Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 20, 2008

B. Weis
C. Appanna
D. McGrew
A. Ramaiah
Cisco Systems

October 18, 2007

Automated key selection extension for the TCP Enhanced Authentication Option
draft-weis-tcp-auth-auto-ks-03

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 April 20, 2008.

Copyright Notice

Copyright (C) The IETF Trust (2007).

Abstract

This memo describes an automated key selection extension for the TCP [RFC0793] authentication option [I-D.bonica-tcp-auth]. This key selection extension allows two TCP endpoints to authenticate TCP segments using a Message Authentication Code (MAC) key chosen dynamically by an endpoint, rather than using a pre-configured MAC key.

Table of Contents

1. Introduction

The TCP Enhanced Authentication Option [I-D.bonica-tcp-auth] specifies a means of providing integrity protection to BGP and other TCP-based routing protocols. It does this by applying a Message Authentication Code (MAC) to the TCP pseudo-header, TCP header, and TCP segment data (if any). Several allowed MAC algorithms are defined.

MAC algorithms take as input a secret key known to the two TCP endpoints, called a MAC key. The TCP Enhanced Authentication Option describes a means of organizing MAC keys in a "key set" associated with a peer TCP endpoint. These keys are chosen out of band, and manually entered into the configuration of the TCP endpoints.

This memo describes a means by which TCP endpoints choose MAC keys using an automated process, and is a more secure and operationally simpler method of key selection. The automatically generated keys are protected during transmission by a long-lived key encryption key (KEK) shared between the TCP endpoints.

This memo also specifies additional strong MAC algorithms that use unique nonces for each TCP segment. This is important because at present the best-performing MACs all have this requirement. MAC algorithms using nonces are only safe to use with an automatic key selection process. This is because an automatic key selection process can quickly and securely react to the condition that a non-unique nonce is about to be used.

Several alternative methods for automatically providing keys for the TCP Enhanced Authentication Option were considered and rejected. These methods and the rejection rationale are described in Appendix A of this document.

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 integrity function computed on data using a secret key to detect modifications of the data (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, or to modify the message in an undetectable fashion, with very high probability. This is true even if the attacker can perform a chosen-message attack, and cause a legitimate user of the system to authenticate messages of its choice.

Message Authentication Code Key (MAC Key). A key used to authenticate a TCP segment.

2. Automatic Key Selection Process

This memo specifies a method for a TCP endpoint to automatically generate a TCP Enhanced Authentication Option MAC key and pass it to a peer in-band. The MAC key is passed in the TCP Enhanced Authentication Option encrypted under a Key Encrypting Key (KEK) known to both TCP end-points. When an encrypted key is included in this 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 either of the TCP end-points. Key encryption algorithms and modes used with the KEK MUST be strong enough so that in-line transmission of the key does not degrade the security offered by the MAC algorithm. One strong KEK algorithm is described below.

Two TCP end-points configure one or more KEKs before the in-band key selection method is used. A KEK is never used directly as a MAC key because using a cryptographic key for multiple purposes (such as a KEK and a MAC key) may cause a cryptographic vulnerability and weaken the key. A KEK typically has a long lifetime.

When the automated key selection method is used, MAC keys are generated as needed using a strong random number generator. The KEK is used to encrypt the MAC key, and the resulting ciphertext is then included in the Encrypted Key portion of the TCP Enhanced Authentication Option. This approach allows for a scheduled automatic generation of keys that can be periodically replaced based on the policy of either TCP. Generating and distributing a MAC key requires no operator intervention on either TCP endpoint.

2.1. KEK Requirements

Before the extension in this memo can be used, both TCP endpoints must have obtained a KEK. The KEK is exchanged using an out-of-band process (e.g., manual configuration or using a separate protocol), which is not discussed in this memo. A TCP endpoint MAY exchange more than one KEK with a particular peer TCP endpoint for the purpose of automatic KEK rollover. However, any such rollover process is outside the scope of this memo. One possible method is described in [I-D.viswanathan-keyrollover].

The use of pair-wise automatically generated MAC Keys is especially powerful, because each side can choose independently when to begin using a new MAC Key for its outbound segments without requiring the peer to coordinate the MAC key rollover event.

2.2. MAC Key Generation

The TCP Enhanced Authentication Option defines a set of attributes for each MAC key. When this extension is used, the attributes are set as follows:

- key identifier i, chosen according to local policy
- Authentication algorithm L[i], chosen according to local policy
- Shared secret S[i] generated using a strong random number generator
- A[i], set according to whether the key is the active key
- E[i], set according to whether the key is an eligible key

2.3. Sender Operations

A TCP Endpoint choosing a new MAC Key uses the following step:

- Generates a MAC Key of the appropriate length using a strong random number generator. A random number generator approved for NIST PUB 140-2 [FIPS.140-2.AnnexC] SHOULD be used.
- Places the MAC Key into the key set as described above. The Key ID i is set to a Key Id value currently unused in the key set. L[i] is set to a chosen authentication algorithm.
- Creates a TCP Enhanced Authentication Option with the K bit set to 1, the Alg ID set to A[i], and the Key ID set to i.
- Adds an Authentication Data formed as described below.

When a TCP end-point sends a new key, it SHOULD leave the previous key in the key set and marked as Eligible until the peer also begins to encrypt using the new key. Doing so allows a continued receipt of TCP segments from the peer, including acknowledgment messages indicating that the segment with the new MAC key was not received.

2.4. Receiver Operations

A TCP Endpoint receiving a new MAC Key uses the following steps:

- Detects that the packet includes an encrypted MAC Key by observing that the K bit is set.

- o Extracts the new MAC Key by decrypting it with the KEK and verifying that the decrypted key is well formed (i.e., the KEK Algorithm ID is a known algorithm id, and the Reserved bits are set to zero).

- o Verifies that the MAC Key was used to authenticate the packet.

- o Places the MAC Key into its key set as described above. A[i] is set to be the authentication algorithm defined in the In-line Encrypted Key Payload. The Key ID i is set to the Key Id value defined in the In-line Encrypted Key Payload.

2.5. Authentication Data Format

The TCP Enhanced Authentication Option defines an Authentication Data field, which always contains at least a Message Authentication Code. When the automated key selection option is used, the Authentication Data field includes both the MAC and an In-line Encrypted Key Payload, as shown in the following figure.

```
 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                  ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 In-line Encrypted Key Payload                 ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Message Authentication Code field contains the output of the MAC algorithm. Its size is deterministic based on MAC algorithm. The In-line Encrypted Key Payload is constructed 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|Res|KEK Alg ID |Res|KEK Key ID |    Encrypted Key              ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

- o Res (2 bits) -- Reserved bits, set to zero.

- o KEK Alg ID (6 bits) -- This field contains an algorithm identifier to be used with the key encrypting key.

- o Res (2 bits) -- Reserved bits, set to zero.

- o KEK Key ID (6 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).

2.5.1. KEK Algorithm ID Types

The MAC algorithms described in [I-D.bonica-tcp-auth] and this memo all use a key of 128-bits or smaller. The following algorithm is suitable to be used as a key encrypting key for these key sizes:

- o 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.

If a MAC algorithm requiring a key of larger than 128 bits is defined for use with this automated key selection extension, then a different key encrypting key algorithm will be required. Two possible methods are defined in [RFC3394] and [RFC3537].

3. MAC Algorithms using Nonces

All MAC algorithms take two types of inputs: the data to be authenticated, and the key. Many MAC algorithms (e.g., AES-128-CMAC-96 and HMAC-SHA-1-96) take only these inputs, which results in an Authentication Data field of the size of the resulting MAC. However, another class of MAC algorithms takes an additional input called a "nonce". Algorithms requiring a nonce tend to be better performing MAC algorithms, and thus have value when used with the TCP Enhanced Authentication Option.

A nonce permutes the output of a MAC algorithm such that it returns a unique value for each ICV value generated with a particular key and nonce pair. However, a particular key and nonce pair MUST NOT be used to authenticate two different sets of data. Doing so may weaken the MAC such that an attacker is able to generate properly formed MACs, which is a catastrophic cryptographic failure. Note that this restriction results in the requirement that a single MAC key MUST NOT be used to protect more than one TCP session. In order to guarantee that nonces used with a particular MAC key are unique, a monotonically increasing sequence number is included in the nonce.

3.1. Authentication Data Format

If the MAC algorithm requires a nonce for its operation, the sequence number part of the nonce 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 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. The first sequence number used with a particular MAC key is typically 1, although it MAY start a higher value. When a sequence number reaches 2**32-1, 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 definition (see table in a later section). There are no restrictions on the size of the Message Authentication Code field. In all cases, the MAC

algorithm definition must produce a result that is a multiple of 8 bits.

When a MAC algorithm requiring a nonce is used with a TCP Extended Authentication Option where K is 1, the Authentication Data field is as follows, with each field defined as described above:

```
 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                 ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
~                  In-line Encrypted Key Payload                ~
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2. MAC Algorithm ID Types

All Algorithm IDs described in the TCP Enhanced Authentication Option document are suitable for use with this option. Additionally, the following nonce based MAC algorithms are defined.

- AES-128-GMAC-96. AES [FIPS.197.2001] with a 128-bit key in the GMAC [GMAC] mode of operation, with the result truncated to 96 bits. 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. The binary value of the direction byte is 00000000 for the TCP endpoint sending the segment containing the encrypted key, and 00000001 for the TCP endpoint receiving the segment containing the encrypted key. The rightmost 32 bits are copied from the Sequence Number field. The AES-128-GMAC-96 algorithm MUST be implemented for an implementation to conform to this specification.

- AES-128-UMAC-96. The UMAC-96 message authentication code [UMAC] with the result truncated to 96 bits. This algorithm also requires a nonce. For the purposes of this document the nonce will be a 40 bit nonce. The nonce is formed as follows. The first eight bits contain a direction byte. The binary value of the direction byte is 00000000 for the TCP endpoint sending the segment containing the encrypted key, and 00000001 for the TCP endpoint receiving the segment containing the encrypted key. The rightmost 32 bits are copied from the Sequence Number field.

4. Discussion

4.1. MAC Option Size

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

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-CMAC-96	96
AES-128-GMAC-96	128
AES-128-UMAC-96	128

4.1.1. Authentication Data Only

The TCP Enhanced Authentication 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 the TCP Enhanced Authentication Option and this memo require 12 bytes. This gives a total of 16 bytes for the TCP MAC option.

MAC algorithms requiring a nonce need 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 an additional four bytes in the segment for a faster cryptographic algorithm.

4.1.2. 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, two bytes are always needed to describe the KEK algorithm and KEK Key Identifier 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 18 bytes for the encrypted key.

Thus, 34 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 38 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 end-points 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.

Additional KEK methods that require fewer bytes passed in the In-line Encrypted Key Payload may be defined at a later time, which would reduce the use of TCP Option bytes.

4.2. Use of the TCP Sequence Number as a MAC Algorithm Nonce

Using an additional four TCP option bytes for a sequence number dedicated to the MAC option is required in order to satisfy the cryptographic requirement of unique nonces. No other value in a TCP packet is guaranteed to be unique. At first glance, the TCP Sequence Number would appear to be suitable. However, the TCP Sequence Number can wrap, after which it increments back through the same sequence number space.

A security system should not depend on an external value when it can be manipulated such that the security constraint of the system is violated. This sort of dependency greatly increases the size of the security boundary (that is, the logical boundary containing all of the security functionality), which makes the validation of the correctness of the security system much more difficult.

In this case, the TCP Sequence Number is a value that can be manipulated elsewhere by the TCP module such that it is not actually unique enough for the security constraint. For example, some TCP redundancy solutions may resend TCP segments starting with the same TCP sequence number but with a different length. This violates the security requirement that a key and nonce are never used on TCP segments with different data.

In summary, the TCP Sequence Number is not suitable for use a MAC algorithm nonce value.

4.3. Retention of automatically generated keys

Automatically generated keys MUST NOT be set as Active (i.e., used for sending) after their lifetime has expired. The expired keys MAY be retained and marked as Eligible for a period of time, as defined by local policy. This is useful for continued receipt of TCP segments from the peer while the new key is being propagated. For example, the TCP endpoint may need to receive acknowledgment messages indicating that the segment with the new MAC key was not received.

Automatically generated keys SHOULD NOT be saved over a reboot. If this advice is ignored, a nonce containing a sequence number greater than the most recently used sequence number MUST be stored with the key. However, a more reliable system would simply generate a new MAC key (and associated nonce, if required) when the system resumes operation.

4.4. TCP sequence number wrapping

When a TCP sequence number wraps around (i.e., from a high number to a low number), an automatically generated key MUST be expired irrespective of lifetime policy and replaced with a new key. If the old key were not expired, there is a slight possibility that the TCP sequence numbers in the segment will both wrap, and both appear to be current in the TCP window. In this case, the segment may be accepted by the receiver as a new segment. Should the replayed segment contain an encrypted MAC key, and if the KEK has not changed, then the receiver will install the old key and no longer communicate properly with the authentic sender of the TCP segments.

5. IANA Considerations

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

The TCP Enhanced Authentication Code header includes an Algorithm ID field. The following two new Algorithm ID types are defined in this document, which require values be assigned to them: AES-128-GMAC-96, AES-128-UMAC-96.

The In-line Encrypted Key Payload contains an Algorithm ID, for which IANA is to create and maintain a registry entitled "Key Encrypting Key Algorithm IDs". This document defines the following initial set of IDs:

KEK Algorithm ID	Value
RESERVED	0
AES-128-ECB	1
Standards Action	2-47
Private Use	48-63

6. Security Considerations

This proposal allows for automatic re-keying for the TCP Enhanced Authentication Option, 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). However, the effective lifetime of a MAC key is more likely to be terminated by the event of a TCP sequence number wrapping, as described in Section 4.4.

- o Automated key selection. Keys chosen with a good random number generator are generally superior in quality to keys chosen by a human operator.

- o Keys are chosen for use of a particular TCP session, and cannot be shared between TCP session to different peers.

Use of automatic key selection 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. Because of the long lifetime, human interaction with the key is unnecessary after initial configuration, other than to verify that the key is entered correctly. This KEK SHOULD be protected in non-volatile storage such that it is not subsequently available except to the TCP Enhanced Authentication Option. Doing so also reduces the likelihood that it requires changing (e.g., due to operator staff turnover).

MAC algorithms requiring a unique nonce per segment (e.g., AES-128-GMAC-96) SHOULD NOT be used be used with manually configured MAC keys. If the sequence number used as an input to the nonce wraps (or is re-initialized 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.

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>.

[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>.

[I-D.bonica-tcp-auth]
Bonica, R., "Authentication for TCP-based Routing and Management Protocols", draft-bonica-tcp-auth-06 (work in progress), February 2007.

[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.

[UMAC] Black, J., Halevi, S., Krawczyk, H., Krovetz, T., and P. Rogaway, "UMAC: Fast and Secure Message Authentication", Advances in Cryptography -- CRYPTO '99 , September 1999, <http://www.cs.ucdavis.edu/~rogaway/umac/umac_full.pdf>.

7.2. Informative References

[FIPS.140-2.AnnexC]
National Institute of Standards and Technology, "Annex C: Approved Random Number Generators for FIPS PUB 140-2, Security Requirements for Cryptographic Modules", FIPS PUB 140-2 Annex C, January 2005, <http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf>.

[I-D.viswanathan-keyrollover]
Viswanathan, S., "Authentication-Key Rollover mechanism for Routing and Management Protocols", draft-viswanathan-keyrollover-00 (work in progress),

October 2006.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, September 2002.

[RFC3537] Schaad, J. and R. Housley, "Wrapping a Hashed Message Authentication Code (HMAC) key with a Triple-Data Encryption Standard (DES) Key or an Advanced Encryption Standard (AES) Key", RFC 3537, May 2003.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

Appendix A. Rejected Alternatives

This draft discusses a means to exchange encrypted keys between TCP endpoints. Several alternatives have been suggested. This section describes those alternatives as well as the rationale by which they were rejected.

Any method of generating keys for use by the TCP Enhanced Authentication option must be implementable within a TCP stack without depending on external management protocols. Therefore, the approach must be relatively simple, yet provide good quality encryption keys in a secure manner. The following methods partially meet this criteria but have flaws that result in their rejection.

A.1. Deriving session keys from a master key

A TCP endpoint could store a long-term master key used to derive session keys. Session keys would be derived heuristically (e.g., using a one-way hash chain) to create a set of ordered keys. This would have the advantage of not needing to pass the session key in a packet between routers.

However, in order to support his method a router would be required to store the position in a sequence to identify previously used keys. This is necessary in order to avoid re-using keys. While that requirement may not initially seem onerous, it should be noted that router configurations are generally stored on media that is not intended to be written frequently (e.g., NVRAM, flash memory). Therefore, reliable storage of ephemeral information (such as the position in a sequence) is problematic. A failure to store the most recently used key would result in a catastrophic security failure, and thus this method is rejected.

A.2. Deriving session keys using the Diffie-Hellman algorithm

It would be possible for a pair of TCP endpoints to use a Diffie-Hellman based protocol to derive session keys. However, since the Diffie-Hellman public numbers would be passed in the first two segments of an exchange, some other security mechanism (e.g., a long-term shared secret) would be necessary to protect the first two segments in the stream.

Diffie-Hellman public numbers with adequate security to derive a 128-bit AES key have been estimated at 3200 bits (400 bytes) [RFC3766]. Numbers this large can consume too many bytes to be effectively transferred in a TCP option. Also, computing the Diffie-Hellman algorithm shared secret during the initial handshake of every BGP session is too much overhead for the control plane of the router. It

is clear that any in-band method based on passing Diffie-Hellman numbers is not feasible.

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

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

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.

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, THE IETF TRUST 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.

Intellectual Property

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).