Network Working Group D Harkins, D Piper
INTERNET-DRAFT Nokia
draft-harkins-ipsra-crack-00.txt August 25, 2000

IKE Challenge/Response for Authenticated Cryptographic Keys (Revised)
<draft-harkins-ipsra-crack-00.txt>

Status of this Memo

Table of Contents

1. Abstract

This memo describes a new IKE authentication method ([HC98]) which provides for mutual authentication when one side is using a legacy-based secret-key authentication technique such as RADIUS, SecurID, or OTP and the other side is using public-key authentication, with optional digital certificates.

The generic protocol described herein is an open-ended IKE phase 1 exchange ([HC98]). The result of this exchange is a mutually authenticated IKE security association ([HC98]). The keys that are derived from this SA are also authenticated and thereby convey this state to any SA's created from it for any other security service, such as IPsec [Pip98].

2. Terms and Definitions

2.1 Requirements Terminology and Notation

Keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT" and "MAY" that appear in this document are to be interpreted as described in [Bra96].

The notation of this memo is similar to [HC98]. Like [HC98] it uses payloads defined in [MSST98]. The notation for the new payload is:

CHRE is the newly defined "challenge/response payload"

To prevent confusion in the protocol diagrams (e.g. between the Diffie-Hellman public values), the client's payloads are sometimes post-fixed with "i", for "initiator", and the gateway's payloads are sometimes post-fixed with "r", for "responder".

2.2 IKE Exchange Integration

This protocol is motivated by mobile IPsec-enabled clients who desire to use legacy authentication techniques instead of digital certificates. Therefore the parties to this exchange are a "client" and a "gateway". The client is always the initiator of this exchange and is assumed to be coming from an IP address that cannot be known a priori by the gateway.

The protocol described in this memo is an IKE exchange using a newly defined IKE authentication method. All other attributes and their status from [HC98] are unaffected. Unless otherwise overridden by a specific requirement in this memo, all requirements in [HC98] exist in this memo.

2.3 IKE Authentication Method Definition

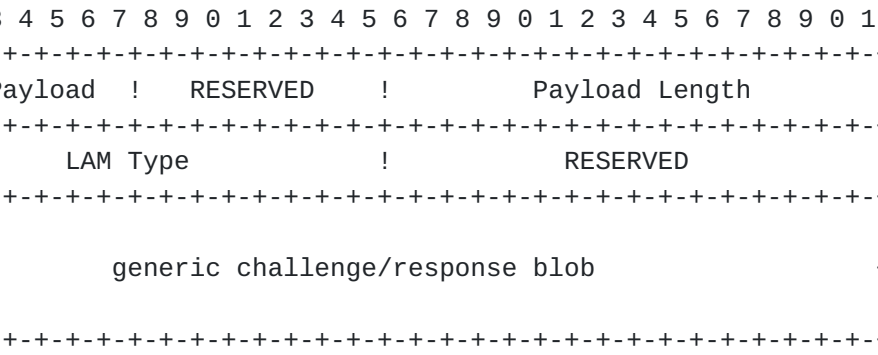
The following new IKE authentication method value is defined for CRACK from the IKE private-use space (see Section 6):

Authentication Mode	Value
IKE_A_CRACK	128

2.4 The Challenge/Response Payload (CHRE)

This draft requires a new payload to carry new information unique to this exchange. The Challenge/Response payload is used to convey a challenge from the gateway to the client and is used by the client to respond to a challenge from the gateway. The Challenge/Response payload contains attributes denoting specific information conveyed from the client to the gateway and back. The actual legacy authentication method will determine the contents of this payload at the various points in the exchange.

This payload consists of the ISAKMP generic header ([MSST98]) and a payload-specific body whose length is not fixed. The "Payload Length" in the generic header includes the length of the header itself. All fields labeled "RESERVED" MUST be filled with zero (0) prior to sending and each party to the exchange MUST verify that value on all payloads it is sent.



The payload type for this payload is 128, which is taken from the ISAKMP private use space (see Section 6). The body of this payload may also contain attributes used to convey authentication information (see Section 4.2).

The LAM Type field denotes the legacy authentication method (see Section 5) associated with the exchange. The LAM Type must be set in all CHRE payloads in an exchange. The LAM Type is selected by the initiator (client) and MUST be set in every CHRE payload to the same value throughout the exchange.

2.5 LAM Types

Different legacy authentication methods are denoted by unique LAM type identifiers in the Challenge/Response payloads. The legacy authentication methods defined for this protocol are as follows:

LAM Type Identifier	Value
CRACK_PASSWORD	1
CRACK_OTP	2
CRACK_CHALLENGE_RESPONSE	3
CRACK_SECURID	4
<reserved>	5-32767
<private use>	32768-65535

If the gateway is not configured to support the requested LAM type while processing the client's first CHRE payload, the gateway MUST terminate the exchange and MUST respond with an ISAKMP Notify (PROPOSAL-NOT-CHOSEN).

A conformant gateway MUST support at least one of the specified LAM Types. A gateway MAY support more than one LAM Type and it's assumed that the choice of which LAM Types are supported is implementation specific and determined from local policy configuration, perhaps on a per-user basis based on the content of the first CHRE payload and its associated attributes.

CRACK_PASSWORD specifies a simple username/password mechanism. It's used for any simple host-based password or one-way hash mechanism. It also useful for proxy-based password authentication schemes, like TACACS and RADIUS.

CRACK_OTP specifies that a one-time password mechanism. It's useful for the S/KEY [Hal95] and OTP [HM96] schemes.

CRACK_CHALLENGE_RESPONSE specifies a token-based challenge/response mechanism. It's useful for a wide variety of cryptographic tokens, typically based on DES.

CRACK_SECURID specifies a SecurID mechanism. It's useful for the RSA SecurID system. The CRACK_SECURID closely resembles CRACK_CHALLENGE_RESPONSE.

2.6 LAM Attributes

The Challenge/Response payload contains attributes used to convey information between the client and the gateway authenticating the client. These are standard [MSST98] attribute payloads associated with the Challenge/Response payloads. The following LAM attributes are valid:

Attribute	Value	Type
CRACK_T_USERNAME	16390	variable
CRACK_T_SECRET	16391	variable
CRACK_T_DOMAIN	16392	variable
CRACK_T_PIN	16393	variable
CRACK_T_CHALLENGE	16394	variable
CRACK_T_MESSAGE	16395	variable
CRACK_T_FIN	16396	basic

CRACK_T_USERNAME specifies the client user identity that's requesting authentication. The syntax and format of CRACK_T_USERNAME is specific to each LAM type.

CRACK_T_SECRET specifies secret information the client sends in an attempt to authenticate, for instance a password or passcode. The syntax and format of CRACK_T_SECRET is specific to each LAM type.

CRACK_T_DOMAIN specifies the domain or realm the client is requesting authentication credentials within. The syntax and format of CRACK_T_DOMAIN is specific to each LAM type.

CRACK_T_PIN specifies the client's PIN. The syntax and format of CRACK_PIN is specific to each LAM type.

CRACK_T_CHALLENGE specifies any challenge the gateway may choose to issue to the client. The syntax and format of CRACK_T_CHALLENGE is specific to each LAM type.

CRACK_T_MESSAGE specifies an ASCII string to be displayed to the user upon receipt of the corresponding CHRE payload. CRACK_T_MESSAGE is valid for all LAM types. Upon receipt, the contents of CRACK_T_MESSAGES SHOULD be displayed to the client user, typically along with the CHRE challenge.

CRACK_T_FIN specifies the server's response to the authentication exchange at all critical decision points specific to each LAM type. The following table defines the values for CRACK_T_FIN:

Finish Types	Value
RESERVED	0
CRACK_FIN_SUCCESS	1
CRACK_FIN_MORE	2

CRACK_FIN_SUCCESS indicates the gateway has successfully authenticated the client. This value successfully terminates the CRACK exchange. This value is legal for all LAM types.

CRACK_FIN_MORE indicates the gateway requires an additional round-trip to authentication the client. This is only legal for LAM types which define its use. It MUST NOT be used unless defined in the corresponding LAM profile.

3. The Protocol

This protocol uses digital signatures and proof of possession of a legacy secret to bind each party to the exchange as well as to the keying material that results from the exchange. This trust is acquired differently for the client and the gateway. The client trusts the gateway's public key either because it came from a certificate which is signed by a trusted certification authority or because the client trusts it by some out-of-band mechanism (for instance it is loaded into his policy store prior to hitting the road). The gateway trusts the client because the client has successfully authenticated himself using a legacy authentication method through a secure channel.

The reader should note that the channel in which the client's legacy proof is transmitted is secure from a man-in-the-middle attack due to the fact that the Diffie-Hellman public values and the attributes from the accepted offer, among other things, are signed. As in [HC98], the signature uses a pseudo-random function (prf), which is either negotiated in the initial SA payload or is the HMAC version [KBC96] of a hash function, over state from the exchange and keyed with "SKEYID".

The "SKEYID*" secret state is generated according to the rules for digital signature authentication of [HC98]. In other words.

```
SKEYID = prf(Ni_b | Nr_b, g^xy)
SKEYID_d = prf(SKEYID, g^xy | CKY-I | CKY-R | 0)
SKEYID_a = prf(SKEYID, SKEYID_d | g^xy | CKY-I | CKY-R | 1)
SKEYID_e = prf(SKEYID, SKEYID_a | g^xy | CKY-I | CKY-R | 2)
```

The data portion of the pseudo-random function consists of the clients's Diffie-Hellman public value concatenated with the gateway's

Diffie-Hellman public value concatenated with the client's cookie (from the [MSST98] header) concatenated with the gateway's cookie concatenated with the body of the client's initial SA offer. Graphically, the signature is over:

```
digest = prf (SKEYID, g^xi | g^xr | CKY-I | CKY-R | SAi_b)
```

Generally the pseudo-random function is the [KBC96] version of the negotiated hash function but this can be overridden if the signature algorithm is tied to a particular hash function (e.g. [DSS]) in which case the pseudo-random function will the the [KBC96] version of the hash function tied to the signature method.

The data being signed includes any padding prepended to the body of the payloads (for alignment to the length of the prime modulus) but does not include the ISAKMP header from any payload. The client MUST verify the signature. If the signature is not valid the exchange MUST be terminated by the client.

First, we describe the protocol abstractly using the aforementioned notation and then separate profiles are defined for each of the various LAM types.

3.1 IKE Challenge/Response Abstract Representation

The IKE Challenge/Response protocol is abstractly defined as follows:

Main Mode using CRACK is defined as

```
Client (I)                      Gateway (R)
-----------                     -----------
HDR, SAi,               --->
                        <---     HDR, SAr
HDR, KEi, Ni,
  [, CERTREQ]           --->
                        <---     HDR, [CERT, ] KEr,
                                   Nr, SIG
HDR*, CHRE              --->
                        <---     HDR*, CHRE
[ HDR*, CHRE            --->
                        <---     HDR*, CHRE ]
```

Aggressive Mode using CRACK is defined as

```
Client (I)                      Gateway (R)
-----------                     -----------
HDR, SAi, KEi, Ni
  [, CERTREQ]           --->
                        <---     HDR, SAr, [CERT, ] KEr,
                                 Nr, SIG
HDR*, CHRE              --->
                        <---     HDR*, CHRE
[ HDR*, CHRE            --->
                        <---     HDR*, CHRE ]
```

Where SIG is a digital signature of the aforementioned information. Any ambiguity about which key was used can be dispelled by optionally sending a certificate payload which indicates the public key that should be used to verify the signature.

Note that the number of messages in an exchange is not fixed. The gateway can respond with any number of challenges (CHRE payloads) to which the client responds with responses (also CHRE payloads) for each. When the gateway has successfully authenticated the client, it responds with a CHRE payload with an associated attribute list containing (at least) the CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS. Depending on the LAM Type, the gateway may respond with CRACK_FIN_MORE, indicating that the exchange needs to continue for an additional round.

3.2 IKE Challenge/Response Failures

CRACK requires the gateway to send ISAKMP Notify payloads under certain circumstances detailed in this section and elsewhere in this draft. These Notify payloads use the same format for the Notification Payload ([MSST98]) and differ only in the "Notification Data" field.

The Notification Payload MUST have the following format:

- Payload length - set to the value 28 + "Notification Data"
- DOI - set to the value zero (0) (ISAKMP)
- Protocol ID - set to the value one (1) (PROTO_ISAKMP)
- SPI Size - set to the value 16
- Notify Message Type - set to the value 24
- SPI - set to the ISAKMP initiator and responder cookies

If the contents of the CHRE payload(s) that the client sends fail to satisfy the legacy authentication method, the gateway MUST terminate the connection and MUST respond with an ISAKMP (AUTHENTICATION-

FAILED) [MSST98].

AUTHENTICATION-FAILED MUST include the following "Notification Data":

- LAM Type (two octets) - set to the associated LAM Type
- RESERVED (two octets) - MUST be zero (0) (alignment)

In addition AUTHENTICATION-FAILED SHOULD contain the following "Notification Data" when applicable:

- Status (variable length) - implementation-specific authentication failure status

If LAM Type, signature algorithm, or corresonding public-key requested by a CERTREQ cannot be located, the gateway MUST terminate the connection and MUST respond with an ISAKMP Notify (PROPOSAL-NOT-CHOSEN) [MSST98].

PROPOSAL-NOT-CHOSEN MUST include the following "Notification Data":

- LAM Type (two octets) - set to the LAM Type the server requires for the client; MAY be different than the LAM Type specified in the first CHRE payloads if the server required a different LAM Type than was offered
- RESERVED (two octets) - MUST be zero (0) (alignment)

In addition PROPOSAL-NOT-CHOSEN SHOULD contain the following "Notification Data" when applicable:

- Status (variable length) - authentication failure status

Because these Notify messages are only sent under the security of the Phase 1 shared secret and only after the gateway has proven its identity to the client, the client can trust the authenticity of these messages and MUST terminate the exchange upon receipt of any of these Notify messages.

4. Legacy Authentication Method (LAM) Profiles

Each defined LAM type uses the CHRE payload and LAM attributes in a different manner. This section profiles the acceptable use of each for the defined LAM types and details the list of acceptable attributes for each profile.

The Challenge/Response profile examples include the exchange of CERTREQ and CERT payloads which may be used when the client does not have access to the server's public-key or has access to multiple server keys. In other examples, the CERTREQ and CERT payloads are

omitted for simplicity, but these MAY be used with any of the defined profiles, according to the additional requirements in Section 3.1.

4.1 LAM Profiles: Password

The Password profile supports legacy operating system (OS) authentication along with proxy-based password authentication protocols, like RADIUS or TACACS+.

It is assumed in this example that the client has the gateway's public key, either through a certificate or a trusted raw public key, prior to initiation of the exchange. This example is given using Main Mode.

```
 Client (I)                    Gateway (R)
-----------                    -----------
HDR1, SAi,             --->
                       <---    HDR2, SAr,
HDR1, KEi, Ni          --->
                       <---    HDR2, KEr, Ni, SIG
HDR3*, CHRE1           --->
                       <---    HDR4*, CHRE2
```

For Password, the CHRE payloads are used as follows:

```
HDR3*, CHRE1           --->
```

The CHRE1 payload contains the client's username as a CRACK_T_USERNAME attribute and a password as a CRACK_T_SECRET attribute. The format of the client password is dictated by the corresponding host OS or proxy authentication server and may be either plaintext or binary.

```
                       <---    HDR4*, CHRE2
```

The CHRE2 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS.

The following attributes are defined for Password:

CRACK_T_USERNAME (client -> gateway, required)

CRACK_T_USERNAME is sent in the client's first CHRE payload and MUST contain the client's username which is used as an index key by the host OS or proxy password authentication server.

CRACK_T_SECRET (client -> gateway, required)

CRACK_T_SECRET is sent in the client's first CHRE payload and MUST contain the client's password.

CRACK_T_DOMAIN (client -> gateway, optional)

CRACK_T_DOMAIN is sent in the client's second message and MAY be used to specify the authentication domain that the client is requesting authentication within.

CRACK_T_FIN (gateway -> client, required)

CRACK_T_FIN is used to successfully terminate the exchange.

4.2 LAM Profiles: One-Time Password

The OTP profile supports both the S/KEY and OTP one-time password schemes.

It is assumed in this example that the client has the gateway's public key, either through a certificate or a trusted raw public key, prior to initiation of the exchange. The example is given using Aggressive Mode.

```
 Client (I)                    Gateway (R)
-----------                    -----------
HDR1, SAi, KEi, Ni     --->
                       <---    HDR2, SAr, KEr, Nr, SIG
HDR3*, CHRE1           --->
                       <---    HDR4*, CHRE2
HDR5*, CHRE3           --->
                       <---    HDR6*, CHRE4
```

For OTP, the CHRE payloads are used as follows:

```
HDR3*, CHRE1           --->
```

The CHRE1 payload contains only any associated attributes such as a username.

```
                       <---    HDR4*, CHRE2
```

The CHRE2 payload contains the OTP server's challenge text which MUST be displayed to the client user.

```
HDR5*, CHRE3           --->
```

The CHRE3 payload contains the client's one-time password response.

<---- HDR6*, CHRE4

The CHRE4 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS.

The following attributes are defined for OTP:

CRACK_T_USERNAME (client -> gateway, required)

CRACK_T_USERNAME is sent in the client's first CHRE payload and MUST contain the client's username which is used as an index key by the OTP server.

CRACK_T_CHALLENGE (gateway -> client, required)

CRACK_T_CHALLENGE is sent in the gateway's first CHRE payload and MUST contain the OTP challenge to be issued to the client.

CRACK_T_SECRET (client -> gateway, required)

CRACK_T_SECRET is sent in the client's second CHRE payload and contains the client's one-time password.

CRACK_T_MESSAGE (gateway -> client, optional)

CRACK_T_MESSAGE is optionally sent in any server message and MAY by used by the server to provide optional text to be displayed to the user along with any associated challenge text.

CRACK_T_FIN (gateway -> client, required)

CRACK_T_FIN is used to successfully terminate the exchange.

4.3 LAM Profiles: Challenge/Response

The Challenge/Response profile supports various token cards that follow a standard challenge/response exchange. The client's token card information (the response) depends on the gateway's request (the challenge).

It is assumed in this example that the client does not have the gateway's public key and requires a certificate issued by a trusted Certification Authority. Note that in this case, identity protection of the gateway is lost. Whether a certificate is requested and sent or not, the client's identity is never open to a passive attack (i.e. the client retains identity protection regardless).

The following example shows an exchange where a full

challenge/response exchange is followed:

```
 Client (I)                     Gateway (R)
-----------                     -----------
HDR1, SAi, KEi, Ni,
  CERTREQ              --->
                       <---     HDR2, SAr, CERT, KEr,
                                  Nr, SIG
HDR3*, CHRE1           --->
                       <---     HDR4*, CHRE2
HDR5*, CHRE3           --->
                       <---     HDR6*, CHRE4
```

If more challenges were required to authenticate this client, message six would be another CHRE payload containing a challenge to the client. This would force a message seven which would be another CHRE payload. This can be repeated until the gateway authenticates the client (or authentication fails, see below).

Alternatively, some challenge-response tokens cache their last computed result and do not require a challenge from the gateway unless they get out of sync (perhaps due to intrusion detection). In this case, the gateway may be able to authenticate the client in the second message and would return, assuming success a CHRE2 containing CRACK_T_FIN attribute with the value of CRACK_T_FIN_SUCCESS. There would also be no fifth nor sixth message.

The following example shows an exchange where the client can pre-compute his expected response:

```
 Client (I)                     Gateway (R)
-----------                     -----------
HDR1, SAi, KEi, Ni,
  CERTREQ              --->
                       <---     HDR2, SAr, CERT, KEr,
                                  Nr, SIG
HDR3*, CHRE1           --->
                       <---     HDR4*, CHRE2
```

For Challenge/Response, the CHRE payloads are used as follows:

```
HDR3*, CHRE1           --->
```

When the client is using a token that can compute the next expected response without requiring a challenge, the CHRE1 payload contains the client's username and expected response. When the client does not have an expected response, or has chosen not to use

the current one for whatever reason, the CHRE payload contains only the client's username.

<--- HDR4*, CHRE2

The CHRE2 payload contains the gateway's challenge text which MUST be displayed to the client user unless the client has presented an expected response (as above) in which case this is identical to CHRE4 below.

HDR5*, CHRE3 --->

The CHRE3 payload, when used, contains the client's response to the gateway challenge.

<--- HDR6*, CHRE4

The CHRE4 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS.

The following attributes are defined for Challenge/Response:

CRACK_T_USERNAME (client -> gateway, required)

CRACK_T_USERNAME is sent in the client's second message and MUST contain the client's username which is used as an index key for authentication by the server.

CRACK_T_SECRET (client -> gateway, required)

CRACK_T_SECRET contains the client's response and is sent in the client's second message if an anticipated challenge is used, and in the client's third message if the client is responding to a gateway challenge.

CRACK_T_PIN (client -> gateway, optional)

CRACK_PIN is optionally sent in any client message and MAY by used if the authentication protocol also requires the client to provide a PIN.

CRACK_T_MESSAGE (gateway -> client, optional)

CRACK_MESSAGE is optionally sent in any server message and MAY by used by the server to provide optional text to be displayed to the user along with any associated challenge text.

CRACK_T_FIN (gateway -> client, required)

CRACK_T_FIN is used to successfully terminate the exchange.

4.4 LAM Profiles: SecurID

The SecurID profile supports the RSA SecurID protocol. With SecurID the client will be passing the output of the SecurID card as the body of the first CHRE payload (in the second message it sends) and its identity as an associated CRACK_T_USERNAME attribute. Assuming the client and gateway are in sync (i.e. they are not in "Next Code" mode) there is a single CHRE payload.

It is assumed in this example that the client has the gateway's public key, either through a certificate or a trusted raw public key, prior to initiation of the exchange.

The following example shows a simple SecurID authentication using aggressive mode:

```
 Client (I)                    Gateway (R)
-----------                    -----------
HDR1, SAi, KEi, Ni     --->
                       <---    HDR2, SAr, KEr,
                                 Nr, SIG1
HDR3*, CHRE1           --->
                       <---    HDR4*, CHRE2
```

For simple SecurID, the CHRE payloads are used as follows:

```
HDR3*, CHRE1           --->
```

The CHRE1 payload contains the client's username and the current Passcode displayed by the client's SecurID token.

```
                       <---    HDR4*, CHRE2
```

The CHRE2 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS.

When the client and gateway clocks are slightly out of sync, the gateway will respond with an additional challenge payload to which the client MUST respond with another reponse payload. This is known as "Next Code" mode.

The following example shows a SecurID authentication where "Next Code" mode is required:

```
    Client (I)                      Gateway (R)
   -----------                      -----------
   HDR1, SAi, KEi, Ni     --->
                          <---      HDR2, SAr, KEr,
                                      Nr, SIG
   HDR3*, CHRE1           --->
                          <---      HDR4*, CHRE2
   HDR5*, CHRE3           --->
                          <---      HDR6*, CHRE4
```

For SecurID with "Next Code", the CHRE payloads are used as follows:

```
   HDR3*, CHRE1           --->
```

The CHRE1 payload contains the client's username and the current Passcode displayed by the client's SecurID token.

```
                          <---      HDR4*, CHRE2
```

The CHRE2 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_MORE.

```
   HDR5*, CHRE3           --->
```

The CHRE3 payload contains the client's next Passcode displayed by the client's SecurID token.

```
                          <---      HDR6*, CHRE4
```

The CHRE4 payload contains a CRACK_T_FIN attribute with the value of CRACK_FIN_SUCCESS.

The following attributes are defined for SecurID:

CRACK_T_USERNAME (client -> gateway, required)

CRACK_T_USERNAME is sent in the client's second message and MUST contain the client's username which is used as an index key by the ACE server.

CRACK_T_PIN (client -> gateway, optional)

CRACK_T_PIN is sent in the client's second message and MAY be used when the SecurID card is not a PINPAD card.

CRACK_T_MESSAGE (gateway -> client, optional)

CRACK_T_MESSAGE is optionally sent in any server message and MAY

by used by the server to provide optional text to be displayed to the user along with any associated challenge text.

CRACK_T_FIN (gateway -> client, required)

CRACK_T_FIN is used to successfully terminate the exchange and to request the client continue under "Next Code" mode.

4.5 LAM Profile Matrix

Each of the LAM's supported by IKE Challenge/Response fall into one of the defined LAM profiles. This section details the classification for those methods, including all of the types defined for the experimental XAUTH protocol [PB99].

Password
- DIAMETER
- LDAP
- NDS (Netware Directory Services)
- NT Domain
- RADIUS
- TACACS
- TACACS+
- UNIX Login

OTP
- OTP
- S/KEY

Challenge/Response
- AXENT Defender
- CheckPoint ActivCard
- CRYPTOCard CRYPTOCard
- Digital Pathways SNK
- LeeMah InfoCard
- Secure Computing SafeWord (Enigma Logic DES Gold)

SecurID
- RSA SecurID

5. The IKE Challenge/Response Vendor ID Signature

This memo describes a protocol that lives on top of [MSST98] and as a companion to [HC98]. These standards-track protocols reserve some of their "magic number" space for private use by mutually consenting parties. It is from this number space that this memo obtains some of the "magic numbers" it needs (payload types, exchange value, attributes). As part of the "mutually consenting parties" part of

the requirement implementors of this protocol are encouraged to use a Vendor ID payload to announce willingness to engage in this protocol. The contents of the Vendor ID payload will be the following hexadecimal string: 0x13f11823f966fa91900f024ba66a86b, which is the result of an MD5 hash of "IKE Challenge/Response for Authenticated Cryptographic Keys (Revised)" without the quotation marks. An [HC98] implementation that implements this protocol that obtains a Vendor ID payload with this string in the body of the payload can assume that the sender of the Vendor ID payload has likewise implemented this protocol and is therefore a "mutually consenting party".

If this protocol is advanced to standards-track status IANA will assign new "magic numbers" out of the appropriate number spaces (the "magic numbers" will no longer be from the private use ranges) and the requirement to use a Vendor ID payload will go away.

6. Security Considerations

The channel that results from the exchange of the first two messages is secured because the gateway signs his Diffie-Hellman public value and it is the resulting SKEYID state (see [HC98]) that protects the channel. The channel is secured from the client's perspective because he knows that the gateway was the actual source of the Diffie-Hellman public value and is an active party to the exchange. The channel is secured from the gateway's perspective because the client has proved proof-of-possession of a long-term shared secret and would not have sent his sensitive information if a man-in-the-middle was detected by the client.

While this seems to be a weak form of assurance, the exchange could only be foiled by an intentionally malfunctioning client and if that is the case then all bets are off regardless of the method of authentication. (If Alice and Bob establish IPsec SA's in the traditional fashion, using a [HC98] exchange nothing could stop Alice from sending all the sensitive information Bob conveys to her to Eve.) Also note that this technique is used in other popular on-line certificate enrollment schemes ([MLSW99]).

As noted, this whole scheme can fail if the client is intentionally malicious. Also, if the token card and knowledge of how to generate valid credentials is conveyed to a third-party this scheme would fail (but not due to any protocol failure).

The number of messages in this protocol is dictated by the type of legacy authentication method employed. Since this protocol is open-ended, a host implementation may wish to limit the number of CHRE round-trips using locally defined policy.

Acknowledgments

The authors would like to thank the sales and marketing staff of all companies who said, "Just give us something that uses token cards!" We would like to recognize Roy Pereira and Stephane Beaulieu, authors of [PB99], which was borrowed from liberally in creation of this memo.

References

[Bra96] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[CR98] P. Calhoun, A. Rubens, "DIAMETER - Base Protocol", draft-calhoun-diameter-02.txt, March 1998, a work in progress.

[DSS] National Institute of Standards and Technology, U.S. Department of Commerce, "Digital Signature Standard", FIPS 186, May 1994.

[Hal95] N. Haller, "The S/KEY One-Time Password System", RFC1760, February 1995.

[HC98] D. Harkins, D. Carrel, "The Internet Key Exchange", RFC2409, November 1998.

[HM96] N. Haller, C. Metz, "A One-Time Password System", RFC1938, May 1996.

[KBC96] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[MLSW99] M. Myers, X. Liu, J. Schaad, and J. Weinstein, "Certificate Management Messages over CMS", draft-ietf-pkix-cmc-05.txt, a work in progress.

[MSST98] D. Maughan, M. Schertler, M. Schneider, J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC2408, November 1998.

[PB99] R. Pereira, S. Beaulieu, "Extended Authentication within ISAKMP/Oakley", draft-ietf-ipsec-isakmp-xauth-05.txt, September, 1999, a work in progress.

[Pip98] Piper, D., "The Internet IP Security Domain Of Interpretation for ISAKMP", RFC 2407, November 1998.

[PKCS1] B. Kaliski, J. Staddon, "PKCS #1: RSA Cryptography Specifications Version 2", September 1998.

[RASW97] C. Rigney, A. Rubens, W. Simpson, S. Willens, "Remote Authentication Dial In User Service (RADIUS)", RFC2138, April 1997.

[RSA] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", Communications of the ACM, v. 21, n. 2, February 1978.

Authors' Address

Dan Harkins <dharkins@cips.nokia.com>
Derrell Piper <ddp@cips.nokia.com>
Nokia Corporation
1538 Pacific Ave
Santa Cruz, CA 95060-9311
United States of America