Network Working Group
Internet-Draft
Updates: 4121 (if approved)
Intended status: Standards Track
Expires: May 25, 2015

N. Williams
Cryptonector
R. Dowdeswell
Dowdeswell Security Architecture

November 21, 2014

Negotiation of Extra Security Context Tokens for Kerberos V5 Generic Security Services Mechanism
draft-williams-kitten-krb5-extra-rt-04

Abstract

This Internet-Draft proposes an extension to the Kerberos V5 security mechanism for the Generic Security Services Application Programming Interface (GSS-API) for using extra security context tokens in order to recover from certain errors. Other benefits include: user-to-user authentication, authenticated errors, replay cache avoidance, and others.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on May 25, 2015.

Copyright Notice

Table of Contents

1. Introduction

The Kerberos V5 [RFC4120] AP protocol, and therefore the Kerberos V5 GSS-API [RFC2743] mechanism [RFC4121] security context token exchange, is a one-round trip protocol. Occasionally there are errors that the protocol could recover from by using an additional round trip, but until now there was no way to execute such an additional round trip. For many application protocols the failure of the Kerberos AP protocol is fatal, requiring closing TCP connections and starting over; often there is no automatic recovery.

This document proposes a negotiation of additional security context tokens for automatic recovery from certain errors. This is done in a backwards-compatible way, thus retaining the existing mechanism OID for the Kerberos V5 GSS mechanism. This also enables other new features.

New features enabled by this extension include:

- error recovery (see Section 5)
- user-to-user authentication (see Section 7)
- some authenticated errors (see Section 5.1)
- replay cache avoidance (see Section 6)
- acceptor clock skew correction (see Section 8)
- symmetric authorization data flows

No new interfaces are needed for GSS-API applications to use the features added in this document.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. New Protocol Elements

We introduce the following new protocol elements. A partial ASN.1 [CCITT.X680.2002] module (for inclusion in the base Kerberos ASN.1 module) is given in Section 4, and references to its contents are made below.

- a new ap-options flag for use in the clear-text part of AP-REQs to indicate the desire for an extra round trip if need be;
- a new authorization data (AD) element for integrity protection of ap-options;
- a new AD element for use in Authenticators for quoting back a challenge from the acceptor;
- a new PDU: KRB-ERROR2, also known as AP-REP2, with additional fields and support for integrity- (and confidentiality-)protected errors and optional _key confirmation_ :
  - a flag is used to indicate which key is used to encrypt the KRB-ERROR2's private part, as in some cases there can be two keys to choose from;
  - when no key available for encrypting the private part of a KRB-ERROR2, the null enctype is used.

These elements are used to construct security context token exchanges with potentially more than two context tokens.

All context tokens are to be prefixed with the InitialContextToken pseudo-ASN.1/DER header from RFC2743, section 3.1, just as RFCs 1964 and 4121 require of the first two context tokens.

2.1. Fields of KRB-ERROR2

The new KRB-ERROR2 PDU is defined in Section 4. The fields of the KRB-ERROR2 encrypted part have the following purpose/semantics:

continue-challenge A challenge to be quoted back in any subsequent context tokens.

stime The acceptor's current time.

susec Microsecond portion of the acceptor's current time.

subkey The acceptor's sub-session key. This MUST be absent when the KRB-ERROR2 enc-part is "encrypted" in the null enctype and key or when the acceptor failed to decrypt the initiator's Authenticator (but, obviously, succeeded at decrypting the Ticket); otherwise it MUST be present.

seq-number The acceptor's initial per-message token sequence number. This MUST be absent when the subkey is absent; otherwise it MUST be present.

error-code When zero-valued, the KRB-ERROR2 is not an error token, but a key-confirmation that requires continuation with an additional AP-REQ.

e-flags Indicates whether the KRB-ERROR2 is final (error token) or not.

e-text A human-readable string (in any language and script) description of the error, if any.

e-data Currently unused but specified for extensibility reasons. SHOULD be absent and MUST be ignored.

e-typed-data TYPED-DATA; see [RFC4120]. Currently unused but specified for extensibility reasons. SHOULD be absent and MUST be ignored.

your-addresses The initiator's network address(es) as seen on the acceptor side. Currently unused due to insufficient GSS-API interfaces, but specified for extensibility reasons. SHOULD be absent, MUST be ignored.

ad-data Authorization-data. This is intended for symmetry, so that acceptors can assert authorization data to the initiator just as the initiator can assert authorization data to the acceptor. (For example, this might be useful in user-to-user authentication.) When present this has the same semantics as in the AP-REQ's Authenticator, but in the opposite direction.

tgt A TGT for use in user-to-user authentication.

2.2. Distinction between KRB-ERROR2 and AP-REP2 PDUs

The ASN.1 does not distinguish between KRB-ERROR2 and AP-REP2 PDUs. A KRB-ERROR2 can serve either or both, the purpose of conveying error information, as well as the purpose of completing the acceptor's side of the context token exchange and providing key confirmation. We could have used three distinct PDUs instead of one.

It is true that a KRB-ERROR2 that only serves the purpose of final key confirmation without continuation could have a different ASN.1 type for its encrypted part, and a different application tag, however, there seems to be little value in this. Distinguishing between errors with and without key confirmation is even less valuable. Therefore we do not distinguish these three possible PDUs.

3. Negotiation and Use of Extra Context Tokens

In the following text "initiator" refers to the mechanism's initiator functionality (invoked via GSS_Init_sec_context()), and "acceptor" refers to the mechanism's acceptor functionality (invoked via GSS_Accept_sec_context()).

To use this feature, the Kerberos GSS mechanism MUST act as follows:

- To request this feature, initiators SHALL add the new ap-options flag to their AP-REQs.
  - And the initiators SHALL repeat the ap-options in the new AD-AP-OPTIONS AD type in the Authenticator.
- Acceptors that wish to request an additional security context token can only do so when initiators indicate support for it, and MUST do so by returning a KRB-ERROR2. The encrypted part of the KRB-ERROR2 SHALL be encrypted in a key derived (with key usage <TBD>) from one of the following keys: the sub-session key from the AP-REQ's Authenticator (use-initiator-subkey) if it could be decrypted, else the session key from the Ticket (use-ticket-session-key), if it could be decrypted, else the null enc-type/key (use-null-enctype).
- Any KRB-ERROR2 emitted by the acceptor SHALL have the continue-needed e-flag set when the GSS_Accept_sec_context() returns GSS_S_CONTINUE_NEEDED to the application, and in this case the token ID SHALL be 02 00 (KRB_AP_REP, even though the token isn't actually an AP-REP) (see [RFC4121] section 4.1).
- When it consumes a KRB-ERROR2, GSS_Init_sec_context() can return an error (GSS_S_FAILURE) and optionally output an error token, or it can attempt recovery (see Section 5) and output a new AP-REQ security context token.
  - Any error token output by GSS_Init_sec_context() MUST be a KRB-ERROR2, and GSS_Init_sec_context() MUST return GSS_S_FAILURE.
  - The initiator MUST quote the challenge from the KRB-ERROR2 using an AD-CONTINUE-CHALLENGE (see below) authorization data element in any AP-REQ or KRB-ERROR2 response to the acceptor's KRB-ERROR2.
  - When GSS_Init_sec_context() outputs a new AP-REQ security context token, it SHALL return GSS_S_CONTINUE_NEEDED if the application requested mutual authentication and the previous acceptor security context token was a recoverable error (rather

than a request for one more AP-REQ), else it SHALL return GSS_S_COMPLETE.

* When GSS_Init_sec_context() returns an error and the acceptor is awaiting a security context token, GSS_Init_sec_context() MAY generate a KRB-ERROR2 or KRB-ERROR to send to the acceptor.

- Acceptors MUST reject additional AP-REQs which do not have a challenge response nonce matching the one sent by the acceptor in the previous KRB-ERROR2.

- Acceptors MUST reject initial security context tokens that contain a challenge response nonce.

- When GSS_Accept_sec_context() returns an error and outputs an error token, the token MUST be either a KRB-ERROR or a KRB-ERROR2, with the latter having the continue-needed flag cleared.

All non-recoverable KRB-ERROR2 tokens SHALL use the token ID 03 00.

Additional AP-REQs produced by the authenticator MUST have the mutual-required ap-options flag set when a) the application requested mutual authentication, and b) the acceptor's KRB-ERROR2 did not supply the required key confirmation. The acceptor MUST respond to the client's last AP-REQ with an AP-REP when the mutual-required ap-options flag is set or when the GSS_C_MUTUAL_FLAG is set in the "checksum 0x8003", otherwise GSS_Accept_sec_context() MUST NOT produce a response token when it returns GSS_S_COMPLETE.

3.1. Number of Security Context Tokens

The first AP-REQ may well result in an error; the second generally should not. Therefore acceptors SHOULD return a fatal error when a second error results in one security context establishment attempt, except when the first error is that the initiator should use user-to-user authentication. This limits the maximum number of round trips to two (not user-to-user) or three (user-to-user).

The mechanism SHOULD impose some limit on the maximum number of security context tokens. For the time being that limit is six.

Note that in the user-to-user cases (see Section 7) it's possible to have up to three round trips under normal conditions if, for example, the acceptor wishes to avoid the use of replay caches (see Section 6), or if the initiator's clock is too skewed, for example.

3.2. Possible Context Token Sequences

The following successful security context token exchange sequences are possible:

- One token (per-RFC4121; mutual authentication not requested): AP-REQ.
  - In principle this can yield an error token in the case of errors, per-RFC2743.
- Two tokens (per-RFC4121; mutual authentication requested): AP-REQ and AP-REP.
- Two tokens (per-RFC4121; mutual authentication requested): AP-REQ and KRB-ERROR.
- Two tokens (per-RFC4121; mutual authentication requested): AP-REQ and KRB-ERROR2 (non-recoverable error, or recoverable error but the acceptor mechanism is configured to not continue).
- Two tokens (per-RFC4121; mutual authentication requested): AP-REQ and KRB-ERROR2 (recoverable error for the acceptor, but not for the initiator, or the initiator application abandons the partially-established security context).
- Three tokens: AP-REQ, KRB-ERROR2 (recoverable error), AP-REQ.
  - The initiator indicates it supports multiple round trips, and a recoverable error results on the acceptor side.
  - Either the initiator did not request mutual authentication, or the KRB-ERROR2 supplied the necessary key confirmation.
- Three tokens: AP-REQ, KRB-ERROR2 (no error, continue needed), AP-REQ.
  - The initiator indicates it supports multiple round trips, and its Authenticator and Ticket decrypt correctly on the acceptor side, but the acceptor wants to continue, e.g., to avoid the need for a replay cache (see Section 6).
  - This can happen in any recoverable error case where the initiator’s Authenticator (and Ticket) decrypt successfully on the acceptor side.

- Four tokens: AP-REQ, KRB-ERROR2 (recoverable error), AP-REQ, AP-REP.
  - The initiator wanted mutual authentication and a recoverable error occurred where the KRB-ERROR2 could not provide key confirmation, leading to the second round trip.
  - This can happen in any recoverable error case where the initiator's Authenticator did not decrypt successfully.
  - This can also happen in the user-to-user case.
  - This case provides replay cache avoidance without a fifth token because the acceptor provides a challenge in its first (KRB-ERROR2) token and the initiator completes the challenges in its second token.
- Five tokens: AP-REQ, KRB-ERROR2 (with user-to-user TGT), AP-REQ, KRB-ERROR2 (recoverable error), AP-REQ.
  - The initiator does not want mutual authentication, the acceptor wants user-to-user authentication, and the initiator's second AP-REQ elicits a recoverable error.
- Six tokens: AP-REQ, KRB-ERROR2 (with user-to-user TGT), AP-REQ, KRB-ERROR2 (recoverable error), AP-REQ, AP-REP.
  - The initiator wants mutual authentication, the acceptor wants user-to-user authentication, and the initiator's second AP-REQ elicits a recoverable error; none of the KRB-ERROR2 tokens was a key-confirmation token.

Other context token sequences might be possible in the future.

In the above sequences the AP-REP tokens can be AP-REP2 tokens as well.

3.3. Per-Message Token Sequence Numbers

It is REQUIRED that each real AP-REQ in a single security token exchange specify the same start sequence number as preceding AP-REQs in the same security context token exchange.

3.4. Early PROT_READY State

The GSS-API allows security mechanisms to support the use of per-message tokens prior to full security context establishment. In this section we'll call this "early PROT_READY". Early PROT_READY is

optional for the GSS-API and for implementations of mechanisms that support it.

The Kerberos V GSS mechanism supports this in the two-token exchange, with the initiator being PROT_READY before consuming the AP-REP. This extension also supports early PROT_READY, which works as follows:

1. The initiator asserts a sub-session key in each AP-REQ that does not follow a key-confirmation KRB-ERROR2, and GSS_Init_sec_context() sets the prot_ready_state return flag on the first call.

    1. If there are multiple such AP-REQs in a security context token exchange, then each such AP-REQ must assert the same sub-session key.

    2. Subsequent AP-REQs need not carry a sub-session key; acceptors MUST ignore sub-session keys from subsequent AP-REQs.

2. GSS_Accept_sec_context() MUST NOT set the prot_ready_state return flag until it has successfully decrypted an AP-REQ's Ticket and Authenticator from the initiator. If the acceptor requests additional context tokens and signals PROT_READY at that point, then it too will be PROT_READY.

Replay protection for early prot_ready per-message tokens depends on the initiator always generating a fresh sub-session key for every security context's initial context token, on the acceptor always generating a fresh sub-session key for its key confirmation token, and on either a replay cache or the challenge/response token provided for in this document:

- An attacker cannot replay an early per-message token without also replaying the corresponding initial security context token (as otherwise the initiator-asserted sub-session keys won't match), and replay protection for the initial security context token provides replay protection for any subsequent early per-message tokens.

- Per-message tokens made after full security context establishment are protected against replay by the use of the acceptor's sub-session key hierarchy (since the initiator must then use that key).

- AP-REPs and key-confirmation KRB-ERROR2s are protected against replays to initiators by the use of the initiator's sub-session

key.

- o Initial security context tokens (and error-recovery AP-REQs) are protected against replay either by a replay cache on the acceptor side, or by the use of additional context tokens for challenge/response replay cache avoidance (see Section 6).

3.5. Other Requirements, Recommendations, and Non-Requirements

All error PDUs in an AP exchange where the AP-REQ has the continue-needed-ok ap-options flag MUST be KRB-ERROR2 PDUs.

Whenever an acceptor is able to decrypt the Ticket from an AP-REQ and yet wishes or has to output a KRB-ERROR2, then the enc-part of the KRB-ERROR2 MUST be encrypted in either the initiator's sub-session key (from the Authenticator) or the Ticket's session key (if the acceptor could not decrypt the Authenticator).

4. ASN.1 Module for New Protocol Elements

A partial ASN.1 module appears below. This ASN.1 is to be used as if it were part of the base Kerberos ASN.1 module (see RFC4120), therefore the encoding rules to be used are the Distinguished Encoding Rules (DER) [CCITT.X690.2002], and the environment is one of explicit tagging.

```
KerberosExtraContextTokens DEFINITIONS ::=
BEGIN
EXPORTS ad-continue-challenge,
        AD-CONTINUE-CHALLENGE,
        KrbErrorEncPartFlags,
        KRB-ERROR2,
        ErrorFlags;
IMPORTS UInt32, Int32, KerberosTime,
        Microseconds, KerberosFlags,
        Checksum, EncryptedData,
        EncryptionKey, KerberosString,
        AuthorizationData, TYPED-DATA,
        HostAddresses, Ticket FROM KERBEROS5;

APOptions       ::= KerberosFlags
        -- reserved(0),
        -- use-session-key(1),
        -- mutual-required(2),
        -- continue-needed-ok(TBD)

-- Challenge (for use in Authenticator)
ad-continue-challenge     Int32 ::= -5 -- <TBD>
AD-CONTINUE-CHALLENGE ::= OCTET STRING

-- AP options, integrity-protected
ad-ap-options             Int32 ::= -6 -- <TBD>
AD-AP-OPTIONS         ::= KerberosFlags

KrbErrorEncPartFlags ::= ENUMERATED {
        use-null-enctype(0),
        use-initiator-subkey(1),
        use-ticket-session-key(2),
        ...
}

-- Application tag TBD
KRB-ERROR2          ::= [APPLICATION 55] SEQUENCE {
        pvno            [0] INTEGER (5),
        msg-type        [1] INTEGER (55), -- TBD
        enc-part-key    [2] KrbErrorEncPartFlags,
```

```
        enc-part        [3] EncryptedData -- EncKRBErrorPart
    }

    -- Alias type name
    AP-REP2             ::= KRB-ERROR2

    ErrorFlags ::= ENUMERATED {
            final(0),
            continue-needed(1),
            ...
    }

    -- Application tag TBD
    EncKRBErrorPart    ::= [APPLICATION 56] SEQUENCE {
            continue-challenge  [0]  AD-CHALLENGE-RESPONSE,
            stime               [1]  KerberosTime,
            susec               [2]  Microseconds,
            subkey              [3]  EncryptionKey OPTIONAL,
            seq-number          [4]  UInt32 OPTIONAL,
            error-code          [5]  Int32,
            e-flags             [6]  ErrorFlags,
            e-text              [7]  UTF8String OPTIONAL,
            e-data              [8]  OCTET STRING OPTIONAL,
            e-typed-data        [9]  TYPED-DATA OPTIONAL,
            -- For recovery from KRB_AP_ERR_BADADDR:
            your-addresses      [10] HostAddresses OPTIONAL,
            ad-data             [11] AuthorizationData OPTIONAL,
            tgt                 [12] Ticket OPTIONAL, -- for user2user
            ...
    }

    END
```

Figure 1: ASN.1 module (with explicit tagging)

5. Recoverable Errors and Error Recovery

The following Kerberos errors can be recovered from automatically using this protocol:

- KRB_AP_ERR_TKT_EXPIRED: the initiator should get a new service ticket;
- KRB_AP_ERR_TKT_NYV: the initiator should get a new service ticket;
- KRB_AP_ERR_REPEAT: the initiator should build a new AP-REQ;
- KRB_AP_ERR_SKEW: see Section 8;
- KRB_AP_ERR_BADKEYVER: the initiator should get a new service ticket;
- KRB_AP_PATH_NOT_ACCEPTED: the initiator should get a new service ticket using a different transit path;
- KRB_AP_ERR_INAPP_CKSUM: the initiator should try again with a different checksum type.

Error codes that denote PDU corruption (and/or an active attack) can also be recovered from by attempting a new AP-REQ, though subsequent AP-REQs may fail for the same reason:

- KRB_AP_ERR_BAD_INTEGRITY
- KRB_AP_ERR_BADVERSION
- KRB_AP_ERR_BADMATCH
- KRB_AP_ERR_MSG_TYPE
- KRB_AP_ERR_MODIFIED

Other error codes that may be recovered from:

- KRB_AP_ERR_BADADDR: the acceptor SHOULD include a list of one or more client network addresses as reported by the operating system, but if the acceptor does not then the continue-needed e-flag MUST NOT be included and the error must be final.

5.1. Authenticated Errors

The following errors, at least, can be authenticated in AP exchanges:

- KRB_AP_ERR_TKT_EXPIRED
- KRB_AP_ERR_TKT_NYV
- KRB_AP_ERR_REPEAT
- KRB_AP_ERR_SKEW
- KRB_AP_PATH_NOT_ACCEPTED
- KRB_AP_ERR_INAPP_CKSUM
- KRB_AP_ERR_BADADDR

6. Replay Cache Avoidance

By using an additional AP-REQ and a challenge/response nonce, this protocol is immune to replays of AP-REQ PDUs and does not need a replay cache. Acceptor implementations MUST not insert Authenticators from extra round trips into a replay cache when there are no other old implementations on the same host (and with access to the same acceptor credentials) that ignore critical authorization data or which don't know to reject initial AP-REQs that contain a challenge response nonce.

In the replay cache avoidance case where there's no actual error (e.g., time skew) the acceptor's KRB-ERROR2 will have KDC_ERR_NONE as the error code, with the continue-needed e-flag.

6.1. Replay Cache Avoidance without Extensions

Many Kerberos services can avoid the use of a replay cache altogether, but it's tricky to know when it's safe to do so. For Kerberos it's safe to not use a replay cache for AP-REQs/ Authenticators when either:

- the application doesn't require replay detection at all and
  - no other acceptor/service application shares the same long-term service keys for its service principal

or

- the application protocol always has the initiator/client send the first per-message token (or KRB-SAFE/PRIV PDU) which can then function as a challenge response, and
  - no other acceptor/service application shares the same long-term service keys for its service principal

It is difficult to establish the second part of the above conjunctions programmatically. In practice this is best left as a local configuration matted on a per-service name basis.

For example, it's generally safe for NFSv4 [RFC3530] to not use a replay cache for the Kerberos GSS mechanism, but it is possible for multiple Kerberos host-based service principals on the same host to share the same keys, therefore in practice, the analysis for NFSv4 requires more analysis. The same is true for SSHv2 [RFC4251] (SSHv2 implementations share the same service principal as other non-GSS Kerberos applications that do sometimes need a replay cache).

7. User-to-User Authentication

There are two user2user authentication cases:

1. the KDC only allows a service principal to use user2user authentication,

2. the service principal does not know its long-term keys or otherwise wants to use user2user authentication even though the KDC vended a service ticket.

In the first case the initiator knows this because the KDC returns KDC_ERR_MUST_USE_USER2USER. The initiator cannot make a valid AP-REQ in this case, yet it must send some sort of initial security context token! For this case we propose that the initiator make an AP-REQ with a Ticket with zero-length enc-part (and null enctype) and a zero-length authenticator (and null enctype). The acceptor will fail to process the AP-REQ, of course, and SHOULD respond with a continue-needed KRB-ERROR2 (using the null enc-type for the enc-part) that includes a TGT for the acceptor.

In the second case the initiator does manage to get a real service ticket for the acceptor but the acceptor nonetheless wishes to use user2user authentication.

In both cases the acceptor responds with a KRB-ERROR2 with the KRB_AP_ERR_USER_TO_USER_REQUIRED error code and including a TGT for itself.

In both cases the initiator then does a TGS request with a second ticket to get a new, user2user Ticket. Then the initiator makes a new AP-REQ using the new Ticket, and proceeds.

8. Acceptor Clock Skew Correction

An initiator in possession of a (short-lived) valid service ticket for a given service principal... must have had little clock skew relative to the service principal's realm's KDC(s), or the initiator must have been able to correct its local clock skew. But the acceptor's clock might be skewed, yielding a KRB_AP_ERR_SKEW error with a challenge. The client could recover from this by requesting a new service ticket with this challenge as an authorization data element. The acceptor should be able to verify this in the subsequent AP-REQ, and then it should be able to detect that its clock is skewed and to estimate by how much.

9. Security Considerations

This document deals with security.

The new KRB-ERROR2 PDU is cryptographically distinguished from the original mechanism's acceptor success security context token (AP-REQ).

Not every KRB-ERROR2 can be integrity protected. This is unavoidable.

Because in the base Kerberos V5 GSS-API security mechanism all errors are unauthenticated, and because even with this specification some elements are unauthenticated, it is possible for an attacker to cause one peer to think that the security context token exchange has failed while the other thinks it will continue. This can cause an acceptor to waste resources while waiting for additional security context tokens from the initiator. This is not really a new problem, however: acceptor applications should already have suitable timeouts on security context establishment.

There is a binding of preceding security context tokens in each additional AP-REQ, via the challenge-response nonce. This binding is weak, and does not detect all modifications of unauthenticated plaintext in preceding security context tokens.

[[anchor1: We could use the GSS_EXTS_FINISHED extension from draft-ietf-kitten-iakerb to implement a strong binding of all context tokens.]]

Early prot_ready per-message tokens have security considerations that are beyond the scope of this document and which are not exhaustively described elsewhere yet. Use only with care.

10. IANA Considerations

[[anchor2: Various allocations are required...]]

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4121] Zhu, L., Jaganathan, K., and S. Hartman, "The Kerberos Version 5 Generic Security Service Application Program Interface (GSS-API) Mechanism: Version 2", RFC 4121, July 2005.

[CCITT.X680.2002] International Telephone and Telegraph Consultative Committee, "Abstract Syntax Notation One (ASN.1): Specification of basic notation", CCITT Recommendation X.680, July 2002.

[CCITT.X690.2002] International Telephone and Telegraph Consultative Committee, "ASN.1 encoding rules: Specification of basic encoding Rules (BER), Canonical encoding rules (CER) and Distinguished encoding rules (DER)", CCITT Recommendation X.690, July 2002.

11.2. Informative References

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC4251] Ylonen, T. and C. Lonvick, "The Secure Shell (SSH) Protocol Architecture", RFC 4251, January 2006.

[I-D.swift-win2k-krb-user2user] Swift, M., Brezak, J., and P. Moore, "User to User Kerberos Authentication using GSS-API", draft-swift-win2k-krb-user2user-03 (work in progress), February 2011.

Authors' Addresses

Nicolas Williams
Cryptonector, LLC

Email: nico@cryptonector.com

Roland Charles Dowdeswell
Dowdeswell Security Architecture

Email: elric@imrryr.org