

INTERNET-DRAFT

<[draft-ietf-cat-kerberos-passwords-04.txt](#)>

Updates: RFC [1510](#)

November 18, 1998

Clifford Neuman

ISI

Glen Zorn

Microsoft Corporation

Jonathan Trostle

CyberSAFE Corporation

Ken Hornstein

ITT/NRL

Integrating Single-use Authentication Mechanisms with Kerberos

0. Status Of this Memo

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

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

To learn the current status of any Internet-Draft, please check the ``1id-abstracts.txt'' listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

The distribution of this memo is unlimited. It is filed as <[draft-ietf-cat-kerberos-passwords-04.txt](#)>, and expires May 11, 1999. Please send comments to the authors.

1. Abstract

This document defines extensions to the Kerberos protocol specification [[RFC1510](#)] which provide a method by which a variety of single-use authentication mechanisms may be supported within the protocol. The method defined specifies a standard fashion in which the preauthentication data and error data fields in Kerberos messages may be used to support single-use authentication mechanisms.

2. Terminology

To simplify the following discussion, we will define those terms which may be unfamiliar to the audience or specific to the discussion itself.

Single-use Preauthentication Data (SPD): Data sent in the padata-

value field of a Kerberos V5 message proving that knowledge of certain unique information is held by a principal. This information

may or may not be identical to the single-use authentication data input to the client. For example, in the case of S/Key, the principal might input a one-time password (in any of several forms); the knowledge of this one-time password is taken to indicate knowledge of the principal's secret passphrase. Similarly, the SPD may or may not contain the provided single-use authentication data. For instance, if a given single-use authentication mechanism includes a token which generates an encryption key for a supported cryptosystem, that key could be used to encrypt portions of the SPD before transmission. As long as the verification process of the mechanism was capable of independently generating the same key, the successful decryption of the SPD would provide assurance that the originator of the message was in possession of the token, as well as whatever information the token required to generate the encryption key.

Single-use Authentication Mechanism (SAM): A system for generating and verifying authentication data which is usable only once.

Single-use Authentication Data (SAD): SAM-specific data provided by a principal as input to client software to be used in the creation of SPD.

3. Motivation and Scope

Several single-use authentication mechanisms are currently in widespread use, including hardware-based schemes from vendors such as Enigma Logic, Digital Pathways and Security Dynamics and software-based methods like S/Key [[RFC1760](#)]. The hardware-based schemes typically require that the authenticating user carry a small, credit-card-sized electronic device (called a token) which is used to generate unique authentication data. Some tokens require the user to enter data into the device. This input may take the form of a Personal Identification Number (PIN), a server-generated challenge string or both. Other tokens do not use a challenge-response technique, instead spontaneously generating new and unique authentication data every few seconds. These tokens are usually time-synchronized with a server. The use of one-time passwords and token cards as an authentication mechanism has steadily increased over the past few years; in addition, the Internet Architecture Board has encouraged the use of SAMs to improve Internet security [[RFC1636](#)].

The widespread acceptance of Kerberos within the Internet community has produced considerable demand for the integration of SAM technology with the authentication protocol. Several currently available implementations of Kerberos include support for some types of token cards, but the implementations are either not interoperable,

or would require the release of source code (not always an option) to make them interoperate. This memo attempts to remedy that problem by specifying a method in which SAM data may be securely transported in Kerberos V5 messages in a standard, extensible fashion. This document does not, however, attempt to precisely specify either the generation or verification of SAM data, since this is

likely to be SAM-specific; nor does it dictate the conditions under which SAM data must be included in Kerberos messages, since we consider this to be a matter of local policy.

4. Generic Approach - Two Models

As outlined above, there are essentially two types of single-use authentication mechanisms: challenge/response and time-based. In order to support challenge/response mechanisms, the Kerberos Key Distribution Center (KDC) must communicate the appropriate challenge string to the user, via the client software. Furthermore, some challenge/response mechanisms require tight synchronization between all instances of the KDC and the client. One example is S/Key and its variants. If the KDC and client do not perform the same number of message digest iterations, the protocol will fail; worse, it might be possible for an eavesdropping attacker to capture a valid S/Key passcode and replay it to a KDC replica which had an outdated iteration number. In the time-based case, no challenge is required. This naturally gives rise to two modes of client behavior, described below.

4.1 Challenge/Response Model

The client begins with an initial **KRB_AS_REQ** message to the KDC, possibly using existing preauthentication methods (PA-ENC-TIMESTAMP (encrypted timestamp), PA-OSF-DCE (DCE), etc.). Depending on whether preauthentication is used, the user may or may not be prompted at this time for a Kerberos password. If (for example) encrypted timestamp preauthentication is used, then the user will be prompted; on the other hand, if no preauthentication is in use the prompt for the password may be deferred (possibly forever). Note that the use of preauthentication here may allow an offline guessing attack against the Kerberos password separate from the SPD. However, if the use of a SAM is required, then the password by itself is not sufficient for authentication. (Specify character strings as UTF-8)

The KDC will determine in an implementation- and policy-dependent fashion if the client is required to utilize a single-use authentication mechanism. For example, the implementation may use IP address screening to require principals authenticating from outside a firewall to use a SAM, while principals on the inside need not. If SAM usage is required, then the KDC will respond with a **KRB_ERROR** message, with the error-code field set to **KDC_ERR_PREAUTH_REQUIRED** and the e-data field containing the ASN.1 structure that is a sequence of PA-DATA fields.

If the type of one of the PA-DATA fields is **PA-SAM-REDIRECT**, the

client should re-execute the authentication protocol from the beginning, directing messages to another of the KDCs for the realm. This is done to allow some methods to require that a single KDC be used for SAM authentication when tight synchronization is needed between all replicas and the KDC database propagation code does not provide such synchronization. The corresponding padata-value will

contain an encoded sequence of host addresses [[RFC1510](#)], from which the client must choose the KDC to be contacted next.

Otherwise, if one of the PA-DATA fields has the type PA-SAM-CHALLENGE, the exchange will continue as described in [section 5](#), below.

[4.2](#) Time-based Model

For mechanisms where no challenge is required, the user (or the client software being utilized) may or may not know a priori whether SAM usage is required. If it does not know, then the initial exchange may proceed as above. If it is known that a use of a single-use authentication mechanism is required then the first exchange can be skipped and the authentication will continue as follows.

[5](#). Authentication Using Passcodes

Prior to performing preauthentication using a single-use authentication mechanism, the client must know whether a challenge is required (if the client doesn't have this information prior to its sending the first KRB_AS_REQ message, it will be informed of the requirement by the KDC, as described in section 4.1). The client does NOT need to know the specific type of SAM in use. If a challenge is required the client will be sent the challenge by the KDC. This means that a client supporting SAMs will be able to work with new methods without modification. The challenge, as well as all other prompts mentioned herein, can be internationalized by the KDC on a per-principal basis.

If a KRB_ERROR message is received from the KDC indicating that SAM usage is required, that message will include in its e-data field a PA-DATA structure that encodes information about the SAM to be used. This includes whether a challenge is required, and if so, the challenge itself; and informational data about the type of SAM that is in use, and how to prompt for the SAD. The SAM type is informational only and does not affect the behavior of the client. The prompt is also informational and may be presented to the user by the client, or it may be safely ignored.

The ASN.1 definition for the SAM challenge is:

```

PA-SAM-CHALLENGE ::= SEQUENCE {
    sam-type[0]                INTEGER,
    sam-flags[1]               SAMFlags,
    sam-type-name[2]           GeneralString OPTIONAL,
    sam-track-id[3]            GeneralString OPTIONAL,
    sam-challenge-label[4]     GeneralString OPTIONAL,
    sam-challenge[5]           GeneralString OPTIONAL,
    sam-response-prompt[6]     GeneralString OPTIONAL,
    sam-pk-for-sad[7]          EncryptionKey OPTIONAL,
    sam-nonce[8]               INTEGER OPTIONAL,
    sam-cksum[9]               Checksum OPTIONAL
}

SAMFlags ::= BIT STRING {
    use-sad-as-key[0],
    send-encrypted-sad[1],
    must-pk-encrypt-sad[2]
}

```

The sam-type field is informational only, but it must be specified and sam-type values must be registered with the IANA.

Initially defined values of the sam-type codes are:

PA_SAM_TYPE_ENIGMA	1	-- Enigma Logic
PA_SAM_TYPE_DIGI_PATH	2	-- Digital Pathways
PA_SAM_TYPE_SKEY_K0	3	-- S/key where KDC has key 0
PA_SAM_TYPE_SKEY	4	-- Traditional S/Key
PA_SAM_TYPE_SECURID	5	-- Security Dynamics
PA_SAM_TYPE_CRYPTOCARD	6	-- CRYPTOCARD

PA_SAM_TYPE_SECURID, PA_SAM_TYPE_DIGI_PATH, PA_SAM_TYPE_ENIGMA, and PA_SAM_TYPE_CRYPTOCARD represent popular token cards. PA_SAM_TYPE_SKEY is the traditional S/Key protocol, in which the SAD verifier does not have knowledge of the principal's S/Key secret. PA_SAM_TYPE_SKEY_K0 is a variant of S/Key that uses the same SAD and PC software or hardware device, but where the zeroth key (the S/Key secret) is actually stored on, and can be used by, the SAD verifier to independently generate the correct authentication data.

Note that using PA_SAM_TYPE_SKEY_K0 gives up one advantage of S/Key, viz., that the information required to generate the SAD need not be stored on the host; but since the SAD verifier (which may be the KDC) is assumed to be more secure than other hosts on the network, it may be acceptable to give up this advantage in some situa-

tions. The advantage of using this S/Key variant is that the security of the network protocol is strengthened since the SAD need not be sent from the client to the KDC. Thus, the SAD can be used as part of the key used to encrypt the encrypted parts of both the SPD and the KRB_AS_REP message, rather than being sent protected by the principal's Kerberos secret key which may have been previously

exposed to an attacker (see [section 6](#), below). In any case, there is a definite advantage to being interoperable with the S/Key algorithm.

Due to the volatility of, and rapid developments in, the area of single-use authentication mechanisms (both software-only and hardware supported), any subsequently defined sam-type codes will be maintained by the IANA.

The sam-flags field indicates whether the SAD is known by the KDC (in which case it can be used as part of the encryption key for the ensuing KRB_AS_REP message), or if it must be provided to the KDC in a recoverable manner. If it is known to the KDC, use-sad-as-key indicates that the SAD alone will be used to generate the encryption key for the forthcoming KRB_AS_REQ and KRB_AS_REP messages, and that the user will not need to also enter a password. We recommend that this option not be used, and that the principal also enter the Kerberos password, but for some situations, presentation of the SAD alone may be acceptable. If the single-use authentication data is not known (and cannot be generated or discovered) by the KDC, then send-encrypted-sad flag will be set, indicating that the SAD must be sent to the KDC encrypted under the principal's secret key. If neither use-sad-as-key nor send-encrypted-sad are set, the client may assume that the KDC knows the SAD, but the Kerberos password should be used along with the passcode in the derivation of the encryption key (see below). The must-pk-encrypt-sad flag is reserved for future use. If this flag is set and a client does not support the must-pk-encrypt-sad option (to be defined in a separate document), the client will not be able to complete the authentication and must notify the user.

Note that there are specific constraints on the integrity of the PA-SAM-CHALLENGE when some of these options are specified. In particular, if any of these flags are set, a cryptographic checksum must be present and verified. If absent, or the checksum does not match the request, the challenge must be considered invalid and the user notified.

The optional sam-type-name field is a UTF-8 character string for informational use only. It may be used by the client to display a short description of the type of single-use authentication mechanism to be used.

The optional sam-track-id field may be returned by the KDC in the KRB_ERROR message. If present, the client should copy this field into the corresponding field of the SAM response sent in the subsequent KRB_AS_REQ message. This field may be used by the KDC to match challenges and responses. It might be a suitably encoded

integer, or even be encrypted data with the KDC state encoded so that the KDC doesn't have to maintain the state internally.

The sam-challenge-label field is informational and optional. If it is included, it will be an UTF-8 encoded character. If present, a client may choose to precede the presentation of the challenge with

this string. For example, if the challenge is 135773 and the string in the sam-challenge-label field is "Enter the following number on your card", the client may choose to display to the user:

Enter the following number on your card: 135773

If no challenge label was presented, or if the client chooses to ignore it, the client might display instead:

Challenge from authentication server: 135773

If the sam-checksum field is empty the sam-challenge-label should be ignored, since in that case it may have been modified by an adversary. Internationalization is supported by allowing customization of the challenge label and other strings on a per-principal basis. Note that this character string should be encoded using UTF-8.

The optional sam-challenge field contains a string that will be needed by the user to generate a suitable response. If the sam-challenge field is left out, it indicates that the SAM in use does not require a challenge, and that the authorized user should be able to produce the correct SAD without one. If the sam-challenge field is present, it is the data that is used by the SAD generator to create the SAD to be used in the production of the SPD to be included in the response.

The sam-response-prompt field is informational and optional. If present, a client may choose to precede the prompt for the response with the specified string; however, it should not be used if the sam-checksum field is empty, since in that case the prompt may have been modified by an adversary. If this field is empty the prompt displayed will be hardcoded into the application, such as

Passcode:

sam-pk-for-sad is an optional field. It is included in the interest of future extensability of the protocol to the use of public-key cryptography.

The sam-nonce field is optional. If present, it should conform to the specification of the nonce field in a KRB_KDC_REQ message [[RFC1510](#)].

The optional sam-cksum field contains a cryptographic checksum of the preceding fields, protected using the same key as that used for preauthentication in the initial KRB_AS_REQ message. While any secure checksum method may be used, the RSA-MD5-DES type is recommended and must be supported by the client. This field should

always be present if the initial KRB_AS_REQ message included a form of cryptographic preauthentication (such as PA_ENC_TIMESTAMP). If the initial message included preauthentication, but this field is missing in the response, the client should reject the response. If this field is present, the client should verify its correctness:

if the sam-cksum field is present in the KRB_ERROR message, but not verified by the client, then an attacker can change the message contents at will. Such a change could cause unexpected instructions to be displayed to the user in the prompt string, or allow the denial of service through the presentation of an invalid challenge. Another effect of a change might be to effect a limited chosen plaintext attack on the Kerberos password where the attacker picks the nonce, and the challenge.

Note that it is possible for the KDC to generate a cryptographic checksum for this message using the users' secret key even though no cryptographic preauthentication was present in the initial AS_REQ message. Doing so, however, makes material available which enables an offline guessing attack against that key.

If the client is performing SAM preauthentication in the initial message, without receipt of a PA-SAM-CHALLENGE (i.e. without waiting for the KRB_ERROR message), and the SAM in use does not require a challenge, the client will prompt for the SAD in an application-specific manner.

Once the user has been prompted for and entered the SAD (and possibly the Kerberos password), the client will derive a key to be used to encrypt the preauthentication data for a KRB_AS_REQ message. This key will be determined as follows:

By default, the key is derived from the password and the SAD by running each through the string_to_key function [[RFC1510](#)] separately; i.e., $K1 = \text{string_to_key}(\text{password})$ and $K2 = \text{string_to_key}(\text{SAD})$. The $K1$ and $K2$ are then combined to form a single key K in a manner specific to the type of key being generated. For DES keys, K will be formed by XORing $K1$ and $K2$, then correcting the parity of K . If the result matches a "weak" or "semiweak" key as described in the DES specification, it is XORed with the hexadecimal constant 0000000000000000F0.

If the send-encrypted-sad flag is set, the key will be derived by running the Kerberos password through the string_to_key function in the normal fashion.

If the use-sad-as-key flag is set and the integrity of the PA-SAM-CHALLENGE PADATA field can be verified using the sam-cksum field, then the SAD is run through the string_to_key function and the result is used as the encryption key for the request. WARNING: the use of single-use authentication data in this manner is NOT recommended unless the range of the SAD is large enough to make an exhaustive

off-line search impractical and the risks involved in the use of SAD alone are fully considered. Also, note that without the availability to the KDC of a relatively static, unique secret key shared with the user, the only mechanisms that can be used to protect the integrity of the PA-SAM-CHALLENGE PADATA field are based on either public key

cryptography or the KDC's a priori knowledge of the SAD itself. In the latter case, the client must obtain the SAD from the user and use it to verify the integrity of the challenge before the new KRB_AS_REQ message is sent.

The sam-pk-for-sad field is reserved for future use. If this field is not empty and the client does not support the use of public-key encryption for SAD (to be defined in a separate document), the client will not be able to complete the authentication and must notify the user.

The client will then send another KRB_AS_REQ message to the KDC, but with a padata field with padata-type equal to PA-SAM-RESPONSE and padata-value defined as follows:

```

PA-SAM-RESPONSE ::= SEQUENCE {
    sam-type[0]                INTEGER,
    sam-flags[1]                SAMFlags,
    sam-track-id[2]             GeneralString OPTIONAL,
    sam-enc-key[3]              EncryptedData
                                -- PA-ENC-SAM-KEY,
    sam-enc-nonce-or-ts[4]      EncryptedData
                                -- PA-ENC-SAM-RESPONSE-ENC,
    sam-nonce[5]                INTEGER OPTIONAL,
    sam-patimestamp[6]          KerberosTime OPTIONAL
}

PA-ENC-SAM-KEY ::= SEQUENCE {
    sam-key[0]                  EncryptionKey
}

PA-ENC-SAM-RESPONSE-ENC ::= SEQUENCE {
    sam-nonce[0]                INTEGER OPTIONAL,
    sam-timestamp[1]            KerberosTime OPTIONAL,
    sam-usec[2]                 INTEGER OPTIONAL,
    sam-sad[3]                  GeneralString OPTIONAL
}

```

The source of the data included in the PA-SAM-RESPONSE structure depends upon whether or not a KRB_ERROR message was received by the client from the KDC.

If an error reply was received, the sam-type and sam-flags fields will contain copies of the same fields from the error message. If the sam-nonce field was present in the KRB_ERROR message, then the PA-ENC-SAM-RESPONSE-ENC structure returned to the KDC will include that value, encrypted as described above, the nonce field in the KRB_AS_REQ message will contain a plaintext copy of the same value

and the sam-timestamp field in the PA-ENC-SAM-RESPONSE-ENC structure will be empty. If a sam-timestamp is present in the PA-ENC-SAM-RESPONSE, the sam-patimestamp field must be supplied in the PA-SAM-RESPONSE. If a KRB_ERROR message was received but the sam-nonce field was omitted from it, the encrypted data will contain a timestamp encrypted in the same manner, the sam-patimestamp field

will contain an unencrypted copy of the same value, and the sam-nonce field will be empty. In either case, if the send-encrypted-sad flag is set in the sam-flags field of the error message, the sam-sad field must contain the entered single-use authentication data.

If no error reply was received (i.e., the client knows that a single-use authentication mechanism is to be used), the sam-type field must be set to a value chosen from the list of registered sam-type codes.

The value of the sam-flags field may vary depending upon the type of SAM in use, but in all cases the must-pk-encrypt-sad flag must be zero. If the send-encrypted-sad flag is set, the sam-sad field must contain the entered single-use authentication data.

Note that if there is no sam-track-id in the request, it should be omitted in the response.

The sam-enc-key field is reserved for future use.

Upon receipt the KDC validates this PADATA in much the same way that it validates the PA-ENC-TS preauthentication method except that it uses the SAD (if available, and possibly in conjunction with saved state information or portions of the preauthentication data) to determine the correct key(s) required to verify the encrypted data. Note that if the KDC uses the sam-track-id field to encode its state, the KDC is responsible for including information in that field to detect modification or replay by an attacker.

The rest of the processing of the request proceeds normally, except that instead of being encrypted in the user's secret key, the KRB_AS_REP message is encrypted in the key obtained above. Note, however, that some single-use authentication mechanisms may require further KRB_AS_REQ/KRB_ERROR exchanges to complete authentication; for example, in order to allow the server to resynchronize with the drifting clock on a time-based token card. In these cases the KDC may respond with another KRB_ERROR message containing a different sam-type value, along with appropriate prompts and/or challenges. This sequence of exchanges will continue until authentication either succeeds or fails.

6. Security considerations

Single-use authentication mechanisms requiring the use of the send-encrypted-sad option are discouraged as their use on the network is less secure than the case where a combination of the user's password and SAD is used as the encryption key. In particular, when the send-encrypted-sad option is used, an attacker who

observes the response and is in possession of the users' secret key (which doesn't change from login to login) can use the key to decrypt the response and obtain the single-use authentication data. This is dependent on the SAM technology used.

If the KDC sets the must-pk-encrypt-sad flag of the sam-flags field but the client software being used does not support public-key cryptography, it is possible that legitimate users may be denied service.

An attacker in possession of the users encryption key (again, which doesn't change from login to login) might be able to generate/modify a SAM challenge and attach the appropriate checksum. This affects the security of both the send-encrypted-sad option and the must-pk-encrypt-sad option.

7. Expiration

This Internet-Draft expires on February 11, 1999.

8. References

[RFC1510]

The Kerberos Network Authentication System; Kohl and Neuman; September 1993.

[RFC1760]

The S/Key One-Time Password System; Haller; February 1995

[RFC1636]

Report of IAB Workshop on Security in the Internet Architecture; Braden, Clark, Crocker and Huitema; June 1994

9. Authors' Addresses

B. Clifford Neuman

USC/Information Sciences Institute
4676 Admiralty Way #1001
Marina del Rey, CA 90292-6695

Phone: 310-822-1511
EMail: bcn@isi.edu

Glen Zorn
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 206-703-1559
EMail: glennz@microsoft.com

Ken Hornstein
ITT/US Naval Research Lab
4555 Overlook Avenue
Washington, DC 20375

Phone: 202-404-4765
EMail: kenh@cmf.nrl.navy.mil

