RSA LABORATORIES

PKCS #11 v2.20 Amendment 5 - Draft 1

Additional PKCS#11 Mechanisms

RSA Laboratories

April 15, 2008

Editor's note: Comments and feedback are welcome and should be sent to the Cryptoki mailing list (Cryptoki@rsasecurity.com) or the editor (pkcs-editor@rsasecurity.com)

Table of Contents

List of Tables

1 Introduction

This document is an amendment to PKCS #11 v2.20 () and describes extensions to

PKCS #11 to support additional mechanisms.

2 Definitions

AES Advanced Encryption Standard, as defined in FIPS PUB 197 .

3 Mechanisms

The following table shows, for the mechanisms defined in this document, their support by different cryptographic operations. For any particular token, of course, a particular operation may well support only a subset of the mechanisms listed. There is also no guarantee that a token that supports one mechanism for some operation supports any other mechanism for any other operation (or even supports that same mechanism for any other operation).

Table 1, Mechanisms vs. Functions

	Functions						
Mechanism	**Encrypt & Decrypt**	**Sign & Verify**	**SR & VR**[1]	**Digest**	**Gen. Key/ Key Pair**	**Wrap & Unwrap**	**Derive**
CKM_AES_GCM	✓						
CKM_AES_CCM	✓						

[1] SR = SignRecover, VR = VerifyRecover

The remainder of this section will present in detail the mechanisms and the parameters which are supplied to them.

3.1 Additional AES Mechanisms

3.1.1 Definitions

Mechanisms:

```
CKM_AES_GCM
CKM_AES_CCM
```

3.1.2 AES Mechanism parameters

- **CK_AES_GCM _PARAMS; CK_AES_GCM _PARAMS_PTR**

CK_AES_GCM_PARAMS is a structure that provides the parameters to the **CKM_AES_GCM** mechanism. It is defined as follows:

```
typedef struct CK_AES_GCM_PARAMS {
  CK_BYTE_PTR pIv;
  CK_ULONG ulIvLen;
  CK_ULONG ulIvBits;
  CK_BYTE_PTR pAAD;
  CK_ULONG ulAADLen;
  CK_ULONG ulTagBits;
} CK_AES_GCM_PARAMS;
```

The fields of the structure have the following meanings:

pIv	pointer to initialization vector
ulIvLen	length of initialization vector in bytes
ulIvBits	length of the initialization vector in bits, can be any number between 1 and 2^{64}. 96-bit IV values can be processed more efficiently, so that length is recommended for situations in which efficiency is critical.
pAAD	pointer to additional authentication data. This data is authenticated but not encrypted.
ulAADLen	length of *pAAD* in bytes.
ulTagBits	length of authentication tag (output following cipher text) in bits. Can be any value between 0 and 128.

CK_AES_GCM_PARAMS_PTR is a pointer to a **CK_AES_GCM_PARAMS**.

- **CK_AES_CCM _PARAMS; CK_AES_CCM _PARAMS_PTR**

CK_AES_CCM_PARAMS is a structure that provides the parameters to the **CKM_AES_CCM** mechanism. It is defined as follows:

```
typedef struct CK_AES_CCM_PARAMS {
   CK_ULONG ulDataLen; /*plaintext or ciphertext*/
   CK_BYTE_PTR pNonce;
   CK_ULONG ulNonceLen;
   CK_BYTE_PTR pAAD;
   CK_ULONG ulAADLen;
   CK_ULONG ulMACLen;
} CK_AES_CCM_PARAMS;
```

The fields of the structure have the following meanings, where L is the size in bytes of the data length's length ($2 < L < 8$):

ulDataLen	length of the data where $0 <= ulDataLen < 2^{8L}$.

pNonce	the nonce.
ulNonceLen	length of *pNonce* (<= 15-L) in bytes.
pAAD	Additional authentication data. This data is authenticated but not encrypted.
ulAADLen	length of *pAuthData* in bytes.
ulMACLen	length of the MAC (output following cipher text) in bytes. Valid values are 4, 6, 8, 10, 12, 14, and 16.

CK_AES_CCM_PARAMS_PTR is a pointer to a **CK_AES_CCM_PARAMS**.

3.1.3 AES-GCM authenticated Encryption / Decryption

Generic GCM mode is described in . To set up for AES-GCM use the following process, where *K* (key) and *AAD* (additional authenticated data) are as described in .

Encrypt:

- Set the IV length *ulIvLen* and bit count *ulIvBits* in the parameter block.
- Set the IV data *pIv* in the parameter block. *pIV* may be NULL *if ulIvLen* is 0.
- Set the AAD data *pAAD* and size *ulAADLen* in the parameter block. *pAAD m*ay be NULL if *ulAADLen* is 0.
- Set the tag length *ulTagBits* in the parameter block.
- Call C_EncryptInit() for **CKM_AES_GCM** mechanism with parameters and key *K*.
- Call C_Encrypt(), or C_EncryptUpdate()*[1] C_EncryptFinal(), for the plaintext obtaining ciphertext and authentication tag output.

Decrypt:

- . Set the IV length *ulIvLen* and bit count *ulIvBits* in the parameter block.
- Set the IV data *pIv* in the parameter block. *pIV* may be NULL *if ulIvLen* is 0.
- Set the AAD data *pAAD* and size *ulAADLen* in the parameter block. *pAAD m*ay be NULL if ulAADLen is 0.
- Set the tag length *ulTagBits* in the parameter block.
- Call C_DecryptInit() for **CKM_AES_GCM** mechanism with parameters and key *K*.
- Call C_Decrypt(), or C_DecryptUpdate()*[1] C_DecryptFinal(), for the ciphertext, including the appended tag, obtaining plaintext output.

[1] "*" indicates 0 or more calls may be made as required

In *pIv* the least significant bit of the initialization vector is the rightmost bit and the initialization vector bits are the rightmost *ulIvBits* bits. *ulIvLen* is the length of the initialization vector in bytes and must be >= (*ulIvBits* + 7) / 8.

The tag is appended to the cipher text and the least significant bit of the tag is the rightmost bit and the tag bits are the rightmost *ulTagBits* bits.

The key type for *K* must be compatible with **CKM_AES_ECB** and the C_EncryptInit/C_DecryptInit calls shall behave, with respect to *K*, as if they were called directly with **CKM_AES_ECB**, *K* and NULL parameters.

3.1.4 AES-CCM authenticated Encryption / Decryption

For IPsec (RFC 4309) and also for use in ZFS encryption. Generic CCM mode is described in .

To set up for AES-CCM use the following process, where *K* (key), nonce and additional authenticated data are as described in .

Encrypt:

- Set the message/data length *ulDataLen* in the parameter block.
- Set the nonce length *ulNonceLen* and the nonce data *pNonce* in the parameter block. *pNonce* may be NULL *if ulNonceLen* is 0.
- Set the AAD data *pAAD* and size *ulAADLen* in the parameter block. *pAAD m*ay be NULL if *ulAADLen* is 0.
- Set the MAC length *ulMACLen* in the parameter block.
- Call C_EncryptInit() for **CKM_AES_CCM** mechanism with parameters and key *K*.
- Call C_Encrypt(), or C_DecryptUpdate()*1 C_EncryptFinal(), for the plaintext obtaining ciphertext output obtaining the final ciphertext output and the MAC. The total length of data processed must be *ulDataLen. The output length will be ulDataLen + ulMACLen.*

Decrypt:

- Set the message/data length *ulDataLen* in the parameter block. This length should not include the length of the MAC that is appended to the cipher text.
- Set the nonce length *ulNonceLen* and the nonce data *pNonce* in the parameter block. *pNonce* may be NULL *if ulNonceLen* is 0.
- Set the AAD data *pAAD* and size *ulAADLen* in the parameter block. *pAAD m*ay be NULL if *ulAADLen* is 0.
- Set the MAC length *ulMACLen* in the parameter block.
- Call C_DecryptInit() for **CKM_AES_CCM** mechanism with parameters and key *K*.

- Call C_Decrypt(), or C_DecryptUpdate()*1 C_DecryptFinal(), for the ciphertext, including the appended MAC, obtaining plaintext output. The total length of data processed must be *ulDataLen* + *ulMACLen.*

The key type for *K* must be compatible with **CKM_AES_ECB** and the C_EncryptInit/C_DecryptInit calls shall behave, with respect to K, as if they were called directly with **CKM_AES_ECB**, *K* and NULL parameters.

A. Manifest constants

The following definitions can be found in the appropriate header file.

```
#define CKM_AES_GCM                    0x00001087
#define CKM_AES_CCM                    0x00001088
```

B. Intellectual property considerations

RSA Security Inc. makes no patent claims on the general constructions described in this document, although specific underlying techniques may be covered.

C. References

[1] RSA Laboratories. *PKCS #11: Cryptographic Token Interface Standard*. Version 2.20, June 2004. URL: ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-20/pkcs-11v2-20.pdf.

[2] FIPS Publication 197, "Specification for the Advanced Encryption Standard," U.S. DoC/NIST, November 2001. URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[3] McGrew, D. and J. Viega, "The Galois/Counter Mode of Operation (GCM)," J Submission to NIST, January 2004. URL: http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec.pdf.

[4] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", IETF RFC 3610, September 2003. URL: http://www.ietf.org/rfc/rfc3610.txt

[5] Housley, R., "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)," IETF RFC 4309, December 2005. URL: http://ietf.org/rfc/rfc4309.txt

D. About PKCS

The *Public Key Cryptography Standards* are documents produced by RSA, The Security Division of EMC, in cooperation with secure systems developers for the purpose of simplifying integration and management of accelerating the deployment of public-key cryptography and strong authentication technology into secure applications, and to enhance the user experience of these technologies.

RSA plans further development of the PKCS series through mailing list discussions and occasional workshops, and suggestions for improvement are welcome. Results may also be submitted to standards forums. For more information, contact:

PKCS Editor
RSA, The Security Division of EMC
174 Middlesex Turnpike
Bedford, MA 01730 USA
pkcs-editor@rsasecurity.com
http://www.rsasecurity.com/rsalabs/

