 0x77, 0x48,

0x54, 0x45, 0x23, 0xB5, 0x24, 0xB0, 0xD5, 0x7D,
0x5E, 0xA7, 0x7A, 0x27, 0x75, 0xD2, 0xEC, 0xFA,
0x03, 0x2C, 0xFB, 0xDB, 0xF5, 0x2F, 0xB3, 0x78,
0x61, 0x60, 0x27, 0x90, 0x04, 0xE5, 0x7A, 0xE6,
0xAF, 0x87, 0x4E, 0x73, 0x03, 0xCE, 0x53, 0x29,
0x9C, 0xCC, 0x04, 0x1C, 0x7B, 0xC3, 0x08, 0xD8,
0x2A, 0x56, 0x98, 0xF3, 0xA8, 0xD0, 0xC3, 0x82,
0x71, 0xAE, 0x35, 0xF8, 0xE9, 0xDB, 0xFB, 0xB6,
0x94, 0xB5, 0xC8, 0x03, 0xD8, 0x9F, 0x7A, 0xE4,
0x35, 0xDE, 0x23, 0x6D, 0x52, 0x5F, 0x54, 0x75,
0x9B, 0x65, 0xE3, 0x72, 0xFC, 0xD6, 0x8E, 0xF2,
0x0F, 0xA7, 0x11, 0x1F, 0x9E, 0x4A, 0xFF, 0x73