Network Working Group
Internet Draft
Intended Status: Informational
Expires: December 14, 2009

K.M. Igoe
National Security Agency
June 12, 2009
J.A. Solinas
National Security Agency
June 12, 2009

AES Galois Counter Mode for the Secure Shell Transport Layer Protocol
draft-igoe-secsh-aes-gcm-03

Status of this Memo

This Internet-Draft will expire on December 14, 2009.

Copyright Notice

Abstract

Secure Shell (SSH, RFC 4251) is a secure remote-login protocol. SSH provides for algorithms that provide authentication, key agreement, confidentiality and data integrity services. The purpose of this document is to show how the AES Galois/Counter Mode can be used to provide both confidentiality and data integrity to the SSH Transport Layer

Table of Contents

1. Introduction

Galois/Counter Mode (GCM) is a block cipher mode of operation that provides both confidentiality and data integrity services. GCM use counter mode to encrypt the data, an operation that can be efficiently pipelined. Further, GCM authentication uses operations that are particularly well suited to efficient implementation in hardware, making it especially appealing for high-speed implementations, or for implementations in an efficient and compact circuit. The purpose of this document is to show how GCM with either AES-128 or AES-256 can be integrated into the Secure Shell Transport Layer Protocol, RFC 4253.

2. Requirements Terminology

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].

3. Applicability Statement

Using AES-GCM to provide both confidentiality and data integrity is generally more efficient than using two separate algorithms to provide these security services.

4. Properties of Galois Counter Mode

Galois Counter Mode (GCM) is a mode of operation for block ciphers which provides both confidentiality and data integrity. NIST Special Publication SP 800 38D [GCM] gives an excellent explanation of Galois Counter Mode. In this document we shall focus on AES GCM, the use of the Advanced Encryption Algorithm (AES) in Galois Counter Mode. AES-GCM is an example of an "algorithm for authenticated encryption with associated data" (AEAD algorithm) as described in [RFC5116].

4.1. AES GCM Authenticated Encryption

An invocation of AES GCM to perform an authenticated encryption has the following inputs and outputs:

```
GCM Authenticated Encryption

    Inputs:
       octet_string PT ;   // Plain text, to be both
                           //    authenticated and encrypted
       octet_string AAD;   // Additional Authenticated Data,
                           //    authenticated but not encrypted
       octet_string IV;    // Initialization vector
       octet_string BK;    // Block cipher key

    Outputs:
       octet_string  CT;   // Cipher Text
       octet_string  AT;   // Authentication Tag
```

Note: in [RFC5116] the IV is called the nonce.

For a given block cipher key BK it is critical that no IV be used more than once. Section 7.1 addresses how this goal is to be achieved in secure shell.

4.2. AES GCM Authenticated Decryption

An invocation of AES GCM to perform an authenticated decryption has the following inputs and outputs:

```
GCM Authenticated Decryption

    Inputs:
       octet_string CT ;   // Cipher text, to be both
                           //    authenticated and decrypted.
       octet_string AAD;   // Additional Authenticated Data,
                           //    authenticated only.
       octet_string AT;    // Authentication Tag
       octet_string IV;    // Initialization vector
       octet_string BK;    // Block cipher key.

    Output:
       Failure_Indicator;  // Returned if the authentication tag
                           //   is invalid.
       octet_string  PT;   // Plain test, returned if and only if
```

```
                                      //    the authentication tag is valid.
```

AES-GCM is prohibited from returning any portion of the plaintext until the authentication tag has been validated. Though this feature greatly simplifies the security analysis of any system using AES-GCM, as we shall see in section 7.2, this creates an incompatibility with the requirements of secure shell.

5. Review of Secure Shell

The goal of secure shell is to establish two secure tunnels between a client and a server, one tunnel carrying client-to-server communications and the other server-to-client communications. Each tunnel is encrypted and a message authentications code is used to insure data integrity.

5.1. Key Exchange

These tunnels are initialized using the secure shell key exchange protocol as described in section 7 of [RFC4253]. This protocol negotiates a mutually acceptable set of cryptographic algorithms, and produces a secret value K and an exchange hash H shared by the client and server. The initial value of H is saved for use as the session_id.

If AES-GCM is selected as the encryption algorithm for a given tunnel, AES-GCM MUST also be selected as the mac algorithm. Conversely, if AES-GCM is selected as the mac algorithm, it MUST also be selected as the encryption algorithm.

As described in section 7.2 of [RFC4253], a hash based key derivation function (KDF) is applied to the shared secret value K to generate the required symmetric keys. Each tunnel gets a distinct set of symmetric keys. The keys are generated as shown in figure 1. The sizes of these keys varies depending upon which cryptographic algorithms are being used.

```
      Initial IV
         Client-to-Sever     HASH( K || H ||"A"|| session_id)
```

```
      Server-to-Client    HASH( K || H ||"B"|| session_id)
   Encryption Key
      Client-to-Sever     HASH( K || H ||"C"|| session_id)
      Server-to-Client    HASH( K || H ||"D"|| session_id)
   Integrity Key
      Client-to-Sever     HASH( K || H ||"E"|| session_id)
      Server-to-Client    HASH( K || H ||"F"|| session_id)
```

Figure 1: Key Derivation in Secure Shell

As we shall see below, SSH AES-GCM requires a 12-octet Initial IV and an encryption key of either 16 or 32 octets. Because an AEAD algorithm such as AES-GCM uses the encryption key to provide both confidentiality and data integrity, the integrity key is not used with AES-GCM.

Either the server or client may at any time request that the secure shell session be rekeyed. The shared secret value K, the exchange hash H, and all the above symmetric keys will be updated. Only the session_id will remain unchanged.

5.2. Secure Shell Binary Packets

Upon completion of the key exchange protocol, all further secure shell traffic is parsed into a data structure known as a secure shell binary packet as shown below in Figure 2 (see also section 6 of [RFC4253]).

```
   uint32    packet_length;  // 0 <= packet_length < 2^32
   byte      padding_length; // 4 <= padding_length < 256
   byte[n1]  payload;        // n1 = packet_length-padding_length-1
   byte[n2]  random_padding; // n2 = padding_length
   byte[m]   mac;            // m  = mac_length
```

Figure 2: Structure of a Secure Shell Binary Packet

The authentication tag produced by AES-GCM authenticated encryption will be placed in the mac field at the end of the secure shell binary packet.

6. AES GCM Algorithms for Secure Shell

6.1. AEAD_AES_128_GCM

AEAD_AES_128_GCM is specified in section 5.1 of [RFC5116]. Due to the of format of secure shell binary packets, the buffer sizes needed to implement AEAD_AES_128_GCM are smaller than those required in [RFC5116]. Using the notation defined in [RFC5116], the input and output lengths for AEAD_AES_128_GCM in secure shell are as follows:

PARAMETER	Meaning	Value
K_LEN	AES key length	16 octets
P_MAX	maximum plaintext length	2^32 - 32 octets
A_MAX	maximum additional authenticated data length	4 octets
N_MIN	minimum nonce (IV) length	12 octets
N_MAX	maximum nonce (IV) length	12 octets
C_MAX	maximum cipher length	2^32 octets

6.2. AEAD_AES_256_GCM

AEAD_AES_256_GCM is specified in section 5.1 of [RFC5116]. Due to the of format of secure shell binary packets, the buffer sizes needed to implement AEAD_AES_256_GCM are smaller than those required in [RFC5116]. Using the notation defined in [RFC5116], the input and output lengths for AEAD_AES_256_GCM in secure shell are as follows:

PARAMETER	Meaning	Value
K_LEN	AES key length	32 octets
P_MAX	maximum plaintext length	2^32 - 32 octets
A_MAX	maximum additional authenticated data length	4 octets
N_MIN	minimum nonce (IV) length	12 octets
N_MAX	maximum nonce (IV) length	12 octets
C_MAX	maximum cipher length	2^32 octets

6.3. Size of the Authentication Tag

Both AEAD_AES_128_GCM and AEAD_AES_256_GCM produce a 16-octet Authentication Tag ([RFC5116] calls this a "message authentication code".) Some applications allow use of a truncated version of this tag. This is not allowed in AES-GCM secure shell. All implementations of AES-GCM secure shell MUST use the full 16-octet Authentication Tag.

7. Processing Binary Packets in AES-GCM Secure Shell

7.1. IV and Counter Management

With AES-GCM, the 12-octet IV is broken into two fields: a 4-octet fixed field and an 8-octet invocation counter field. The invocation field is treated as a 64-bit integer and is incremented after each invocation of AES-GCM to process a binary packet.

```
uint32  fixed;                  // 4 octets
uint64  invocation_counter;     // 8 octets
```

Figure 3: Structure of an SSH AES-GCM nonce

AES-GCM produces a keystream in blocks of 16-octets which is used to encrypt the plaintext. This keystream is produced by encrypting the following 16-octet data structure:

```
uint32  fixed;                  // 4 octets
uint64  invocation_counter;     // 8 octets
uint32  block_counter;          // 4 octets
```

Figure 4: Structure of an AES input for SSH AES-GCM

The block_counter is initially set to one (1) and incremented as each block of key is produced.

The reader is reminded that SSH requires that the data to be encrypted MUST be padded out to a multiple of the block size (16-octets for AES-GCM).

7.2. Formation of the Binary Packet

In AES-GCM secure shell, the inputs to the authenticated encryption are:

```
PT (Plain Text)
   byte      padding_length; // 4 <= padding_length < 256
   byte[n1]  payload;        // n1 = packet_length-padding_length-1
   byte[n2]  random_padding; // n2 = padding_length
```

AAD (Additional Authenticated Data)
   uint32   packet_length; // 0 <= packet_length < 2^32
IV (Initialization Vector)
   As described in section 7.1.
BK (Block Cipher Key)
   The appropriate Encryption Key formed during the Key Exchange.

As required in [RFC4263], the random_padding MUST be at least 4 octets in length but no more than 255 octets. The total length of the PT MUST be a multiple of 16-octets (the block size of AES).

The binary packet is the concatenation of the 4-octet packet_length, the cipher text CT, and the 16-octet authentication tag AT.

7.3. Treatment of the Packet Length Field

Section 6.3 of [RFC4253] requires that the packet length, padding length, payload and padding fields of each binary packet be encrypted. This presents a problem for SSH AES-GCM because:

1) The tag can not be verified until we parse the binary packet
2) The packet can not be parsed until the packet_length has been decrypted.
3) The packet_length can not be decrypted until the tag has been verified.

When using AES-GCM with secure shell, the packet_length field is to be treated as additional authenticated data, not as plaintext. This violates the requirements of [RFC4253]. The repercussions of this decision are discussed in the security considerations section of this document.

8. Security Considerations

The security considerations in [RFC4251] apply.

8.1. Use of the Packet Sequence Number in the AT

[RFC4253] requires that the formation of the AT involve the packet sequence_number, a 32-bit value that counts the number of binary packets that have been sent on a given SSH tunnel. Since the sequence_number is, up to an additive constant, just the low 32-bits of the invocation_counter, the presence of the invocation_counter field in the IV insures that the sequence_number is indeed involved in the formation of the integrity tag, though this involvement differs slightly from the requirements in section 6.4 of [RFC4253].

8.2. Non-encryption of Packet Length

As discussed in section 5.2.1, there is an incompatibility between GCM's requirement that no plaintext be returned until the authentication tag has been verified, secure shell's requirement that the packet length be encrypted, and the necessity of decrypting the packet length field to locate the authentication tag. This document addresses this dilemma by requiring that, in AES-GCM, the packet length field will not be encrypted but will instead be processed as Additional Authenticated Data.

In theory, one could argue that encryption of the entire binary packet means that the secure shell dataflow becomes a featureless octet stream. But in practice, the secure shell dataflow will come in bursts, with the length of each burst strongly correlated to the length of the underlying binary packets. Encryption of the packet length does little in and of itself to disguise the length of the underlying binary packets. Secure shell provides two other mechanisms, random padding and SSH_MSG_IGNORE messages, that are far more effective than encrypting the packet length in masking any structure in the underlying plaintext stream that might be revealed by the length of the binary packets.

9. IANA Considerations

IANA will add the following two entries to the Secure Shell Encryption Algorithm name Registry described in [RFC4250]:

Name	Reference
AEAD_AES_128_GCM	Section 6.1

AEAD_AES_256_GCM	Section 6.2

IANA will add the following two entries to the Secure Shell MAC Algorithm name Registry described in [RF4250]:

Name	Reference
AEAD_AES_128_GCM	Section 6.1
AEAD_AES_256_GCM	Section 6.2

10. References

10.1. Normative References

[GCM] Dworkin, M, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-30D, November 2007.

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

[RFC4250] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

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

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryptions", RFC 5116, January 2008.

Author's Addresses

Kevin M. Igoe
NSA/CSS Commercial Solutions Center
National Security Agency
EMail: kmigoe@nsa.gov

Jerome A. Solinas
National Information Assurance Research Laboratory
National Security Agency
EMail: jasolin@orion.ncsc.mil

Acknowledgement

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