INTERNET-DRAFT
Expires: January 2001

S. Farrell
Baltimore Technologies
M. Nystr÷m
RSA Security
July 2000

# Securely Available Credentials
<draft-farrell-sacred-00.txt>

## Status of this Memo



## Abstract

As the number, and more particularly the number of different types, of devices, connecting to the Internet increases, credential mobility becomes an issue for IETF standardization. This draft presents some requirements, a strawman framework and outlines a protocol for securely available credentials.

## Table Of Contents

## 1. Introduction.

Private credentials are used to support various Internet protocols, e.g. S/MIME, IPSec, TLS. In a number of environments end users wish to use the same set of private credentials from different end user equipment. In a "typical" desktop environment, the user already has many tools available to allow import/export of these credentials. However, with some devices, especially wireless and other more constrained devices, the tools required simply do not exist.

This leads to a high level requirement for protocol(s) that allow these private credentials to be made available over the Internet. Such credentials are highly sensitive, since they are the basis for many of the security features of the Internet protocols referred to above. There are therefore stringent security requirements that any proposed protocol(s) must meet.

The typical credential envisaged here is often either the entirety or just the private part of what is often called a personal security environment or PSE [RFC2459]. Other forms of credential are of less direct interest here.

Since the credential concerned may be required to be present in order to use PKI based authentication mechanisms, we cannot rely on such mechanisms, at least for "download" (towards the end user) operations. Note also that since PSEs often include "root" CA information, (as well as private keys), it may not be possible to depend on PKI based authentication of network servers.

In the remainder of this draft we present a set of requirements, propose a general framework and provide an outline of a protocol to meet the requirements.

The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC2119].

<<editorial comments are in angle brackets, like this>>

## 2. Requirements

The following requirements assume that the is a credential server from which credentials are downloaded to the end user device, and to which credentials are uploaded from an end user device.

1. Credential download (to end user) and upload (to credential server) MUST be supported
2. Credentials MUST only be downloadable following user

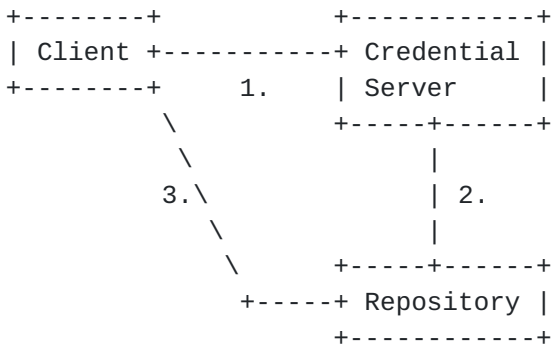
authentication

3. Credential upload MAY require user authentication
4. Users MUST be able to manage their credentials stored on the credential server <<not further developed so far>>

5. The protocol(s) MUST support a range of user authentication mechanisms
6. The protocol(s) MUST support a range of credential formats.
7. The protocol(s) MUST be extensible so as to be able to support a wide range of end user devices, accessed using a range of transport protocols
8. Different end user devices MAY be used to download/upload/manage the same set of credentials
9. Credentials MUST be protected whilst in transit
10. The credential server MUST NOT have easy access to the cleartext credentials
11. It MUST be possible to ensure the confidentiality/privacy of all interactions between users and credential servers
12. Credential servers MUST be authenticated to the user for all operations except (possibly) download
13. It MUST be possible to authenticate the credential server to the user prior to download (if the user is able to verify the authentication)
14. It MUST be possible to cache credentials locally on the end user device without unnecessarily exposing the private parts of the credential
15. The user SHOULD only have to enter a single secret value in order to download a credential.

## 3. Framework

The diagram below shows the components involved in the sacred framework. The numbers refer to protocols that may be defined.




Client: The entity that wants the credential.

Credential Server: The server that knows how/who/when to give the credential to the client.

Repository: Where the credentials are stored. The repository might have access control features but those generally aren't sufficient in

themselves for protecting credentials.

Private information in credentials must be protected by encryption. The key for this encryption can be derived from a password, but can also be something stronger. As an example of a stronger method, one

could consider the user authenticating to the credential store using some strong authentication method (not requiring public-key cryptography) and then downloading (over a confidentiality-protected connection) not only the credentials but also (parts of) the key to unlock the card. This way, the key used to protect the credentials will not be derived from the user's password solely. Further, in order to make it harder for a credential store administrator to find out about users' private objects, the key stored at the operator's server may be combined with a user-derived key to form an actual key-encryption key.

Credentials must be integrity protected, since it would otherwise be possible for an attacker to substitute parts of the credentials (e.g. trusted certificates) with something more advantageous for the attacker. Once again, the key for this integrity-protection may be derived from a user password, but in this case, it could also be the user's own private key (assuming that private objects on the token are decrypted before the integrity check is done).

The basic framework is as follows:

- credential servers MUST NOT be presented with credentials in cleartext form
- all user, credential server communications MUST use HTTP over TLS
- only TLS ciphersuites providing strong confidentiality MAY be used

The framework for uploads is as follows:

- the credential server MUST be authenticated, that is only TLS ciphersuites providing strong confidentiality and server authentication MAY be used
- the user sends a POST message with the POST data containing the credentials and other required information
- upload of credentials can either be authenticated, or unauthenticated, any authentication information (for the upload operation itself) is carried inside the POST data
- the POST data MUST include information which will be used later for authentication during download, this information may take various forms

<<XML is just being used here for convenience, and may well be replaced.>>

This upload message is described in the following XML DTD:

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT SacredUploadData (OperationAuthenticationData?,
                            DownloadAuthenticationData,
                            CredentialData)>
```

```
   <!ELEMENT OperationAuthenticationData (#PCDATA) >
   <!ATTLIST OperationAuthenticationData method ID #REQUIRED>
   <!ELEMENT DownloadAuthenticationData (#PCDATA)>
   <!ATTLIST DownloadAuthenticationData method ID #REQUIRED>
```

```
<!ELEMENT CredentialData (#PCDATA)>
<!ATTLIST CredentialData format ID #REQUIRED>
```

OperationUploadData, if present, is intended to authenticate the user for the upload operation. DownloadAuthenticationData, which MUST be present, contains indicates how the user will authenticate for subsequent downloads and contains method specific data. CredentialData contains the format information and the actual credential itself.

The HTTP response to this POST message SHOULD be a text/html page containing a HREF, which indicates URL from which the credential MAY subsequently be downloaded.

The download operation is as follows:

- HTTP authentication is used to authenticate users to credential servers(*)
- a simple GET request for the download URL is issued
- the HTTP response contains the credential and information about the credential format

(*) This assumes that an HTTP SASL authentication scheme is defined in addition to the current HTTP Basic and Digest authentication schemes [RFC2617]. Work on such a scheme is on-going [<<REF>>].

The HTTP response MUST contain a CredentialData element as defined in the above DTD.

<<management operations are TBS>>

## 4. Outline Protocol

In terms of the framework above, the specific protocol proposed as mandatory to implement requires support for:

- OTP [RFC2289] and, optionally, S/KEY [RFC1760] for user authentication
- uploads are unauthenticated and contain the OTP or S/KEY username for downloads, and optionally the pass-phrase in the DownloadAuthenticationData
- PKCS#15 [PKCS#15] is the credential format <<profile TBS>>
- the OTP (or S/KEY) passphrase is completely separate from any password based encryption keys used to protect the PKCS#15 credential <<this breaks requirement 15 above, but can be fixed later>>

## 5. A "strong" password scheme

In this section we present an idea for a "strong" password scheme. The idea here is to tie down the "MUST" implement mechanisms for the framework. <<NOTE: For now, this scheme is merely illustrative, before being adopted it would have to be checked out in detail.>>

Assumption: User has credential Cd.

To secure a credential:

Idea is to hash Cd and derive both a key and a password from the hash. The security of the key/password derivation is driven by a "level" (number of bits of hash used to derive password).

H=hash(Cd)

K=fold(H,level)
PWD=password-generator(K)
secured-credential=encrypt(K,Cd)

password-generator can be the OTP six word picker scheme (apparently about 11 bits per word)

fold function can fold/truncate the hash to a key, based on the number of bits indicated in the level (e.g. level1 is 40 bit; level 2 = 60 bit; level 3 = 80 bit)

User uploads nickname + secured-credential to credential server.

When roaming:

user->server: nickname
server->user: secured-credential

To expose credential:

K=reverse-password(PWD)
recovered=decrypt(K,secured-credential)
status=compare(fold(H(recovered)),K)

## 6. Open Issues

This document is intended to foster discussion of the requirements, framework and protocols that might be used to support credential mobility. However, the authors recognize that there are many issues that remain to be resolved. Some of the most pressing are:

IPR: Some of the useful mechanisms are encumbered

Robustness: Credential stores should not unacceptably increase the potential for denial-of-service or other attacks

Performance: Users should not typically have to wait too long for access to credentials

Bootstrapping: The use of, e.g. TLS server authentication, depends on the client having a (set of) trusted root(s) - as the protocol

may be providing these roots, there may be some hard bootstrapping issues.

Finally, we note that whether or not the user authentication, credential protection and specific credential formats should be separated, or should be intertwined, is an open issue that warrants careful consideration.

## 7. Security Considerations

Mobile credentials will never be as secure as a "pure" smart card solution. However, reasonable security may be accomplished through some simple means, as outlined above. One should keep in mind, however, that platforms to which credentials are downloaded usually cannot be regarded as tamper-resistant, and it therefore is not too hard to analyze contents of their memories. Further, storage of private keys, even if they are encrypted, on a credential server, will be unacceptable in some environments.

## 8. References

[PKCS15] "PKCS #15 v1.1: Cryptographic Token Information Syntax Standard," RSA Laboratories, June 2000.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026.

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

[RFC2459] Housley, R., Ford, W., Polk, T, & Solo, D., "Internet Public Key Infrastructure - X.509 Certificate and CRL profile", RFC 2459.

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

Author's Addresses

Stephen Farrell,
Baltimore Technologies,
61/62 Fitzwilliam Lane,
Dublin 2,
IRELAND

tel: +353-1-647-3000
email: stephen.farrell@baltimore.ie

Magnus Nystr÷m
RSA Security

Box 10704
121 29 Stockholm
Sweden

tel: +46 8 725 0900
email: magnus@rsasecurity.com

Full Copyright Statement