Internet Engineering Task Force (IETF)
Request for Comments: 8315
Updates: 5537
Category: Standards Track
ISSN: 2070-1721

M. Baeuerle
STZ Elektronik
February 2018

Cancel-Locks in Netnews Articles

Abstract

This document defines an extension to the Netnews Article Format that may be used to authenticate the withdrawal of existing articles. This document updates RFC 5537.

Status of This Memo

Copyright Notice

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 withdrawal of an article is valid; that is to say the poster, posting agent, moderator, or injecting agent that processed the original article has requested to withdraw it via the use of a cancel control article ([RFC5537] Section 5.3) or a Supersedes header field ([RFC5537] Section 5.4).

This document defines two new header fields: Cancel-Lock and Cancel-Key. The Cancel-Lock header field contains hashes of secret data. The preimages can later be used in the Cancel-Key header field to authenticate a cancel or supersede request.

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 articles to add or remove newsgroups ([RFC5537] Section 5.2), an OpenPGP [RFC4880] signature is appropriate.

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Header Fields

This section describes the formal syntax of the new header fields using ABNF [RFC5234]. Non-terminals not defined in this document are defined in Section 3 of [RFC5536].

The new header fields Cancel-Lock and Cancel-Key are defined by this document, extending the list of article header fields defined in [RFC5536].

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 [RFC6234]; see also [RFC1321] and [RFC6151] for MD5, [RFC3174] for SHA1, and [SHA] for the SHA2 family. The Base64 encoding used is defined in Section 4 of [RFC4648].

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

Because the hash algorithm for <scheme> cannot be negotiated, unnecessary proliferation of hash algorithms should be avoided. The hash algorithms "sha224" and "sha384" are only added to the "Netnews Cancel-Lock Hash Algorithms" registry (Section 8.3) because implementations exist that support them. Implementations SHOULD NOT use the hash algorithms "sha224" and "sha384" to generate <scheme>.

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 (comments and/or folding whitespace) 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-key) [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

Use cases:

- o The poster of an article wants to cancel or supersede existing articles.
- o A moderator wants the ability to cancel articles after approving them.
- o An injecting agent wants to act as a representative for a posting agent that has no support for the authentication system described in this document.
- o A news administrator wants the ability to cancel articles that were injected by its system (because, for example, they violate its abuse policy).

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 and 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, including one or more <c-lock> elements.

If multiple <c-lock> elements are added to the Cancel-Lock header field by a single agent, each <c-lock> element MUST use a unique key "K" to improve security.

If an injecting agent (or moderator) wants to act as a representative for a posting agent without support for the authentication system described in this document, then it MUST be able to positively authenticate the poster and MUST be able to automatically add a working Cancel-Key header field for all proto-articles with cancelling or superseding attempts from that poster.

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.

If multiple <c-lock> elements are appended to the Cancel-Lock header field by a single agent, each <c-lock> element MUST use a unique key "K" to improve security.

If an injecting agent (or moderator) wants to act as a representative for a posting agent without support for the authentication system described in this document, then the same requirements apply as those mentioned in Section 3.1.

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

The Cancel-Key header field contains one or more of the secret strings that were used to create the Cancel-Lock header field of the original article. Knowledge of at least one of the secret strings is required to create a match for successful authentication.

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 and 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 (acting as a representative for the posting agent) has added a Cancel-Lock header field to an article listed in the Control (with a "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 corresponds 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 (acting as a representative for the posting agent) has extended the Cancel-Lock header field in an article listed in the Control (with a "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 corresponds 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 relaying or serving agent receives an article that attempts to cancel or supersede a previous article via a 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 must contain 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 <c-key-string> part of the <c-key> element is hashed using the algorithm defined by its <scheme> part.

2. For each <c-lock> element with the same <scheme> in the original article, its <c-lock-string> part is compared to the calculated hash.

3. If a <c-lock-string> part is equal to the calculated hash, 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

The following algorithm is RECOMMENDED to calculate the key "K" based on a local secret <sec>.

The result of the function

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

is the key "K" for an article with a Message-ID <mid> that belongs to the User-ID (or UID) <uid> (e.g., the login name of the user). The Hashed Message Authentication Code (HMAC) is outlined in [RFC2104]. The 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.

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

The User-ID <uid> must not contain angle brackets (to ensure that concatenation of different <uid> and <mid> elements cannot give the same results).

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

In general, different values for the secret <sec> must be used if multiple <c-lock> elements are added by a single agent.

The local secret <sec> should have a length of at least the output size of the hash function that is used by the HMAC (256 bits / 32 octets for SHA256) and must be a cryptographically random value [RFC4086].

Note that the hash algorithm used as the base for the HMAC operation is not required to be the same as that 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 an empty string as <uid> (as recommended in Section 4 for posting agents):

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

mid: <12345@mid.example>
sec: ExampleSecret
K  : HMAC-SHA256(sec, mid) ;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 the 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 the 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 recommended in Section 4):

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

uid: JaneDoe
mid: <12345@mid.example>
sec: AnotherSecret
K  : HMAC-SHA256(sec, uid+mid) ;uid+mid 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 the 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 the article <12345@mid.example>, it looks like this:

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

5.3. Other Examples

Another 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
```

Remember that parsing for <scheme> must be case insensitive.

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 draft versions 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 Algorithms" registry (Section 8.3).

```
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 draft versions 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 authentication system defined in this document provides no integrity-checking properties. Arbitrary modifications can be applied to an article on its way through the network, regardless of the presence of a Cancel-Key header field. A serving agent that receives an article that contains a Cancel-Key header field with a matching <c-key> element only gets the information that the withdrawal of the target article was approved by a legitimate person or agent.

Example: A valid <c-key> element is extracted from a cancel control article and inserted into a forged supersede article. All servers on the network that receive the forged supersede article before the cancel control article should accept the forged supersede. But because everybody can post articles with forged identity information in the header (same as with spam email), the same result can be achieved by sending a forged new article using no authentication system at all.

For originator and integrity checks, a signature-based authentication system is required (normally, OpenPGP [RFC4880] is used for this purpose). Both systems can be combined.

The important property of the hash function used for <scheme> is the preimage resistance. A successful preimage attack either reveals the real Cancel-Key (that was used to create the Cancel-Lock of the original article) or gives a different Cancel-Key (that matches a Cancel-Lock too). This 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 arbitrary Cancel-Lock header field is not helpful to break the authentication system defined in this document (if a specific article is defined as the 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 previous specification of the authentication system defined in this document -- [USEFOR-CANCEL-LOCK] -- is nearly two decades old. The client-side implementations are moving forward extremely slowly, too (newsreaders from the last millennium are still in heavy use). What is defined today should be strong enough for the next two decades or so.

- The collision resistance of SHA1 is already broken; therefore, it is now obsolete for digital signatures as used in Transport Layer Security (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 the base) for the HMAC that is recommended 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 too recent and has not been studied enough to be considered more secure than SHA2.

The operation HMAC(sec, uid+mid) as recommended 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 needs 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 a greater margin for security here. Breaking <scheme> only allows the authentication of 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 were generated with this <sec> in the past. Changing <sec> at regular intervals can be used to mitigate potential damage.

If an agent adds or appends multiple <c-lock> elements, it must not use the same K for them (by using different secrets (<sec>)). Adding multiple <c-lock> elements with the same <scheme> and the same K makes no sense (because it would result in identical <c-lock> elements); therefore, the case of different <scheme> values is relevant: a preimage attack on the different hash algorithms may be easier if the attacker knows that the output of those hash algorithms was created with the same input.

If an implementation chooses to not implement the key calculation algorithm recommended in Section 4 or to implement it with the HMAC based on a different hash function than <scheme>, the key size used should match the output size of the hash function used for <scheme>.

8. IANA Considerations

IANA has registered the following header fields in the "Permanent Message Header Field Names" registry, 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): RFC 8315

Header field name: Cancel-Key
Applicable protocol: netnews
Status: standard
Author/change controller: IETF
Specification document(s): RFC 8315
```

The "Netnews Cancel-Lock Hash Algorithms" registry is maintained by IANA.

The registry is available at <https://www.iana.org/assignments/netnews-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 [RFC8126]. 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 hash algorithm (see the definition of <scheme> in Section 2) can be used; in particular, 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.

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

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 reason for this would 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 Section 7 of this document
Published specification: RFC 8315
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 Section 7 of this document
Published specification: RFC 8315
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: sha224
Security considerations: See Section 7 of this document
Published specification: RFC 8315
Contact for further information: Author of this document
Intended usage: LIMITED USE
Owner/Change controller: IESG <iesg@ietf.org>
Note: sha256 should be used instead; this is a truncated variant
```

Netnews Cancel-Lock hash algorithm name: sha256
Security considerations: See Section 7 this document
Published specification: RFC 8315
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: sha384
Security considerations: See Section 7 of this document
Published specification: RFC 8315
Contact for further information: Author of this document
Intended usage: LIMITED USE
Owner/Change controller: IESG <iesg@ietf.org>
Note: sha512 should be used instead; this is a truncated variant

Netnews Cancel-Lock hash algorithm name: sha512
Security considerations: See Section 7 of this document
Published specification: RFC 8315
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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://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, <https://www.rfc-editor.org/info/rfc3864>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5234] Crocker, D., Ed., and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc5537>.

[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, <https://www.rfc-editor.org/info/rfc6234>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

9.2. Informative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6151>.

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

[USEFOR-CANCEL-LOCK] Lyall, S., "Cancel-Locks in Usenet articles.", Work in Progress, draft-ietf-usefor-cancel-lock-01, November 1998.

Acknowledgements

The author acknowledges the original author of the Cancel-Lock authentication system, as documented in [USEFOR-CANCEL-LOCK]: Simon Lyall. Simon wrote the original document and approved the usage of his work for this document. This document is mostly based on his work. (It was originally intended as revision 02 but was 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, Urs Janssen, Richard Kettlewell, Marcel Logen, Holger Marzen, Dennis Preiser, and Emil Schuster. Thanks also to Peter Faust and Alfred Peters for providing statistical data about the algorithms currently in use.

Special thanks to the Document Shepherd, Julien Elie; and to the responsible Area Director, Alexey Melnikov.

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