D. Kemp
NSA
INTERNET-DRAFT
Expires 25 December 1996
June 11, 1996

The Public Key Login Protocol

<draft-kemp-auth-pklogin-00.txt>

1. Status of this memo

2. Abstract

This document specifies the Public Key Login protocol, a security protocol which uses digital signatures to provide strong authentication for interactive sessions. It is suitable for use with unmodified interactive session protocols, dial-up Network Access Servers, remote mail servers, and similar applications. It provides functionality similar to One Time Passwords or handheld authentication tokens, and supports both one-way and mutual authentication. The protocol does not require the use of public key certificates and the associated infrastructure but can make use of them if available.

The Public Key Login protocol provides strong authentication at the beginning of a session and perhaps periodically thereafter, but does not provide data integrity or confidentiality. If these additional security services are required, other mechanisms must be used.

3. Introduction

Despite the development of communication protocols and mechanisms providing a full suite of security services (IPSEC, TLS, GSS-API), there remains a need for authentication mechanisms that can be used in environments with unmodified application and network protocols (Telnet, TCP/IP, PPP) and no security infrastructure (PKIX, DNSSEC, Kerberos, etc.). Plain password authentication is still widely used in these “traditional” environments because it is well understood and easy to implement. The critical weaknesses of plain passwords are also well understood, and several approaches have been developed to provide stronger forms of authentication. Applications for strong authentication include:

* Firewall proxies.
* Modem connections to dial-up Network Access Servers, using authentication/authorization/accounting protocols such as [RADIUS] and TACACS+.
* Access to POP3 or IMAP4 remote mail servers.
* Interactive sessions to individual hosts in a local environment, using a common authentication framework such as OSF Pluggable Authentication Modules [PAM].

Strong authentication methods designed specifically to be easy-to-deploy in traditional environments include:

* the One Time Password [OTP] protocol (once known as S/Key(TM)), which uses a one-way cryptographic hash function to compute passwords that cannot be intercepted and reused.
* various handheld authentication tokens (SecureID, SNK, SafeWord), which use a shared secret key to either transform the time-of-day into a non-reusable password, or compute a response to a non-repeating challenge.

This document proposes a third type of protocol, based on public key digital signatures, to provide strong authentication in traditional environments. Like the alternatives, the Public Key Login (PKL) protocol is relatively easy to incorporate into existing systems, but it addresses the following limitations of other strong authentication methods:

One Time Password systems must be initialized to allow a fixed number of logins. After that number of uses, the server must be reinitialized with a new seed before authentication can be used again. For human-to-machine authentication reinitialization is not needed very often, but machine-to-machine authentication (e.g. polling a mail server every five minutes) can use up the available logins fairly quickly and reinitialization can become a burden. In addition, OTP relies on a user-generated secret which may be subject to guessing if not properly chosen. OTP is subject to a “race attack” by an eavesdropper if the server does not take special precautions to disallow multiple simultaneous login attempts. In contrast, the PKL protocol has no fixed login count, it uses cryptographic quality random private keys, and it does not require special protection from eavesdroppers because a unique challenge is used for each login attempt.

Handheld authentication tokens rely on possession of a secret key as proof of identity. This does not provide non-repudiation - anyone able to obtain the secret key (by compromising a server, for example) is able to masquerade as the user to any servers using that secret key. Since user tokens contain only a small number of keys (typically from one to ten), users wishing to authenticate to a large number of servers must either carry multiple tokens or share keys among multiple servers. The PKL protocol needs only one user key to authenticate to any number of servers. Each party has sole control of its own private key and the level of protection provided for it. If signature generation is implemented in a hardware token, the user's private key need not ever be exposed outside the token; this requires an attacker to take physical possession of a user's token in order to masquerade as that user.

Neither OTP nor current handheld tokens allow the user to authenticate a server; the PKL protocol supports both unilateral and mutual authentication.

PKL can be implemented in either hardware or software, but it is particularly suited for use with portable cryptographic devices in the form of PC Cards, Smartcards, and Smartdisks. Such devices perform the processor-intensive digital signature generation and verification operations, and may provide better protection for a user's private key than can systems which generate digital signatures in software.

4. Protocol

The authentication exchange used in this protocol is defined by ISO 9798-3 [ISO9798] and the NIST Public Key Authentication Standard [PKAS]. This document specifies a particular instantiation of these standards, with a level of detail sufficient to enable the development of interoperable implementations.

Authentication is conducted between two entities, designated A and B. In the following discussion, entity A will be referred to as the client or user, and entity B will be referred to as the server. This matches the common case of a user authenticating to a host, but use of this terminology does not mean that the protocol cannot be used in other situations such as unilateral authentication of a host by a user, or machine-to-machine authentication.

```
 ----------                              ----------
|          | (optional request) ------> |          |
|          |                            |          |
|          | <--------------- challenge |          |
|    A     |                            |    B     |
|          | response ----------------> |          |
| "client" |                            | "server" |
|          | <------- response (mutual) |          |
|          |                            |          |
|          | (optional status) -------> |          |
 ----------                              ----------
```

The PKL authentication exchange consists of either two or three messages. The first two are used for unilateral authentication, in which B authenticates the identity of the A. All three messages are used for mutual authentication, in which each party authenticates the other. The messages are:

PKL1: The initial challenge from B to A
PKL2: The response from A to B in both unilateral and mutual authentication exchanges
PKL3: The response from B to A in a mutual authentication exchange

In addition, optional messages may be used before and after the authentication exchange to provide additional unauthenticated (advisory) information.

An optional authentication request message may precede the authentication exchange to indicate the client's desired protocol version, encoding, and key exchange methods. The request message also includes an opaque data field which may be used to convey application-specific information such as a user/account identifier:

{PKL0:} Authentication request from A to B

After the authentication exchange, one of the following optional status messages may sent to return status information:

{PKL3:} Status message from B to A after unilateral authentication
{PKL4:} Status message from A to B after mutual authentication

4.1 Message Encoding

This document describes a single message encoding format, although an optional field is defined below to enable the specification of additional message formats. In the default format, each message consists of a message label, one or more tagged value fields, and a null field which serves as the message terminator:

```
PKL#:tag1-value1:tag2-value2:...::
```

The fixed-format message label ("PKL#") provides a marker for the beginning of authentication protocol messages. In many applications of this protocol, the authentication messages will travel over the same channel as other application-layer data. Implementations may need to scan a continuous datastream in order to recognize and locate authentication messages before being able to process them.

Each field consists of a tag (for example RB), a dash "-", and a value. Field values are encoded using the [BASE64] mime-content-type, which enables the transfer of binary values over ASCII-only channels. The length of the value is implicitly included in the Base64 encoding. Fields are separated by colon characters ":", and the message terminator is represented by adjacent colons "::". Fields with no value consist of just the tag without a trailing "-". An actual message would look like:

```
PKL1:K1:B-cAQU5EUk:RB-nZImJjnTNHJUtX::
```

The following notation is used to describe the authentication protocol: a field name such as "Rb" refers to a full tag-value string using the tag "RB". Within the signature argument list, the field names indicate the order in which raw (unencoded) values are included in the signature calculation. Optional fields are enclosed in curly brackets “{}”; fields not identified as optional are required. If an optional field is not used in a message, the following field separator “:” is also omitted since a null field would terminate the message at that point. Alternatives are denoted by a vertical bar “|”; only one of the listed alternatives may be sent in any particular message. Fields may appear in arbitrary order within a message, and no field except those specifically designated below may appear more than once in a message. The optional authentication request message “PKL0:” may include multiple fields of the same type to allow the client to specify a list of acceptable capabilities; the fields which may appear multiple times are identified with an asterisk “*”.

The PKL protocol messages include the following fields:

```
Authent. Request:  {PKL0:{V}*:{F}*:{K}*:{U}::}
Challenge:         PKL1:K:Rb:{V}:{B}:{U}:{E}::
Response:          PKL2:Ra:Sa(Ra,Rb,{X}):{A|Ca}:{U}:{X}:{M}:{E}::
Mutual Response:   PKL3:Sb(Rb,Ra,{X}):{B|Cb}:{U}:{X}:{E}::
Unilateral Status: {PKL3:E::}
Mutual Status:     {PKL4:E::}
```

Fields are labeled using the following tags:

```
V#       Protocol version number
F#       Message encoding format
K#       Public key access method
RA,RB    Non-repeating value (nonce) generated by A or B
A,B      Entity ID used to retrieve the public key for A or B
CA#,CB#  Certificate or certificate path for entity A or B
SA,SB    Digital Signature generated by A or B
U#       Unsigned opaque data
X#       Signed opaque data
M        Mutual authentication request flag
E#       Acknowledgment or error reply code
```

The character “#” indicates an integer (one or more ASCII digits “0” - “9”) used to qualify the meaning of the tag. Tag qualifier codes are defined in the sections below; additional codes may be defined in future versions of the protocol to accommodate additional requirements.

Tags and message labels are ASCII constants and must be capitalized as shown. Whitespace is not allowed within message labels or field tags or between a tag and the following “-” character. Whitespace is allowed within values as specified in [BASE64].

4.2 Protocol Version

This document describes version 1 of the Public Key Login protocol. The optional V field indicates the protocol version being used; if not present, version 1 is assumed. Protocol evolution may be accommodated by defining additional field qualifier codes or field types without requiring a new version number to be assigned. Implementations should reject messages containing unrecognized field tags other than the allowed Opaque Data tags. If a server receives an authentication request message specifying a version that it does not support, it may either begin an authentication exchange with a supported version and including an explicit version field, or it may reject the request with an error reply code. The following version code is defined:

V1: PKL protocol version 1

4.3 Message encoding format

This optional field appearing of the authentication request message indicates the format used to encode message labels, field tags, and field data. If this field is not present, the default encoding F1 is used. The following encoding is defined:

F1: ASCII message labels and field tags as defined in this document, Base64-encoded field values

4.4 Public Key access method

This field has no associated value; the tag qualifier is used to select which messages contain entity ID or certificate fields. The entity which issues the challenge (referred to as the server) selects the key access method; client implementations must support any of the required methods. The client may request preferred method(s) using the optional authentication request message.

K0: Implicit IDs - the server already knows the claimed identity of the client based on a login ID or some other means, prior to beginning the authentication exchange. No entity IDs or Certificates are contained in the authentication messages. The server uses the claimed ID to retrieve a public key and verify the client’s signature. If the client supports multiple personalities (private keys), it must generate a signature using the personality expected by the server. This may require use of configuration options or interactive dialog with the user.

K1: The server ID is contained in message PKL1 and no entity IDs or Certificates are contained in the following messages. The client may use the server ID to select a personality (private key) for signature generation, as well as to retrieve the server’s public key for mutual authentication.

K2: The client ID is contained in message PKL2 and the server ID is contained in optional message PKL3. The IDs correspond to the personality used by the signer, and are used by the verifier to retrieve the signer’s public key or certificate.

K3: The client’s certificate is contained in message PKL2 and the server’s certificate is contained in optional message PKL3. The entity IDs are derived from the contents of the certificates (Subject Distinguished Name, or Issuer/Serial Number, etc).

K4: The client's and server's full certificate paths are contained in messages PKL2 and PKL3. In addition to requiring a large amount of data to be exchanged, this method requires the parties to agree on a path root (common point of trust). This protocol does not define any method of negotiating desired Certificate Authorities.

K5: The client's and server's full certificate paths and auxilliary information (Certificate Revocation Lists (CRL) and Compromised Key Lists (CKL)) are contained in messages PKL2 and PKL3.

The following table summarizes the ID/Certificate fields contained in each of the messages as specified by the Key Access Method, and the methods which must be supported by client implementations:

	PKL1	PKL2	PKL3	Required?
K0:	-	-	-	No
K1:	B	-	-	Yes
K2:	-	A	B	Yes
K3:	-	CA	CB	Yes
K4:	-	CApath	CBpath	No
K5:	-	CApath+	CBpath+	No

4.5 Nonces

The Ra and Rb fields are non-repeating values generated by the client and server respectively. The protocol definition does not specify the size or content of these fields because such requirements are not necessary to achieve interoperability. However, the security of the protocol depends critically on the quality of the data in these fields. See the discussion below under Security Considerations.

4.6 Identifiers and Certificates

The PKL protocol requires the use of public/private key pairs and associated parameters to generate and verify digital signatures, but does not specify how implementations are to maintain this information. In the simplest case, keys and parameters are statically configured on the client and server. This is equivalent to configuring a One Time Password system or symmetric key authentication token. Alternatively, public key certificates can be statically configured, transmitted within the PKL protocol, or retrieved from a directory server. Certificates allow one or more trusted authorities to vouch for the binding between an entity identifier (ID) and a public key. The public key is used to authenticate a signature; once authentication has succeeded, the ID is used to authorize particular actions such as logging in to a specific account. In the traditional environments for which this protocol is intended, the client ID might be a login name assigned when the user is given an account on a system; the server ID might be a host name, DNS name, or other reference by which the user establishes a connection to a particular host.

When using ID-based Key Access options (K0,K1,K2), implementations use an ID to retrieve a public key, signature algorithm parameters, and optionally any other information (validity dates, revocation lists, parent certificates, etc.) needed to assure that the binding between ID and public key is valid. Implementations may maintain a mapping between entity IDs and certificate IDs (Distinguished Names) to allow retrieval of certificates from a directory server, or they may use the certificate ID directly as the entity ID.

When using certificate-based Key Access options (K3,K4,K5), public key certificates are transmitted as part of the authentication exchange. In this case, implementations are able to verify that the binding between subject ID and public key is valid by validating all certificates in the path between a trusted party and the entity being authenticated. In addition, it may be necessary to map the certificate contents to an entity ID in order to make access control decisions.

A single certificate format is defined in this version of the protocol:

CA1,CB1 DER-encoded X.509 certificate(s). Because the DER encoding is self-delimiting, multiple certificates may simply be concatenated for use with the K4 and K5 (certificate path) Key Access options. The first certificate must be that of the entity to be authenticated. Note that certificate path validation procedures are outside the scope of this protocol.

Other public key certificate formats may be accommodated by defining additional tag qualifier codes.

4.7 Signed and Unsigned Opaque Data

The “X#” and “U#” fields represent signed and unsigned data which is not interpreted by the PKL protocol, and which can be used for application-specific purposes. The tags X:, U:, X0: - X9: and U0: - U9: are reserved; implementations of this version of the protocol must not generate reserved field tags and must fail to authenticate if a reserved field tag is received. Implementations may use tag codes greater than 9 for any purpose. The receipt of an unrecognized but non-reserved tag code should not cause an error, although this behavior is not mandatory. A registration procedure involving the IANA may be used to allocate tag codes for these fields, beginning with the reserved values.

Signed data is transmitted in the “X” field and included in the digital signature calculations. It is thus protected from modification and bound to the sender’s authenticated identity. The PKL protocol makes no use of this field except to guarantee it’s integrity and authenticity. This field might be used, for example, with a key establishment protocol to bind a session key to authenticated endpoint identities.

Unsigned data is transmitted in the “U” field but is not used in any way and is not protected from modification by the protocol. Implementations must not rely on unsigned data in any way that affects the success or failure of authentication or affects any actions based on the authenticated identity. The unsigned data field is included in the protocol solely to accommodate unforeseen applications, and should be used only after thoughtful analysis.

4.8 Signatures

Digital signatures are calculated over the fields listed in the argument list, in the specified order. For example "Sa(Ra,Rb,{X})" specifies the following processing steps:

* Concatenate the required values Ra and Rb
* Append the optional signed data X#, if present in the message
* Calculate the digital signature
* Base64-encode the signature value

To achieve interoperability, signatures must be calculated using the specified argument order. Note: there is no particular significance to the order in which nonces appear in the signed data - (Ra,Rb) in message PKL2 vs. (Rb,Ra) in message PKL3; it is defined this way solely for consistency with [ISO9798].

The algorithm used to generate and verify digital signatures is specified in the entity's public key certificate, or in equivalent configuration data if certificates are not used. The value of the signature fields "SA" and "SB" depend on the algorithm used:

Digital Signature Standard: The [DSS] signature of a message is a pair of bit strings designated R and S. The value of the signature field is R followed by (concatenated with) S. The DSS public key (community parameters P, Q, and G, and user-specific public value Y) may be extracted from a certificate, manually configured, or obtained by other methods outside the scope of this protocol.

RSA Signature: The RSA signature of a message is a single bit string designated S, which is the value of the signature field. The RSA public key (modulus N and public exponent E) may be extracted from a certificate, manually configured, or obtained by other means.

4.9 Mutual Authentication Flag

The optional "M" field in message PKL2 allows the client to request mutual authentication. If it is present, the server must generate message PKL3 to allow the client to authenticate the server. If the "M" field is not present, the server must not send message PKL3. Note that there is no reason to allow the server to request mutual authentication, since the server cannot force a client to authenticate any information sent by the server. This field has no tag qualifier code or associated value.

Implementations of the PKL protocol should support both unilateral and mutual authentication.

4.10 Reply Codes

This optional field has no associated value; the tag qualifier contains the acknowledgment or error status code. This field is not sent by default, but if a reply code field is sent in one authentication message, all following messages in the current exchange must include reply codes. In this manner either client or server can request that reply codes be used.

In environments where the authentication exchange is visible to the user, it is generally obvious that an authentication request has either succeeded or failed. Additional information can be communicated in free form outside the authentication protocol, and reply codes may not be needed. In other environments, such as modem dialer chat scripts, it may be useful to transmit status information in machine-readable form using this field.

The following reply codes are defined, based on the three digit format defined in [FTP]. The first digit is a “2” for success and “5” for failure. The second digit is a “0” for protocol syntax status and a “3” for authentication status. The third digit gives additional detail:

- E200: NOP. This code is sent by either the client or server to request that reply codes be used in the following messages, if no other status is applicable to the current message.
- E230: Authentication succeeded.
- E500: Syntax error - unrecognized or invalid field tag.
- E501: Base64 value decoding error.
- E502: Requested protocol version “V#” not supported.
- E503: Requested message encoding “F#” not supported.
- E504: Requested key access method “K#” not supported.
- E505: Mutual authentication not supported.
- E530: Signature verification failure.
- E531: Invalid entity ID - public key could not be retrieved.
- E532: Invalid public key certificate - many possible causes, including expired or revoked certificate, path validation failed, unknown or unsupported certificate format, etc.

5. Registration

The PKL protocol allows certificates to be either passed within the authentication exchange or referenced by an entity ID. The latter case generates considerably less traffic during authentication, but requires that identifiers be assigned to certificates or public keys before the authentication protocol can be used. This section specifies a registration protocol by which entity IDs can be assigned to certificates.

The registration protocol consists of two messages, which may be exchanged offline (via email, file transfer, floppy disk, etc.). The messages are protected from modification by a secret key, which should be sent separately from the registration messages. The registration messages contain entity IDs, Certificates, and a signature field to provide integrity protection. They are encoded using the same format as the authentication messages:

```
B->A    PKR1:A:CB:SB(K,A,CB)::
A->B    PKR2:B:A:CA:SA(K,B,A,CA)::
```

Registration consists of the following steps:

* One of the parties (either A or B) generates a registration key K and sends it to the other party. It is recommended that K be sent over a different channel than the registration messages, for example by telephone, fax, mail, or courier service. Implementations must allow K to be specified in hexadecimal format, and should also allow K to be entered in a more user-friendly "pass-phrase" format. If a pass-phrase is used, it must be canonicalized by removing all leading and trailing whitespace, converting any imbedded whitespace to single space characters, and translating all lower case characters to upper case.
* The "server" B generates an entity ID A by which it will refer to the user, constructs message PKR1 containing A, the server's certificate path CB, and the signature field SB. B then transmits the message to A. B computes the signature field by concatenating the registration key K, entity ID A, and certificate path CB, and signing the result.
* The "user" A generates an entity ID B by which it will refer to the server. The client implementation reads message PKR1, prompts the user for K, verifies the signature SB, and permanently registers the mapping between entity ID B and the server's certificate CB.
* A constructs message PKR2 in a manner similar to the construction of PKR1 and transmits the message to B.
* The server implementation reads message PKR2, prompts the operator for K, verifies the signature SA, and permanently registers the mapping between A and CA and the association between A and B. Note that message PKR2 contains both entity IDs A and B; the server maintains this association so that when it receives an authentication request from A, it can include its own (user-specific) ID B in the challenge.

The registration protocol is somewhat complicated by the conflicting requirements of keeping entity IDs short and mnemonic while maintaining uniqueness. If fully-qualified IDs (Distinguished Names, DNS/email names, etc. plus a namespace identifier) were mandated, the registration protocol could be slightly simplified, but implementations might still need to maintain an association between fully-qualified IDs and system-specific identifiers such as usernames.

6. Examples

A minimal unilateral authentication exchange might look like:

```
B->A    PKL1:K1:B-cAQU5EUk:RB-nZImJjnTNHJUtX::
A->B    PKL2:RA-Asho84kLN3/IJm:SA-Wiy6IesKvjL5rL9WjXUb9BkAmw=::
```

A mutual authentication exchange with status codes might look like:

```
B->A    PKL1:K1:B-cAQU5EUk:RB-nZImJjnTNHJUtX::
A->B    PKL2:RA-Asho84kLN3/IJm:SA-Wiy6IesKvjL5rL9WjXUb9BkAmw=:E200::
B->A    PKL3:SB-6IesKvjL5rL9WjXUb9MwT9bpObY=:E230::
A->B    PKL4:E230::
```

If certificates are exchanged, the messages are considerably longer.

7. Implementation Experience

A protocol similar to this one, the “Mosaic Authentication Protocol” (MAP), has been in use since April 1994 to provide network and dial-up access to computers at NSA, and has been commercialized in the LOCKout(TM) product from Secure Computing Corporation. MAP was also the basis for HTTP-Fortezza, a prototype demonstration of strong access control for Web applications.

The PKL protocol is a generalization of MAP, designed to comply with the NIST Public Key Authentication Standard [PKAS] and to address specific problems identified during operational use of MAP. No implementations of PKL currently exist, but all Government applications of MAP are expected to migrate to PKL by the effective date of the NIST Authentication Standard (currently TBD).

8. Security Considerations

8.1 Authentication, Integrity, and Confidentiality

The primary limitation of authentication-only protocols, including this one, is that they provide no protection to the data exchanged once a session has been established. Authentication provides no protection against eavesdropping, modification, or session hijacking. In cases where comprehensive protection methods ([IPSEC], [TLS], etc.) are available and the need for data protection outweighs the performance impact, a comprehensive method should be used in lieu of an authentication-only method. In some situations it may be useful to combine an authentication-only protocol with other methods, for example to provide user authentication at the application layer over Virtual Private Networks using IPSEC at the network layer.

8.2 Challenge Response vs. Time

The Public Key Login protocol uses an authentication method based on signed challenges. This avoids the following problems faced by public key authentication methods (for example, [X.509]) based on signed time fields:

* Clocks must be synchronized. This may be difficult to achieve in some situations.
* Even environments which provide tightly-controlled clocks (using e.g. NTP), the security of the authentication protocol depends on the secu-

rity of the time synchronization method. A protocol which does not have this critical reliance on an external service is preferable.
* Even in environments which provide synchronized clocks which are resistant to failure or attack, time-based authentication methods must allow a window of acceptance to accommodate communication delays. Eliminating vulnerability to replay attacks within this window requires additional complication, particularly when multiple servers are involved.

In contrast, challenge-response authentication is only subject to replay attacks if a challenge is repeated. This is easily prevented, as addressed in the following section.

8.3 Nonces

The ability of challenge-response protocols to protect against replay attacks relies on the assumption that nonces will not be repeated during the lifetime of the digital signature keys with which they are used. Random numbers are often used as nonces. However, a number chosen at random will have a small but non-zero probability of taking any particular value. The probability of repeating one of the previously-used nonces increases with continuing use of the authentication protocol with a particular signature key. Nonces containing a deterministic component such as time or a sequence number have zero probability of repeating as long as the deterministic component is reliable, but achieving the required level of reliability is problematic. Accordingly, it is recommended that implementations of this protocol construct nonces by concatenating a random value with a long-term non-repeating value such as time. Note that this application of a time value does not require that clocks be synchronized either with real-world time or between the authenticating parties. If a single user authenticates to multiple servers and the servers' clocks are not synchronized, then any time component of a nonce does not contribute to replay protection for the server with the later clock value. Therefore, implementations of this protocol should construct nonces with a random component sufficiently large to provide adequate protection without relying on a non-repeating component.

8.4 Identities and Authorization

Note that implementations must take care to ensure that access control or other actions are based on the identity which has been authenticated, not on claimed or otherwise unauthenticated identities. This is particularly important when certificates are contained in the protocol messages, because the mapping from certificate Distinguished Name to username may be more difficult than mapping in the opposite direction, from username to certificate. For example, a user attempting to login with username "Bob" could present a valid certificate for user "Fred" and a signature generated with Fred's private key. After successfully verifying the signature, the server must not log the user on to Bob's account.

9. References

[BASE64] Internet RFC 1521, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", Section 5.2, "Base64 Content-Transfer-Encoding", Borenstein & Freed, September 1993, http://ds.internic.net/rfc/rfc1521.txt

[DSS] NIST FIPS Pub 186, "Digital Signature Standard (DSS)", 19 May 1994, http://csrc.ncsl.nist.gov/fips/fips186.txt

[FTP] Internet RFC 959, "File Transfer Protocol", Section 4.2, "FTP Replies", Postel & Reynolds, October 1985, http://ds.internic.net/rfc/rfc959.txt

[ISO9798] ISO/IEC 9798-3, "Information technology - Security techniques - Entity authentication mechanisms - Part 3: Entity authentication using a public key algorithm", Section 5.1.2 "Two pass unilateral authentication" and Section 5.2.2 "Three pass mutual authentication", 1993

[OTP] Internet-Draft (work-in-progress), "A One-Time Password System", 21 November 1995, ftp://ftp.ietf.org/internet-drafts/draft-haller-otp-05.txt, see also RFC 1760.

[PKAS] NIST FIPS Pub JJJ, "Entity Authentication using Public Key Technology", Draft 29 March 1996 http://csrc.ncsl.nist.gov/fips/pkauth.txt

[PAM] OSF RFC 86.0, "Unified Login with Pluggable Authentication Modules (PAM)", October 1995 http://www.pilgrim.umass.edu/pub/osf_dce/RFC/rfc86.0.txt

[RADIUS] Internet-Draft (work-in-progress), "Remote Authentication Dial In User Service (RADIUS)", November 1995, ftp://ftp.ietf.org/internet-drafts/draft-rigney-radius-01.txt

[SHA] NIST FIPS Pub 180-1, "Secure Hash Standard", 31 May 1994, http://csrc.ncsl.nist.gov/fips/fip180-1.txt

[X.509] ISO 9594-8 / ITU X.509, "The Directory - Authentication Framework", (1988 version: http://sunsite.doc.ic.ac.uk/computing/ccitt/ccitt-standards/1992/X/x509_1.asc.gz)

10. Acknowledgements

Thanks to Hugh Harney of SPARTA, Inc. and Dave Balenson of Trusted Information Systems for their reviews of this draft.

11. Author's Address

David P. Kemp
National Security Agency, X31
9800 Savage Road
Fort Meade, MD 20755
USA

Email: dpkemp@missi.ncsc.mil