Network Working Group
Internet Draft

Bellovin and Moskowitz
AT&T Labs Research

Expiration Date: May 2000

February 2000

Client Certificate and Key Retrieval for IKE

draft-bellovin-ipsra-getcert-00.txt

1. Status of this Memo

2. Abstract

xxxx

3. Introduction

(Insert justification, possibly copied wholesale from section 1.1 and maybe 2.1/2.2 of draft-kelly-ipsra-userauth-00.txt)

We consider inadvisable to change IKE [RFC2409] to meet these needs. IKE is a complex protocol; adding more features to it is a bad idea. Instead, we propose a layered approach: use standard IKE, with certificates, but provide a simple mechanism to provide clients with keys and certificates.

A number of objections have been raised to using certificates. The most important is that we lack a public key infrastructure (PKI). We do not agree that this is an obstacle. Our proposal provides a simple mechanism for certificate generation and retrieval, while still relying on legacy authentication infrastructures. Furthermore, we provide for an easy migration path to certificate use once organizational PKIs are deployed.

Our purpose at this point is not to present a firm protocol. Rather, we sketch several ideas for what such a protocol could look like. Final details can be determined if and when the working group opts for this path.

In the interests of simplicity, we have chosen to reuse standard protocols and components. In particular, we use HTTP [RFC2616] for transport, HTML [RFC1866] as a data representation and TLS [RFC2246] for confidentiality. However, we do not mandate (or even necessarily encourage) use of a actual Web browser for certificate retrieval.

As an alternative, we present a transient shared secret generation mechanism for IKE.

3.1. Requirements Keywords

The keywords "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", and "MAY" that appear in this document are to be interpreted as described in [RFC2119].

4. Protocol Definition

4.1. Client-Side Certificate Generation

The first proposal uses the <KEYGEN> HTML tag understood by recent versions of Netscape Navigator. The client first connects to TCP port SIGNCERT, and initiates a TLS negotiation. It then sends the following HTTP command:

```
GET /SignClientCert/username HTTP/1.1
```

followed by a null line, where "username" is replaced by a site-specific identifier for that user. (Other HTTP commands MAY be entered; however, they MUST be ignored by the server.) The server responds, using HTTP, with a message that includes at least the following

```
<FORM ACTION=anything>
<KEYGEN NAME=username CHALLENGE="challenge string">
</FORM>
```

Any text outside of the FORM and </FORM> tags MUST be ignored by the key generation process. However, the text MAY be displayed to the user if desired.

In response the client generates a private/public key pair of appropriate size. (How that length is determined is beyond the scope of this document. Most likely, it would be done by local configuration in accordance with administrative fiat.) The client then signs its public key and challenge, encodes them as DER strings (both per http://home.netscape.com/eng/security/ca-interface.html), and uploads them via an HTTP request using the URL specified in the FORM tag. This request will generally include an appropriate Authorization: line, using whatever form of authentication is locally preferred. If it does not, the server MUST return a 401 error line, per [RFC2616] and [RFC2617]; the client MUST then resubmit the request with the appropriate authentication information. (This two-phase process is permitted in order to support challenge/response forms of authentication.)

The returned HTML includes the signed certificate in base 64 encoding, bracketed by

```
-----BEGIN CERTIFICATE-----
```

and

-----END CERTIFICATE-----

lines. That text is used as the IKE certificate; all other text MUST be ignored.

It is also necessary to send back the certificate authority information; that is bracketed by

-----BEGIN CA CERTIFICATE-----

and

-----END CA CERTIFICATE-----

lines.

Obviously, the forgoing scheme has been carefully designed to permit the client to be a Web browser. If so, the browser MUST have some mechanism that will let it automatically export both the private key and the returned certificate to IKE.

4.2. Server-Side Key Pair Generation

Client-side key generation can be slow. An alternative is to have the server generate the key pairs. The server can generate many spare key pairs during idle periods; it is also likely to have better sources of randomness than most clients do. While server-side generation of private keys is normally not as secure, there is no danger here; the resulting key is used solely for access, which is at the complete discretion of the server in any event.

A similar TLS-protected HTTP sequence is used to request a server-generated key:

GET /SignClientCert/username HTTP/1.1

followed by a null line, where "username" is replaced by a site-specific identifier for that user. (Other HTTP commands MAY be entered; however, they MUST be ignored by the server.) The server responds, using HTTP, with a message containing the certificate and private key.

This time, however, the initial request must be authenticated. More precisely, if permitted by the local authentication policy the client MUST include an appropriate Authentication: line. If the authentication fails, or if appropriate credentials are not included, the server MUST respond witha 401 error code, at which point the

client should retransmit the request, this time including the appropriate credentials.

The HTTP returned by the server MUST include the certificate in the same format:

-----BEGIN CERTIFICATE-----

and

-----END CERTIFICATE-----

as in the prior case. (CA information is also sent back, as described earlier.) In this case, however, it MUST also send the corresponding private key:

-----BEGIN PRIVATE KEY-----

and

-----END PRIVATE KEY-----

encoded in base 64. All returned content besides those two blocks MUST be ignored by the client.

4.3. Server-Side Key Storage

Server-side key storage can be used as part of a full local PKI, but in situations where the client machines cannot or should not possess the private key permanently. This might arise with shared machines or IPsec-capable "kiosk" machines. The behavior is similar to the previous case; however, the private key is returned in a block of the following format:

-----BEGIN PRIVATE KEY, cipher, validation-----

and

-----END PRIVATE KEY-----

where "cipher" denotes the cipher used to encrypt the private key and "validation" denotes the check sequence used to verify decryption.

The private key is encrypted before it is uploaded to the authentication server, to protect users' long-term secrets against server compromise. The key is encrypted in CFB-8 mode (OPEN ISSUE: should this be CBC mode, since IPsec needs that code anyway?);

depending on the block size of the cipher, the first n bytes of the block are used as the IV. Currently, the only mandatory cipher is 3DES, in which case the IV is 8 bytes long; when AES support is added, a 16-byte IV will be used. Entry of the decryption key by the user is implementation-dependent. (OPEN ISSUE: should we define a standard mapping from a human-friendly string to this key?)

The only current "validation" field required is SHA-1. The SHA-1 check field is the last 20 bytes of the plaintext. To validate a decryption, the client must decrypt all of the returned text after the IV and apply the appropriate function to the plaintext minus the check field. If the check field returned matches the calculated check field, the decryption is correct.

4.4. Server-Generated Shared Secrets

The techniques outlined above require no changes to the IPsec server, nor do they impose any requirements on the location of the authentication server. In particular, the two need not be co-located, nor need they talk; the certificates generated (or returned) by the authentication server are ordinary certificates that can be accepted by standard IPsec servers. It is, however, desirable to configure the latter to accept all certificates of a certain format if signed by an authentication server. The latter in turn can possess certificates signed by a root key whose certificate is known to the IPsec servers, providing further decoupling.

However, this independence comes at a cost: the TLS negotiation necessary to protect the HTTP-like exchange is expensive, but it will be followed by an equally expensive IKE exchange. Our last variant involves generation of a short-lived shared secret by the authentication server; in addtion to being returned to the client, this secret is then transmitted to the IPsec server along with the client's (current) IP address. This permits shared-secret IKE authentication, which avoids the expense of the digital signature operations in certificate-based IKE.

The request is made by an HTTP command:

```
GET /SharedSecret/IPaddress/username HTTP/1.1
```

(The client's IP address is included in case the request is being sent via an HTTP proxy.) Authentication is performed as in the previous two cases.

The returned value is encoded in base 64 and is bracketed by

```
-----BEGIN SECRET-----
```

and

-----END SECRET-----

lines. The IP address of the proper IPsec server -- there may be several, and the client needs to know which has been informed of the secret -- is similarly encoded and is bracketed by

-----BEGIN IPADDRESS-----

and

-----END IPADDRESS-----

lines.

Transmission of the secret to the IPsec server is beyond the scope of this document.

5. Authentication Techniques

Although this document is carefully agnostic about the user authentication techniques to be used, there are two underlying assumptions. First, we assume that the authentication is actually being performed by a back-end RADIUS server with its accompanying database. Second, we wish to support a variety of common authentication techniques, including ordinary passwords, time-varying tokens, and challenge/response tokens. All are believed to be accomodated by this framework.

6. Certificate Characteristics

Signed or generated certificates should, as noted, have a distinctive name format that can be recognized and accepted by the IPsec servers. The expiration time of the certificates is limited by local policy on reuse. In some cases, these certificates will valid for several hours (and hence several sessions, if needed); in other cases, they will expire within a very few minutes and are thus practically usable only for a single IKE exchange. (Note that this also requires tight time synchronization between the authentication server, the IPsec servers, and -- if they care -- the IPsec clients.

7. Syntax

Clearly, this draft does not pay too much attention to syntactic issues. It is worth discussing whether or not certificates should be sent back in PKCS#7 format instead.

8. Security Considerations

The client -- the program and the ultimate human -- MUST check the server's TLS certificate to guard against man-in-the-middle attacks.

<<more>>

9. Acknowledgements

Paul Hoffman contributed many useful ideas to the shared secret section of this document.

10. References

[RFC1866] "Hypertext Markup Language - 2.0". T. Berners-Lee, D. Connolly. November 1995.

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels". S. Bradner. March 1997.

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

[RFC2246] "The TLS Protocol Version 1.0". T. Dierks, C. Allen. January 1999.

[RFC2401] "Security Architecture for the Internet Protocol". S. Kent, R. Atkinson. November 1998.

[RFC2409] "The Internet Key Exchange (IKE)". D. Harkins, D. Carrel. November 1998.

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

[RFC2617] "HTTP Authentication: Basic and Digest Access Authentication". J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, L. Stewart. June 1999.

[RFC2692] "SPKI Requirements". C. Ellison. September 1999.

[RFC2693] "SPKI Certificate Theory". C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, T. Ylonen. September 1999.

11. Author Information

Steven M. Bellovin
AT&T Labs Research
Shannon Laboratory
180 Park Avenue
Florham Park, NJ 07974
USA
Phone: +1 973-360-8656
Email: smb@research.att.com

Robert G. Moskowitz
ICSA.net
1200 Walnut Bottom Rd.
Carlisle, PA 17013
Email: rgm@icsa.net