

TCPM Working Group
Internet-Draft
Expires: June 5, 2006

B. Weis
C. Appanna
D. McGrew
A. Ramaiah
Cisco Systems
December 2, 2005

TCP Message Authentication Code Option
draft-weis-tcp-mac-option-00

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with [Section 6 of BCP 79](#).

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

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/1id-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on June 5, 2006.

Copyright Notice

Copyright (C) The Internet Society (2005).

Abstract

This memo describes a TCP [[RFC0793](#)] extension to enhance security for BGP [[I-D.ietf-idr-bgp4](#)] and other TCP-based protocols requiring message authentication. It provides message authentication using a Message Authentication Code (MAC), which is a superior authentication method to the keyed MD5 method previously used. The option also includes provision for automatic generation and distribution of MAC

keys. A set of MAC algorithms are specified, as well as guidance when to use each one.

Table of Contents

1.	Introduction	3
1.1.	Requirements notation	3
1.2.	Terminology	3
2.	Proposal	4
3.	Syntax	6
3.1.	Option Header	6
3.1.1.	MAC Algorithm ID Types	6
3.2.	Authentication Data	7
3.2.1.	Authentication Data Size	8
3.3.	Encrypted Key	8
3.3.1.	KEK Algorithm ID Types	8
4.	Discussion	10
4.1.	Key Management Methods	10
4.1.1.	Locally Managed MAC Keys	10
4.1.2.	Automatic Generation of Keys	10
4.2.	MAC Option Size	11
4.2.1.	Authentication Data Only	11
4.2.2.	Adding an Encrypted Key	11
5.	IANA Considerations	12
6.	Security Considerations	13
7.	References	14
7.1.	Normative References	14
7.2.	Informative References	14
	Authors' Addresses	16
	Intellectual Property and Copyright Statements	17

1. Introduction

The TCP MD5 Signature Option [[RFC2385](#)] specifies an option to provide integrity protection to BGP sessions using the MD5 algorithm and a key in a keyed MAC construction. MD5 does not provide a sufficient level of security, and recently published attacks motivate its replacement. Also, the keyed hash MAC construction used by [RFC2385](#) has serious cryptographic weaknesses. An attacker who can find a collision in the underlying hash function can forge a message authentication code using a simple chosen-message attack [[MACS](#)].

This proposal describes a set of message authentication codes used to verify integrity of a TCP message. Message authentication codes are a well-accepted method of verifying message integrity.

This proposal can use message authentication codes that require unique nonces. This is important because at present the best-performing MACs all have this requirement. The MAC Option can also securely transport new key material between end-points, using a long-lived encryption key. Neither of these options is mandatory to implement, in order to ensure that the MAC Option can easily be implemented in current TCP stacks.

1.1. Requirements notation

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](#)].

1.2. Terminology

Key Encrypting Key (KEK). A key used with a cryptographic algorithm to encrypt another key.

Message Authentication Code (MAC). A keyed cryptographic checksum on data that uses a secret key to detect modifications of a message (e.g., a TCP segment). An attacker who does not know the secret key is unable to generate the MAC corresponding to a particular message, with very high probability. This is true even when the attacker can perform a chosen-message attack, and cause a legitimate user of the system to authenticate messages of its choice.

2. Proposal

This proposal describes a means of using a Message Authentication Code (MAC) algorithm to authenticate a TCP segment. Modern MAC algorithms were developed specifically for detecting modification to data (e.g., TCP segments) and are significantly stronger cryptographic methods than the TCP MD5 Signature Option. A number of MAC algorithms may be used, each of which has different strengths and weaknesses (described later in this proposal). The key used to create the MAC is assumed to be known only to the TCP end-points.

The MAC authentication data is created by applying a MAC algorithm and a key to the following data:

1. the TCP pseudo-header (in the order: source IP address, destination IP address, zero-padded protocol number, and segment length)
2. the TCP header, excluding options, and assuming a checksum of zero
3. the TCP segment data (if any)

Creating a MAC in this manner, an attacker does not only need to create a valid TCP segment (including a valid TCP sequence number), but also must be able to create a valid MAC. Creating a valid MAC requires knowledge of the key used to create the MAC.

Unlike [RFC 2385](#), this proposal does not apply the MAC algorithm to the key in the same manner as the MAC algorithm is applied to the TCP pseudo-header, header, and data. The MAC algorithms defined for this option use the key as a direct input to the algorithm, so there is no need to additionally include it in the MAC data. Additionally, including the key in the hashed data may weaken the MAC. This is because using the key as part of the message interferes with the reduction-based methods used to design encryption algorithms and to prove their security [[KDM](#)].

This proposal includes an option for passing the MAC key in-band, encrypted under a Key Encrypting Key (KEK) known to both TCP end-points. When an encrypted key is included in a TCP option, it is used to authenticate the current segment, and all subsequent segments in the TCP exchange until a new key is chosen by one or the other of the TCP end-points. Algorithms used with the KEK MUST be strong enough so that an attacker observing the segment cannot determine the key in a reasonable amount of time. One strong KEK algorithm is described below.

Several methods of key management are possible with this proposal, and are discussed later in this document.

3. Syntax

This option has three parts: TCP option header, authentication data, and an optional encrypted key for use verifying integrity of this and future segments within this TCP flow.

3.1. Option Header

The option header has the following format:

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|      Kind      |      Length      | MAC Alg ID |K|      Key ID      |
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+

```

- o Kind (8 bits). Value that identifies this option as the Cryptographic Option.
- o Length (8 bits). Length in octets of the option, including its header.
- o MAC Alg ID (7 bits). Unique identifier describing the cryptographic algorithm and mode to be used as the MAC algorithm.
- o K (1 bit). When set to 1, signals that an encrypted key is included in the option.
- o Key ID (8 bits). A value identifying a particular key to both the sender and receiver. If K is set to one, the Key ID identifies the KEK used to protect encrypted MAC key. A value of 0 indicates that the key in use has no identifier.

3.1.1. MAC Algorithm ID Types

The following MAC Algorithms are strong MAC algorithms suitable for use with this option.

- o HMAC-SHA-1-96. HMAC with the SHA-1 hash function, with output truncated to 96 bits [[FIPS.198.2002](#)]. This algorithm MAY be implemented for an implementation to conform to this option.
- o AES-128-GMAC-96. AES [[FIPS.197.2001](#)] with a 128-bit key in the GMAC [[GMAC](#)] mode of operation, with a 96-bit tag. This algorithm requires a 96-bit unique nonce. The nonce is formed as follows. The leftmost 56 bits are all set to zero. The next eight bits contain a direction byte; its binary value is 00000000 for the sender, and 00000001 for the receiver. The rightmost 32 bits are

copied from the Sequence Number field. This algorithm MAY be implemented for an implementation to conform to this option.

- o AES-128-CMAC-96. AES with a 128-bit key in the CMAC mode of operation [[FIPS.800-38B](#)], with output truncated to 96 bits. The AES-128-CMAC-96 algorithm MUST be implemented for an implementation to conform to this option.
- o AES-128-UMAC-96. The UMAC-96 message authentication code [UMAC] with a 96-bit output tag. This algorithm also requires a nonce. For the purposes of this document the nonce will be 40 bit nonce. The nonce is formed as follows. The first eight bits contain a direction byte; its binary value is 00000000 for the sender, and 00000001 for the receiver. The rightmost 32 bits are copied from the Sequence Number field. This algorithm MAY be implemented for an implementation to conform to this option.

3.2. Authentication Data

The authentication data provides segment integrity. If the MAC algorithm does not requires a nonce, then the Authentication Data is simply comprised of the MAC bytes, as follows.

```

      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
~                               Message Authentication Code                               ~
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

- o Message Authentication Code (variable). The size of the MAC varies according to the MAC algorithm (see table at the end of this section).

If the MAC algorithm requires a nonce for its operation, it MUST be included at the beginning of the Authentication data, as follows.

```

      0              1              2              3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                               Sequence Number                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
~                               Message Authentication Code                               ~
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

- o Sequence Number (32 bits). A unique monotonically increasing value used as a base for a nonce for algorithms requiring a unique value for each ICV value generated with a particular key. When a sequence number reaches the maximum value, the key MUST NOT be

used to authenticate any further packets.

- o Message Authentication Code (variable). The size of the MAC varies according to the MAC algorithm (see table at the end of this section).

[3.2.1.](#) Authentication Data Size

The size of the authentication data field varies depending on the output of the MAC algorithm and whether or not the MAC algorithm requires a sequence number field. The following table lists the MAC algorithms identified in this proposal and the resulting size of the authentication data field.

MAC Algorithm	Authentication Data Size (bits)
HMAC-SHA-1-96	96
AES-128-GMAC-96	128
AES-128-CMAC-96	96
AES-128-UMAC-96	128

[3.3.](#) Encrypted Key

When K is set to 1, the encrypted key is included in the TCP MAC Option following the authentication data. The length of the key can be determined from the algorithm type, and need not be explicitly included in the option.

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
~   KEK Alg ID |                               Encrypted Key           ~
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+

```

- o KEK Alg ID (8 bits) -- This field contains an algorithm type to be used with the key encrypting key.
- o Encrypted Key (variable). The size of the encrypted key field depends upon the size of the encrypted key (see below).

[3.3.1.](#) KEK Algorithm ID Types

The following algorithms are suitable to be used with a key encrypting key.

AES-128-ECB. The MAC key is encrypted using an AES 128-bit key encrypting key, resulting in a 128-bit encrypted key. Use of ECB mode is acceptable because only one block is being encrypted. This algorithm MUST NOT be used to encrypt a MAC key larger than 128 bits. (An example of an algorithm that could be used to encrypt a MAC larger than 128 bits is the NIST Key Wrap [[KEYWRAP](#)].)

4. Discussion

4.1. Key Management Methods

Experience with key management for [RFC 2385](#) have shown that key management for TCP option keys is subject to varying operational constraints. A single pre-configured MAC key is neither secure, nor operationally sound. This proposal provides for several different methods of managing multiple MAC keys, depending on the operational requirements of the communicating TCP end-points. Two of these methods are described in detail below.

4.1.1. Locally Managed MAC Keys

In this method, two TCP end-points each configure one or more MAC keys. When keys have a unique Key ID, it is placed in the option header to declare which key was used to calculate the MAC. In order to guarantee that keys are not used longer than their cryptographic lifetime, keys must be periodically changed. Some methods of changing keys is described in [[I-D.ramaiah-key-rollover](#)].

When locally managed MAC keys are used, the proposed option only contains the option header and authentication data. K in the option header is never set to 1 when locally managed MAC keys are used. A MAC algorithm requiring a nonce MUST NOT be used in this mode due to the risk of re-using a nonce.

4.1.2. Automatic Generation of Keys

In this method, the two TCP end-points configure one or more KEKs having a long lifetime. A KEK is used to encrypt a randomly generated MAC key, which is then included in the Encrypted Key portion of the TCP option. This allows for the scheduled automatic generation of keys that can be periodically replaced or based on the policy of either TCP.

Each KEK has a Key ID associated with it. When a particular KEK is used to generate a MAC key, the TCP endpoint places its Key ID in the option header to declare which key was used to encrypt the MAC key.

When a TCP end-point begins to send with a new key, it SHOULD retain the previous key until the peer also begins to encrypt using the new key. Doing so allows a continued receipt of TCP segments from the peer, including ack messages indicating that the segment with the new MAC key was not received.

MAC algorithms requiring a nonce can be used safely, since the keys are randomly chosen.

[4.2.](#) MAC Option Size

The cumulative number of TCP option bytes is currently limited to 40 bytes. The TCP MAC Option can consume more or fewer bytes, depending on a number of factors. The following sections describe several scenarios.

[4.2.1.](#) Authentication Data Only

The option consumes four bytes for the option header. If K is not set to one, then the total size of the TCP MAC option is only the additional number of bytes needed by the MAC algorithm. All MAC algorithms described in this proposal not requiring a nonce require 12 bytes. This gives a total of 16 bytes for the TCP MAC option, which is four bytes less than used by [RFC 2385](#).

MAC algorithms requiring a nonce require an additional four bytes to carry a sequence number in the authentication data portion of the option. This results in a total of 20 bytes. However, MAC algorithms requiring a nonce tend to consume fewer software and/or hardware resources than other MAC algorithms. Using a MAC algorithm requiring a nonce trades off of an additional four bytes in the segment for a faster cryptographic algorithm.

[4.2.2.](#) Adding an Encrypted Key

If K is set to one, then the encrypted key field is added to the MAC option. This adds the ability to do in-band keying, and simplify key management operations, but with a cost of additional TCP option bytes. When an encrypted key is included, one byte is always needed to describe the KEK algorithm used to encrypt the MAC key. The KEK algorithm also determines the number of bytes needed for the encrypted MAC key. The one KEK algorithm defined in this proposal requires 16 bytes, which results in a total of 17 bytes for the encrypted key.

Thus, 33 bytes total bytes are required when paired with a MAC algorithm not needing a nonce (although 36 bytes may be used if padding is added). A total of 37 bytes are required when paired with a MAC algorithm needing a nonce (or 40 bytes if padding is added). However, the encrypted key is only required when one of the TCP endpoints requires a new key (i.e., at the start of a TCP session, or when the security policy mandates a change later on in the session.) All other segments in the TCP session contain only the Authentication Data portion, which remains a modest size.

5. IANA Considerations

The terms "Standards Action" and "Private Use" in this section indicate the policies described for these terms in [[RFC2434](#)].

A new TCP Option Kind value must be defined in the IANA TCP Parameters registry.

The option header contains an 8-bit message type, for which IANA is to create and maintain a registry entitled "MAC Algorithm IDs". This document defines the following message authentication code types:

MAC Algorithm ID	Value
-----	-----
RESERVED	0
HMAC-SHA-1-96	1
AES-128-GMAC-96	2
AES-128-CMAC-96	3
AES-128-UMAC-96	4
Standards Action	5-64
Private Use	65-127

The option header contains an 8-bit message type, for which IANA is to create and maintain a registry entitled "Key Encrypting Key Algorithm IDs". This document defines the following KEK types:

KEK Algorithm ID	Value
-----	-----
RESERVED	0
AES-128-ECB	1
Standards Action	2-128
Private Use	129-254

Note to RFC Editor: this section may be removed on publication as an RFC.

6. Security Considerations

This proposal describes a strong authentication method for authenticating TCP segments. It defines the use of cryptographic MAC algorithms, which are considered state-of-the-art. As such, their expected lifetime of usefulness extends for several years. But cryptographic algorithms have an effective lifetime, depending on advancing processor speed and cryptographic research. This proposal provides for the future addition of new MAC algorithms as they are needed.

MAC algorithms requiring a unique nonce per segments (e.g., AES-128-GMAC-96) MUST NOT be used with manually configured keys. If the sequence number used as an input to the nonce wraps (or is re-initializing after a system reboot), a single nonce would be used multiple times with a single key. This would cause a catastrophic cryptographic failure, with the amount of damage dependant upon the actual algorithm.

Management of [RFC 2385](#) keys has been a significant operational problem, both in terms of key synchronization and key selection. Current guidance [[RFC3562](#)] warns against sharing [RFC 2385](#) keys between systems, and recommends changing keys according to a schedule. The same general operational issues are relevant for the management of MAC keys. This proposal allows for in-band automatic re-keying, which provides the following key management benefits:

- o Automated key lifetime management. A system can rollover keys triggered by any means chosen by the system (e.g., volume lifetime, time lifetime).
- o Automated key selection. Keys chosen with a good random number generator are superior in quality to manually chosen keys.
- o Keys are chosen for use of a particular TCP session, and not shared between TCP session to different peers.

Use of in-line key management requires a static KEK with a long lifetime. Whereas the KEK needs to be changed periodically, the length of the period should be very long, compared to the lifetime of the MAC keys.

[7.](#) References

[7.1.](#) Normative References

- [FIPS.197.2001]
National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001, <<http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf>>.
- [FIPS.198.2002]
National Institute of Standards and Technology, "The Keyed-Hash Message Authentication Code (HMAC)", FIPS PUB 198, March 2002, <<http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf>>.
- [FIPS.800-38B]
National Institute of Standards and Technology, "Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication", FIPS PUB 800-38B, May 2005, <http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf>.
- [GMAC]
McGrew, D. and J. Viega, "Galois/Counter Mode of Operation (GCM)", Submission to NIST modes of operation, May 2005, <<http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-revised-spec.pdf>>.
- [RFC0793]
Postel, J., "Transmission Control Protocol", STD 7, [RFC 793](#), September 1981.
- [RFC2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.
- [RFC2434]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 2434](#), October 1998.

[7.2.](#) Informative References

- [I-D.ietf-idr-bgp4]
Rekhter, Y., "A Border Gateway Protocol 4 (BGP-4)", [draft-ietf-idr-bgp4-26](#) (work in progress), October 2004.
- [I-D.ramaiah-key-rollover]
Ramaiah, A., Mynam, S., and C. Appanna, "Key rollover schemes for TCP-based routing applications", [draft-ramaiah-key-rollover-00](#) (work in progress),

November 2005.

- [KDM] Black, J., Rogaway, P., and T. Shrimpton, "Encryption-Scheme Security in the Presence of Key-Dependent Messages", Selected Areas in Cryptography 2002 (SAC 2002), Lecture Notes in Computer Science, vol. 2595, Springer-Verlag, 2003., May 2002, <<http://www.cs.ucdavis.edu/~rogaway/papers/kdm.html>>.
- [KEYWRAP] "Draft NIST AES Key Wrap Specification", November 2001, <<http://csrc.nist.gov/CryptoToolkit/kms/key-wrap.pdf>>.
- [MACS] Bellare, M., Canetti, R., and H. Krawczyk, "Keying Hash Functions for Message Authentication", Proceedings of Crypto'96 , LNCS 1109, pp. 1-15., June 1996, <An extended version of this paper is available at <http://www.research.ibm.com/security/bck2.ps>>.
- [RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", [RFC 2385](#), August 1998.
- [RFC3562] Leech, M., "Key Management Considerations for the TCP MD5 Signature Option", [RFC 3562](#), July 2003.

Authors' Addresses

Brian Weis
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-526-4796
Email: bew@cisco.com

Chandrashekhar Appanna
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-526-6198
Email: achandra@cisco.com

David McGrew
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-301-349-5815
Email: mcgrew@cisco.com

Anantha Ramaiah
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1-408-525-6486
Email: ananth@cisco.com

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in [BCP 78](#) and [BCP 79](#).

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at <http://www.ietf.org/ipr>.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2005). This document is subject to the rights, licenses and restrictions contained in [BCP 78](#), and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.