Independent Submission
Internet-Draft
Updates: 5537 (if approved)
Intended status: Standards Track
Expires: October 10, 2017

M. Baeuerle
STZ Elektronik

April 8, 2017

Cancel-Locks in Netnews articles
draft-baeuerle-netnews-cancel-lock-04

Abstract

This document defines an extension to the Netnews Article Format that may be used to authenticate the cancelling and superseding of existing articles. If approved, this document updates RFC5537.

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 October 10, 2017.

Table of Contents

1. Introduction

The authentication system defined in this document is intended to be used as a simple method to verify that the author of an article which cancels ([RFC5537] Section 5.3) or supersedes ([RFC5537] Section 5.4) another one is either the poster, posting agent, moderator or injecting agent that processed the original article when it was in its proto-article form.

One property of this system is that it prevents tracking of individual users.

There are other authentication systems available with different properties. When everybody should be able to verify who the originator is, e.g. for control messages to add or remove newsgroups ([RFC5537] Section 5.2), an OpenPGP [RFC4880] signature is suited.

1.1. Conventions Used in This Document

Any term not defined in this document has the same meaning as it does in [RFC5536] or [RFC5537].

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

1.2. Author's Note

Please write the letters "ae" in "Baeuerle" as an a-umlaut (U+00E4, "ä" in XML), the first letter in "Elie" with an acute accent (U+00C9, "É" in XML), the letters "ss" in Janssen as an eszett (U+00DF, "ß" in XML) and the letters "ue" in Baden-Wuerttemberg as an u-umlaut (U+00FC, "ü" in XML) wherever this is possible.

2. Header Fields

This section describes the formal syntax of the new header fields using ABNF [RFC5234]. It extends the syntax in Section 3 of [RFC5536] and non-terminals not defined in this document are defined there. The [RFC5536] ABNF should be imported first before attempting to validate these rules.

The new header fields Cancel-Lock and Cancel-Key are defined by this document, they follow the rules described in [RFC5536] Section 2.2:

```
fields =/ *( cancel-lock / cancel-key )
```

Each of these header fields MUST NOT occur more than once in an article.

Both new header field bodies contain lists of encoded values. Every entry is based on a <scheme>:

```
scheme       = "sha256" / "sha512" / 1*scheme-char / obs-scheme
scheme-char  = ALPHA / DIGIT / "-" / "/"
```

The hash algorithms for <scheme> are defined in [SHA], see also [RFC1321] and [RFC6151] for MD5, [RFC3174] for SHA1 and [RFC6234] for the SHA2 family. The Base64 encoding used is defined in Section 6.8 of [RFC2045].

This document defines two values for <scheme>: "sha256" and "sha512". The scheme "sha256" is mandatory to implement.

2.1. Cancel-Lock

```
cancel-lock     = "Cancel-Lock:" SP c-lock-list CRLF
c-lock-list     = [CFWS] c-lock *(CFWS c-lock) [CFWS]
c-lock          = scheme ":" c-lock-string
c-lock-string   = *(4base64-char) [base64-terminal]
base64-char     = ALPHA / DIGIT / "+" / "/"
base64-terminal = 2base64-char "==" / 3base64-char "="
```

Comments in CFWS can cause interoperability problems, so comments SHOULD NOT be generated but MUST be accepted.

If <scheme> is not supported by an implementation, the corresponding <c-lock> element MUST be skipped and potential following <c-lock> elements MUST NOT be ignored.

<c-lock-string> is the Base64 encoded output of a hash operation (defined by <scheme>) of the Base64 encoded key "K" that is intended to authenticate the person or agent that created or processed respectively the proto-article up to injection (inclusively):

```
Base64(hash(Base64(K)))
```

Because of the one-way nature of the hash operation the key "K" is not revealed.

2.2. Cancel-Key

```
cancel-key   = "Cancel-Key:" SP c-key-list CRLF
c-key-list   = [CFWS] c-key *(CFWS c-lock) [CFWS]
c-key        = scheme ":" c-key-string
c-key-string = c-lock-string / obs-c-key-string
```

Comments in CFWS can cause interoperability problems, so comments SHOULD NOT be generated but MUST be accepted.

If <scheme> is not supported by an implementation, the corresponding <c-key> element MUST be skipped and potential following <c-key> elements MUST NOT be ignored.

<c-key-string> is the Base64 encoded key "K" that was used to create the <c-lock> element in the Cancel-Lock header field body (as defined in Section 2.1 of this document) of the original article:

```
Base64(K)
```

The relaxed syntax definition of <c-key-string> above is required for backward compatibility with implementations that are not compliant with this specification. Compliant implementations SHOULD generate valid Base64 (that is to say the syntax of <c-lock-string> as defined in Section 2.1 of this document) and MUST accept strings of <base64-octet> characters (that is to say the syntax of <obs-c-key-string> as defined in Section 6 of this document).

3. Use

3.1. Adding an initial Cancel-Lock header field to a proto-article

A Cancel-Lock header field MAY be added to a proto-article by the poster or posting agent which will include one or more <c-lock> elements.

If the poster or posting agent doesn't add a Cancel-Lock header field to a proto-article, then an injecting agent (or moderator) MAY add one or more provided that it positively authenticates the author. The injecting agent (or moderator) MUST NOT add this header field to a proto-article unless it is able to authenticate all cancelling or superseding attempts from the poster and automatically add a working Cancel-Key header field or extend an existing one for such proto-articles.

Other agents MUST NOT add this header field to articles or proto-articles that they process.

3.2. Extending the Cancel-Lock header field of a proto-article

If a Cancel-Lock header field has already been added to a proto-article then any agent further processing the proto-article up to the injecting agent (inclusively) MAY append additional <c-lock> elements to those already in the header field body.

Use cases for extending the Cancel-Lock header field body:

- A moderator wants the ability to cancel articles after approving them.
- An injecting agent acts representitive for posting agents without support for the autentication system described in this document.
- A news administrator wants the ability to cancel articles that were injected by its system (because they e.g. violate its abuse policy).

Once an article is injected then this header field MUST NOT be altered. In particular, relaying agents beyond the injecting agent MUST NOT alter it.

3.3. Adding a Cancel-Key header field to a proto-article

A Cancel-Key header field MAY be added to a proto-article containing a Control or Supersedes header field by the poster or posting agent which will include one or more <c-key> elements. They will correspond to some or all of the <c-lock> elements in the article referenced by the Control (with a "cancel" command as defined in [RFC5537]) or Supersedes header field.

If, as mentioned in Section 3.1 an injecting agent (or moderator) has added a Cancel-Lock header field to an article listed in the Control (with "cancel" command as defined in [RFC5537]) or Supersedes header field then (given that it authenticates the poster as being the same as the poster of the original article) it MUST add the Cancel-Key header field with at least one <c-key> element that correspond to that article.

Other agents MUST NOT alter this header field.

3.4. Extending the Cancel-Key header field of a proto-article

If a Cancel-Key header field has already been added to a proto-article then any agent further processing the proto-article up to the injecting agent (inclusively) MAY append additional <c-key> elements to those already in the header field body.

If, as mentioned in Section 3.2 an injecting agent (or moderator) has extended the Cancel-Lock header field in an article listed in the Control (with "cancel" command as defined in [RFC5537]) or Supersedes header field then (given that it authenticates the poster as being the same as the poster of the original article) it MUST extend the Cancel-Key header field body with at least one <c-key> element that correspond to that article.

Once an article is injected then this header field MUST NOT be altered. In particular, relaying agents beyond the injecting agent MUST NOT alter it.

3.5. Check a Cancel-Key header field

When a serving agent receives an article that attempts to cancel or supersede a previous article via Control (with a "cancel" command as defined in [RFC5537]) or Supersedes header field, the system defined in this document can be used for authentication. The general handling of articles containing such attempts as defined in [RFC5537] is not changed by this document.

To process the authentication, the received article must contain a Cancel-Key header field and the original article a Cancel-Lock header field. If this is not the case, the authentication is not possible (failed).

For the authentication check, every supported <c-key> element from the received article is processed as follows:

1. The <code-string> part of the <c-key> element is hashed using the algorithm defined by its <scheme> part.

2. For all <c-lock> elements with the same <scheme> in the original article their <code-string> part is compared to the calculated hash.

3. If one is equal, the authentication is passed and the processing of further elements can be aborted.

4. If no match was found and there are no more <c-key> elements to process, the authentication failed.

4. Calculating the key data

This section is informative, not normative.

It is suggested to use the function:

```
K = HMAC(uid+mid, sec)
```

to create the key "K" for an article with Message-ID <mid> that belongs to the User-ID <uid> (e.g. the login name of the user). HMAC is outlined in [RFC2104]. HMAC is computed over the data <uid+mid> (with '+' representing the concatenation operation), using <sec> as a secret key held locally that can be used for multiple articles. This method removes the need for a per-article database containing the keys used for every article. [[Q1: Security review: Some existing implementations concatenates the <uid> part with <sec> instead of <mid>. This variant was not used to ensure that <sec> is directly used as HMAC key (to avoid confusion with the length considerations below).]]

A posting agent should add the Message-ID header field to the proto-article itself and use the content of the header field body as <mid> (including literal angle brackets).

A posting agent, that uses a dedicated local secret <sec> for every user, should use an empty string for the <uid> part.

The local secret <sec> should have a length of at least the output size of the hash function that is used by HMAC (256 bit / 32 octets for SHA256). If the secret is not a random value, but e.g. some sort of human readable password, it should be much longer. In any case it is important that this secret can not be guessed.

Note that the hash algorithm used as base for the HMAC operation is not required to be the same as specified by <scheme>. An agent that verifies a Cancel-Key header field body simply checks whether one of its <c-key> elements matches one of the <c-lock> elements with the same <scheme> in the Cancel-Lock header field body of the original article.

Common libraries like OpenSSL can be used for the cryptographic operations.

5. Examples

5.1. Without UID

Example data for creation of a <c-lock> element with HMAC-SHA256 and empty string as <uid> (as suggested in Section 4 for posting agents):

```
Message-ID: <12345@mid.example>
```

```
mid: <12345@mid.example>
sec: ExampleSecret
K  : HMAC-SHA256(mid, sec) ;mid used as data, sec as secret key
```

Calculation of Base64(K) using the OpenSSL command line tools in a POSIX shell:

```
$ printf "%s" "<12345@mid.example>" \
  | openssl dgst -sha256 -hmac "ExampleSecret" -binary \
  | openssl enc -base64
qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=
```

This can be used as <c-key-string> for cancelling or superseding the article <12345@mid.example>.

Calculation of Base64(SHA256(Base64(K))) required for <c-lock-string> using the OpenSSL command line tools in a POSIX shell:

```
$ printf "%s" "qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=" \
  | openssl dgst -sha256 -binary \
  | openssl enc -base64
s/pmK/3grrz++29ce2/mQydzJuc7iqHn1nqcJiQTPMc=
```

Inserted into the Cancel-Lock header field body of article <12345@mid.example> it looks like this:

```
Cancel-Lock: sha256:s/pmK/3grrz++29ce2/mQydzJuc7iqHn1nqcJiQTPMc=
```

Inserted into the Cancel-Key header field body of an article that should cancel or supersede article <12345@mid.example> it looks like this:

```
Cancel-Key: sha256:qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=
```

5.2. With UID

Example data for creation of a <c-lock> element with HMAC-SHA256 and "JaneDoe" as <uid> (as suggested in Section 4):

```
Message-ID: <12345@mid.example>
```

```
uid: JaneDoe
mid: <12345@mid.example>
sec: AnotherSecret
K  : HMAC-SHA256(uid+mid, sec) ;uid+mid used as data, sec as secret key
```

Calculation of Base64(K) using the OpenSSL command line tools in a POSIX shell:

```
$ printf "%s" "JaneDoe<12345@mid.example>" \
  | openssl dgst -sha256 -hmac "AnotherSecret" -binary \
  | openssl enc -base64
yM0ep490Fzt83CLYYAytm3S2HasHhYG4LAeAlmuSEys=
```

This can be used as <c-key-string> for cancelling or superseding the article <12345@mid.example>.

Calculation of Base64(SHA256(Base64(K))) required for <c-lock-string> using the OpenSSL command line tools in a POSIX shell:

```
$ printf "%s" "yM0ep490Fzt83CLYYAytm3S2HasHhYG4LAeAlmuSEys=" \
  | openssl dgst -sha256 -binary \
  | openssl enc -base64
NSBTz7BfcQFTCen+U4lQ0VS8VIlZao2b8mxD/xJaaeE=
```

Inserted into the Cancel-Lock header field body of article <12345@mid.example> it looks like this:

```
Cancel-Lock: sha256:NSBTz7BfcQFTCen+U4lQ0VS8VIlZao2b8mxD/xJaaeE=
```

Inserted into the Cancel-Key header field body of an article that should cancel or supersede article <12345@mid.example> it looks like this:

```
Cancel-Key: sha256:yM0ep490Fzt83CLYYAytm3S2HasHhYG4LAeAlmuSEys=
```

5.3. Other examples

Other matching pair of Cancel-Lock and Cancel-Key header fields:

```
Cancel-Lock: sha256:RrKLp7YCQc9T8HmgSbxwIDlnCDWsgy1awqtiDuhedRo=
Cancel-Key: sha256:sSkDke97Dh78/d+Diu1i3dQ2Fp/EMK3xE2GfEqZlvK8=
```

With obsolete syntax (uses a <c-key-string> with invalid/missing Base64 padding):

```
Cancel-Lock: sha1:bNXHc6ohSmeHaRHHW56BIWZJt+4=
Cancel-Key: ShA1:aaaBBBcccDDDeeeFFF
```

Let's assume that all the examples above are associated to the same article (e.g. created by different agents):

```
Cancel-Lock: sha256:s/pmK/3grrz++29ce2/mQydzJuc7iqHn1nqcJiQTPMc=
             sha256:NSBTz7BfcQFTCen+U4lQ0VS8VIlZao2b8mxD/xJaaeE=
             sha256:RrKLp7YCQc9T8HmgSbxwIDlnCDWsgy1awqtiDuhedRo=
             sha1:bNXHc6ohSmeHaRHHW56BIWZJt+4=
Cancel-Key: sha256:qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=
            sha256:yM0ep490Fzt83CLYYAytm3S2HasHhYG4LAeAlmuSEys=
            sha256:sSkDke97Dh78/d+Diu1i3dQ2Fp/EMK3xE2GfEqZlvK8=
            ShA1:aaaBBBcccDDDeeeFFF
```

5.4. Manual checks

Manual checks using the OpenSSL command line tools in a POSIX shell:

```
$ printf "%s" "qv1VXHYiCGjkX/N1nhfYKcAeUn8bCVhrWhoKuBSnpMA=" \
  | openssl dgst -sha256 -binary \
  | openssl enc -base64
s/pmK/3grrz++29ce2/mQydzJuc7iqHn1nqcJiQTPMc=

$ printf "%s" "yM0ep490Fzt83CLYYAytm3S2HasHhYG4LAeAlmuSEys=" \
  | openssl dgst -sha256 -binary \
  | openssl enc -base64
NSBTz7BfcQFTCen+U4lQ0VS8VIlZao2b8mxD/xJaaeE=

$ printf "%s" "sSkDke97Dh78/d+Diu1i3dQ2Fp/EMK3xE2GfEqZlvK8=" \
  | openssl dgst -sha256 -binary \
  | openssl enc -base64
RrKLp7YCQc9T8HmgSbxwIDlnCDWsgy1awqtiDuhedRo=

$ printf "%s" "aaaBBBcccDDDeeeFFF" \
  | openssl dgst -sha1 -binary \
  | openssl enc -base64
bNXHc6ohSmeHaRHHW56BIWZJt+4=
```

6. Obsolete Syntax

Implementations of earlier drafts of this specification defined a different value for <scheme> than this version. The following value for <scheme> is now deprecated and SHOULD NOT be generated anymore. Serving agents SHOULD still accept it for a transition period as long as the corresponding hash function is not considered unsafe (see Section 7 for details), or already marked as OBSOLETE in the Netnews Cancel-Lock hash algorithm registry (Section 8.1).

```
obs-scheme = "sha1"
```

It is important for backward compatibility that the deprecated value for <scheme> is not phased out too early. Security and compatibility concerns should be carefully weighed before choosing to remove <obs-

scheme> from existing implementations (or not implementing it in new ones).

Earlier drafts of this specification allowed more liberal syntax for <c-key-string>:

```
obs-c-key-string = 1*base64-octet
base64-octet     = ALPHA / DIGIT / "+" / "/" / "="
```

<obs-c-key-string> SHOULD NOT be generated but MUST be accepted.

7. Security Considerations

The important properties of the hash function used for <scheme> are the preimage and second preimage resistance. A successful preimage attack would reveal the real <c-key-string> element that was used to create the Cancel-Lock header field body of the original article. A successful second preimage attack would allow to create a new, different <c-key-string> element that, if used in the Cancel-Key header field body, matches a <c-lock-string> element in the Cancel-Lock header field body of the original article too. Both cases would break the authentication system defined in this document.

Collision resistance of the hash function used for <scheme> is less important. Finding two <c-key> elements for the Cancel-Key header field that match to a <c-lock> element of an arbitary Cancel-Lock header field is not helpful to break the authentication system defined in this document (if a specific article is defined as target). Only collateral damage by arbitrary cancel or supersede is possible.

Currently there is no known practicable preimage and second preimage attack against the hash function SHA1. Therefore there is no hurry to replace it. The reasons why this document specifies hash functions from the SHA2 family are:

- The last draft for the authentication system defined in this document is nearly two decades old. The client side implementations are moving forward extremely slowly too (newsreaders from the last millenium are still in heavy use). What is defined today should be strong enough for at least the next decades.

- The collision resistance of SHA1 is already broken, therefore it is now obsolete for digital signatures as used in TLS. It is intended that an implementation of the authentication system defined in this document can share the same cryptographic library functions that are used for TLS.

- It is intended that the same hash function can be used for <scheme> and (as base) for the HMAC that is suggested in Section 4. See notes below for HMAC-MD5 and HMAC-SHA1.

- The SHA2 family of hash algorithms is widely supported by cryptographic libraries. In contrast, SHA3 is currently not supported by e.g. OpenSSL.

The operation HMAC(uid+mid, sec) as suggested in Section 4 must be able to protect the local secret <sec>. The Message-ID <mid> is public (in the Message-ID header field body) and <uid> is optional. An attacker who wants to steal/use a local secret only need to break this algorithm (regardless of <scheme>), because Cancel-Key header fields are explicitly published for every request to cancel or supersede existing articles.

Even if HMAC-MD5 and HMAC-SHA1 are not considered broken today, it is desired to have some more security margin here. Breaking <scheme> only allows to authenticate a single forged cancel or supersede request. With <sec> in hand it is possible to forge such requests for all articles that contain Cancel-Lock header field bodies with elements that are generated with this <sec> in the past. Changing <sec> in regular intervals can be used to mitigate the potential damage.

If an implementation choose to not implement the key calculation algorithm as suggested in Section 4, or to implement it with HMAC based on a different hash function than <scheme>, the key size used should be at least 128 bit with "sha256" for <scheme> and at least 80 bit with "sha1" for <scheme>. [[Q2: Security review: Should these recommendations remain in the document, or does an RFC exist to refer to with regards to security recommendations?]]

8. IANA Considerations

IANA has registered the following header fields in the Permanent Message Header Field Repository, in accordance with the procedures set out in [RFC3864]:

Header field name: Cancel-Lock
Applicable protocol: netnews
Status: standard
Author/change controller: IETF
Specification document(s): This document

```
Header field name: Cancel-Key
Applicable protocol: netnews
Status: standard
Author/change controller: IETF
Specification document(s): This document
```

The Netnews Cancel-Lock hash algorithm registry will be maintained by IANA.

The registry will be available at <https://www.iana.org/assignments/netnews-cancel-lock-parameters/>.

8.1. Algorithm Name Registration Procedure

IANA will register new Cancel-Lock hash algorithm names on a First Come First Served basis, as defined in BCP 26 [RFC5226]. IANA has the right to reject obviously bogus registration requests, but will perform no review of claims made in the registration form.

Registration of a Netnews Cancel-Lock hash algorithm is requested by filling in the following template and sending it via electronic mail to IANA at <iana@iana.org>:

```
Subject: Registration of Netnews Cancel-Lock hash algorithm X
Netnews Cancel-Lock hash algorithm name:
Security considerations:
Published specification (recommended):
Contact for further information:
Intended usage: (One of COMMON, LIMITED USE, or OBSOLETE)
Owner/Change controller:
Note: (Any other information that the author deems relevant may be
      added here.)
```

Any name that conforms to the syntax of a Netnews Cancel-Lock algorithm Section 2 can be used. Especially, Netnews Cancel-Lock algorithms are named by strings consisting of letters, digits, hyphens and/or slashes.

Authors may seek community review by posting a specification of their proposed algorithm as an Internet-Draft. Netnews Cancel-Lock hash algorithms intended for widespread use should be standardized through the normal IETF process, when appropriate.

8.2. Change control

Once a Netnews Cancel-Lock hash algorithm registration has been published by IANA, the owner may request a change to its definition.

The change request follows the same procedure as the initial registration request.

The owner of a Netnews Cancel-Lock hash algorithm may pass responsibility for the algorithm to another person or agency by informing IANA; this can be done without discussion or review.

The IESG may reassign responsibility for a Netnews Cancel-Lock hash algorithm. The most common case of this will be to enable changes to be made to algorithms where the owner of the registration has died, has moved out of contact, or is otherwise unable to make changes that are important to the community.

Netnews Cancel-Lock hash algorithm registrations MUST NOT be deleted; algorithms that are no longer believed appropriate for use can be declared OBSOLETE by a change to their "intended usage" field; such algorithms will be clearly marked in the registry published by IANA.

The IESG is considered to be the owner of all Netnews Cancel-Lock hash algorithms that are on the IETF Standards Track.

8.3. Registration of the Netnews Cancel-Lock hash algorithms

This section gives a formal definition of the Netnews Cancel-Lock hash algorithms as required by Section 8.1 for the IANA registry.

```
Netnews Cancel-Lock hash algorithm name: md5
Security considerations: See corresponding section of this document
Published specification: This document
Contact for further information: Author of this document
Intended usage: OBSOLETE
Owner/Change controller: IESG <iesg@ietf.org>
Note: Do not use this algorithm anymore

Netnews Cancel-Lock hash algorithm name: sha1
Security considerations: See corresponding section of this document
Published specification: This document
Contact for further information: Author of this document
Intended usage: LIMITED USE
Owner/Change controller: IESG <iesg@ietf.org>
Note: This algorithm is intended for backward compatibility
```

Netnews Cancel-Lock hash algorithm name: sha256
Security considerations: See corresponding section of this document
Published specification: This document
Contact for further information: Author of this document
Intended usage: COMMON
Owner/Change controller: IESG <iesg@ietf.org>
Note: This algorithm is mandatory to implement

Netnews Cancel-Lock hash algorithm name: sha512
Security considerations: See corresponding section of this document
Published specification: This document
Contact for further information: Author of this document
Intended usage: COMMON
Owner/Change controller: IESG <iesg@ietf.org>
Note: This algorithm is optional

9. References

9.1. Normative References

[RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996, <http://www.rfc-editor.org/info/rfc2045>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <http://www.rfc-editor.org/info/rfc3864>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[RFC5536] Murchison, K., Ed., Lindsey, C., and D. Kohn, "Netnews Article Format", RFC 5536, DOI 10.17487/RFC5536, November 2009, <http://www.rfc-editor.org/info/rfc5536>.

[RFC5537] Allbery, R., Ed. and C. Lindsey, "Netnews Architecture and Protocols", RFC 5537, DOI 10.17487/RFC5537, November 2009, <http://www.rfc-editor.org/info/rfc5537>.

[SHA] National Institute of Standards and Technology, "Secure Hash Standard (SHS)", FIPS 180-4, DOI 10.6028/FIPS.180-4, August 2015, <http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

9.2. Informative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <http://www.rfc-editor.org/info/rfc1321>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <http://www.rfc-editor.org/info/rfc2104>.

[RFC3174] Eastlake 3rd, D. and P. Jones, "US Secure Hash Algorithm 1 (SHA1)", RFC 3174, DOI 10.17487/RFC3174, September 2001, <http://www.rfc-editor.org/info/rfc3174>.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, DOI 10.17487/RFC4880, November 2007, <http://www.rfc-editor.org/info/rfc4880>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <http://www.rfc-editor.org/info/rfc6151>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

[USEFOR-CANCEL-LOCK] Lyall, S., "Cancel-Locks in Usenet articles.", Work in Progress, November 1998.

Appendix A. Acknowledgements

The author acknowledges the original author of the Cancel-Lock authentication system as documented in draft-ietf-usefor-cancel-lock: Simon Lyall. He has written the original draft and former version [USEFOR-CANCEL-LOCK] and approved the usage of his work for this document. This document is mostly based on his work and was originally intended as revision 02. It must be renamed because the USEFOR IETF WG is now closed.

The author would like to thank the following individuals for contributing their ideas and reviewing this specification: Russ Allbery, Julien Elie, Urs Janssen, Richard Kettlewell, Marcel Logen, Holger Marzen, Dennis Preiser, Emil Schuster. And Peter Faust and Alfred Peters for providing statistic data about the algorithms currently in use.

Appendix B. Document History (to be removed by RFC Editor before publication)

B.1. Changes since -03

- Added note for change interval of <sec> in Section 7.
- Changed wording in Section 7.
- Splitted Section 5 into multiple subsections.
- Added example with UID in Section 5.
- Changed "SHOULD NOT" to uppercase in Section 6.
- Reformatted Section 8, Section 8.1 and Section 8.3.
- Fixed spelling in Section 4.

B.2. Changes since -02

- Added Section 8.2.
- Added note about algorithm names in Section 8.1.
- Added "/" to scheme-char in Section 2.
- Removed case sensitivity of scheme and normative reference to RFC7405 in Section 2 again.
- Added "sha512" scheme in Section 2.

- Changed wording in Section 8.3.
- Fixed typo "canceling" in Section 5.
- Changed calculation formulas to use "Base64" in Section 2.1 and Section 2.2.
- Added obsolete algorithm "md5" in Section 8.3.
- Added note that posting agents should add the Message-ID header field to proto-articles and use its content for <mid> in Section 4.
- Added <uid> part to key calculation in Section 4.
- Added note to generate CFWS without comments in Section 2.1 and Section 2.2.
- Changed ABNF to allow CFWS at the beginning of header fields in Section 2.1 and Section 2.2.
- Changed wording for "header"/"header field"/"header field body".
- Added Section 3.4.
- Changed wording in Section 3.1.
- Allowed additional whitespace at the beginning of header fields in Section 2.1 and Section 2.2.
- Changed definition of "c-key-string" in Section 2.2.
- Added "obs-c-key-string" to Section 6.
- Fixed typo in Section 2.2 ("c-lock" replaced by "c-key").
- Added key length recommendation in Section 7.
- Renamed "sha-256" scheme to "sha256".
- Modified header and abstract section to list RFC5537 as updated by this document again.
- Added "USEFOR-CANCEL-LOCK" as informative reference.
- Changed wording in Section 4.

B.3. Changes since -01

- Changed wording in Section 7.
- Added example for HMAC calculation in Section 5.
- Changed wording in Section 4.
- Added use cases to Section 3.2.
- Replaced wording "injecting-agent" by "injecting agent".
- Added Definition for "LOWER" in Section 2.
- Added Section 8.3.
- Added Section 8.1.
- Added new entries for header field registry in Section 8.
- Removed recommendation that moderators and injecting agents should add only one Cancel-Lock or Cancel-Key respectively to the list in Section 3.1, Section 3.2 and Section 3.3.
- Added missing headerfield termination to Section 2.1 and Section 2.2.
- Removed definition for "code-string" from Section 2. Added stricter definition "c-lock-string" to Section 2.1. Added backward compatible definition "c-key-string" to Section 2.2.
- Use different wording in Section 2.2.
- Changed wording to reflect that an injecting agent is allowed to create Cancel-Lock headerfields in Section 2.1.
- Fixed wording and typo in Section 2.
- Added normative reference to RFC7405 because case-sensitivity is used in ABNF.
- Added reference to RFC5536 (Section 2.2) in Section 2.
- Added references to RFC4880 and RFC5537 in Section 1.
- Replaced the wordings "remove" by "cancel" and "replace" by "supersede".

- Modified header and abstract section to no longer list RFC5536 and RFC5537 as updated by this document.

B.4. Changes since -00

- Added additional note that deprecated "scheme" values should be preserved for backward compatibility as long as reasonable.
- Removed deprectated scheme "md5" (not in use anymore).
- Added descriptions how to generate "code-string" to Section 2.1 and Section 2.2.
- Removed length limitiation in ABNF of "scheme".
- Changed copyright notice to use text from TLP section 6.c.iii.
- Removed references from "abstract" section.
- Changed "SHOULD NOT" to uppercase in Section 6.
- Added line wraps to CLI commands in Section 5.

B.5. Changes since draft-ietf-usefor-cancel-lock-01

- Renamed document because the USEFOR IETF WG is now closed.
- Added more details how to check Cancel-Key header fields in Section 3.5.
- Added more details to Section 7.
- Added updated ABNF for Cancel-Lock and Cancel-Key header fields.
- Deprecated "md5" and "sha1" schemes.
- Added "sha-256" scheme.
- Reworded the abstract section and added references.
- Added note to other authentication systems to Section 1.
- Added command line check examples to Section 5.

B.6. Changes since draft-ietf-usefor-cancel-lock-00

- References to SHA-160 changed to SHA1
- "scheme" is now a case insensitive token and the number "1" has been changed to "sha1".
- Added some examples and fixed the section numbering.
- Updated 2nd paragraph on section 2.2 to make clear what exactly is being hashed and how.
- Changed paragraph 2 of 3.1 to discourage injection agents from adding the header.
- Removed the Clue-string as this complicated the scheme without adding realistic functionality
- Moderators can now add these headers under the same conditions as injection agents.

Author's Address

Michael Baeuerle
STZ Elektronik
Hofener Weg 33C
Remseck, Baden-Wuerttemberg 71686
Germany

Fax: +49 7146 999061
EMail: michael.baeuerle@stz-e.de