Network Working Group
Request for Comments: 5008
Category: Informational

R. Housley
Vigil Security
J. Solinas
NSA

September 2007

Suite B in Secure/Multipurpose Internet Mail Extensions (S/MIME)

Status of This Memo

Abstract

This document specifies the conventions for using the United States National Security Agency's Suite B algorithms in Secure/Multipurpose Internet Mail Extensions (S/MIME) as specified in RFC 3851.

1. Introduction

This document specifies the conventions for using the United States National Security Agency's Suite B algorithms [SuiteB] in Secure/Multipurpose Internet Mail Extensions (S/MIME) [MSG]. S/MIME makes use of the Cryptographic Message Syntax (CMS) [CMS]. In particular, the signed-data and the enveloped-data content types are used.

Since many of the Suite B algorithms enjoy uses in other environments as well, the majority of the conventions needed for the Suite B algorithms are already specified in other documents. This document references the source of these conventions, and the relevant details are repeated to aid developers that choose to support Suite B. In a few cases, additional algorithm identifiers are needed, and they are provided in this document.

1.1. 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 RFC 2119 [STDWORDS].

1.2. ASN.1

CMS values are generated using ASN.1 [X.208-88], the Basic Encoding Rules (BER) [X.209-88], and the Distinguished Encoding Rules (DER) [X.509-88].

1.3. Suite B Security Levels

Suite B offers two security levels: Level 1 and Level 2. Security Level 2 offers greater cryptographic strength by using longer keys.

For S/MIME signed messages, Suite B follows the direction set by RFC 3278 [CMSECC], but some additional algorithm identifiers are assigned. Suite B uses these algorithms:

	Security Level 1	Security Level 2
Message Digest:	SHA-256	SHA-384
Signature:	ECDSA with P-256	ECDSA with P-384

For S/MIME-encrypted messages, Suite B follows the direction set by RFC 3278 [CMSECC] and follows the conventions set by RFC 3565 [CMSAES]. Again, additional algorithm identifiers are assigned. Suite B uses these algorithms:

	Security Level 1	Security Level 2
Key Agreement:	ECDH with P-256	ECDH with P-384
Key Derivation:	SHA-256	SHA-384
Key Wrap:	AES-128 Key Wrap	AES-256 Key Wrap
Content Encryption:	AES-128 CBC	AES-256 CBC

2. SHA-256 and SHA-256 Message Digest Algorithms

This section specifies the conventions employed by implementations that support SHA-256 or SHA-384 [SHA2]. In Suite B, Security Level 1, the SHA-256 message digest algorithm MUST be used. In Suite B, Security Level 2, SHA-384 MUST be used.

Within the CMS signed-data content type, message digest algorithm identifiers are located in the SignedData digestAlgorithms field and the SignerInfo digestAlgorithm field. Also, message digest values are located in the Message Digest authenticated attribute. In addition, message digest values are input into signature algorithms.

The SHA-256 and SHA-384 message digest algorithms are defined in FIPS Pub 180-2 [SHA2, EH]. The algorithm identifier for SHA-256 is:

```
id-sha256  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistalgorithm(4) hashalgs(2) 1 }
```

The algorithm identifier for SHA-384 is:

```
id-sha384  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistalgorithm(4) hashalgs(2) 2 }
```

There are two possible encodings for the AlgorithmIdentifier parameters field. The two alternatives arise from the fact that when the 1988 syntax for AlgorithmIdentifier was translated into the 1997 syntax, the OPTIONAL associated with the AlgorithmIdentifier parameters got lost. Later, the OPTIONAL was recovered via a defect report, but by then many people thought that algorithm parameters were mandatory. Because of this history some implementations encode parameters as a NULL element and others omit them entirely. The correct encoding for the SHA-256 and SHA-384 message digest algorithms is to omit the parameters field; however, to ensure compatibility, implementations ought to also handle a SHA-256 and SHA-384 AlgorithmIdentifier parameters field, which contains a NULL.

For both SHA-256 and SHA-384, the AlgorithmIdentifier parameters field is OPTIONAL, and if present, the parameters field MUST contain a NULL. Implementations MUST accept SHA-256 and SHA-384 AlgorithmIdentifiers with absent parameters. Implementations MUST accept SHA-256 and SHA-384 AlgorithmIdentifiers with NULL parameters. Implementations SHOULD generate SHA-256 and SHA-384 AlgorithmIdentifiers with absent parameters.

3. ECDSA Signature Algorithm

This section specifies the conventions employed by implementations that support Elliptic Curve Digital Signature Algorithm (ECDSA). The direction set by RFC 3278 [CMSECC] is followed, but additional message digest algorithms and additional elliptic curves are employed. In Suite B, Security Level 1, ECDSA MUST be used with the SHA-256 message digest algorithm and the P-256 elliptic curve. In Suite B, Security Level 2, ECDSA MUST be used with the SHA-384 message digest algorithm and the P-384 elliptic curve. The P-256 and P-384 elliptic curves are specified in [DSS].

Within the CMS signed-data content type, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of SignedData. In addition, signature algorithm identifiers are located in the SignerInfo signatureAlgorithm field of countersignature attributes.

Within the CMS signed-data content type, signature values are located in the SignerInfo signature field of SignedData. In addition, signature values are located in the SignerInfo signature field of countersignature attributes.

As specified in RFC 3279 [PKIX1ALG], ECDSA and Elliptic Curve Diffie-Hellman (ECDH) use the same algorithm identifier for subject public keys in certificates, and it is repeated here:

```
id-ecPublicKey  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840) ansi-x9-62(10045) keyType(2) 1 }
```

This object identifier is used in public key certificates for both ECDSA signature keys and ECDH encryption keys. The intended application for the key may be indicated in the key usage field (see RFC 3280 [PKIX1]). The use of separate keys for signature and encryption purposes is RECOMMENDED; however, the use of a single key for both signature and encryption purposes is not forbidden.

As specified in RFC 3279 [PKIX1ALG], ECDSA and ECDH use the same encoding for subject public keys in certificates, and it is repeated here:

```
ECPoint  ::=  OCTET STRING
```

The elliptic curve public key (an OCTET STRING) is mapped to a subject public key (a BIT STRING) as follows: the most significant bit of the OCTET STRING becomes the most significant bit of the BIT STRING, and the least significant bit of the OCTET STRING becomes the least significant bit of the BIT STRING. Note that this octet string may represent an elliptic curve point in compressed or uncompressed form. Implementations that support elliptic curves according to this specification MUST support the uncompressed form and MAY support the compressed form.

ECDSA and ECDH require use of certain parameters with the public key. The parameters may be inherited from the certificate issuer, implicitly included through reference to a named curve, or explicitly included in the certificate. As specified in RFC 3279 [PKIX1ALG], the parameter structure is:

```
EcpkParameters  ::=  CHOICE {
  ecParameters  ECParameters,
  namedCurve    OBJECT IDENTIFIER,
  implicitlyCA  NULL }
```

In Suite B, the namedCurve CHOICE MUST be used. The object identifier for P-256 is:

```
ansip256r1  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840) ansi-x9-62(10045) curves(3) prime(1) 7 }
```

The object identifier for P-384 is:

```
secp384r1  OBJECT IDENTIFIER  ::=  { iso(1)
    identified-organization(3) certicom(132) curve(0) 34 }
```

The algorithm identifier used in CMS for ECDSA with SHA-256 signature values is:

```
ecdsa-with-SHA256  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-sha2(3) 2 }
```

The algorithm identifier used in CMS for ECDSA with SHA-384 signature values is:

```
ecdsa-with-SHA384  OBJECT IDENTIFIER  ::=  { iso(1) member-body(2)
    us(840) ansi-X9-62(10045) signatures(4) ecdsa-with-sha2(3) 3 }
```

When either the ecdsa-with-SHA256 or the ecdsa-with-SHA384 algorithm identifier is used, the AlgorithmIdentifier parameters field MUST be absent.

When signing, the ECDSA algorithm generates two values, commonly called r and s. To transfer these two values as one signature, they MUST be encoded using the Ecdsa-Sig-Value type specified in RFC 3279 [PKIX1ALG]:

```
Ecdsa-Sig-Value  ::=  SEQUENCE {
  r  INTEGER,
  s  INTEGER }
```

4. Key Management

CMS accommodates the following general key management techniques: key agreement, key transport, previously distributed symmetric key-encryption keys, and passwords. In Suite B, ephemeral-static key agreement MUST be used as described in Section 4.1.

When a key agreement algorithm is used, a key-encryption algorithm is also needed. In Suite B, the Advanced Encryption Standard (AES) Key Wrap, as specified in RFC 3394 [AESWRAP, SH], MUST be used as the key-encryption algorithm. AES Key Wrap is discussed further in Section 4.2. The key-encryption key used with the AES Key Wrap

algorithm is obtained from a key derivation function (KDF). In Suite B, there are two KDFs, one based on SHA-256 and one based on SHA-384. These KDFs are discussed further in Section 4.3.

4.1. ECDH Key Agreement Algorithm

This section specifies the conventions employed by implementations that support ECDH. The direction set by RFC 3278 [CMSECC] is followed, but additional key derivation functions and key wrap algorithms are employed. S/MIME is used in store-and-forward communications, which means that ephemeral-static ECDH is always employed. This means that the message originator uses an ephemeral ECDH key and that the message recipient uses a static ECDH key, which is obtained from an X.509 certificate. In Suite B, Security Level 1, ephemeral-static ECDH MUST be used with the SHA-256 KDF, AES-128 Key Wrap, and the P-256 elliptic curve. In Suite B, Security Level 2, ephemeral-static ECDH MUST be used with the SHA-384 KDF, AES-256 Key Wrap, and the P-384 elliptic curve.

Within the CMS enveloped-data content type, key agreement algorithm identifiers are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm field.

As specified in RFC 3279 [PKIX1ALG], ECDSA and ECDH use the same conventions for carrying a subject public key in a certificate. These conventions are discussed in Section 3.

Ephemeral-static ECDH key agreement is defined in [SEC1] and [IEEE1363]. When using ephemeral-static ECDH, the EnvelopedData RecipientInfos keyAgreeRecipientInfo field is used as follows:

version MUST be 3.

originator MUST be the originatorKey alternative. The originatorKey algorithm field MUST contain the id-ecPublicKey object identifier (see Section 3) with NULL parameters. The originatorKey publicKey field MUST contain the message originator's ephemeral public key, which is a DER-encoded ECPoint (see Section 3). The ECPoint SHOULD be represented in uncompressed form.

ukm can be present or absent. However, message originators SHOULD include the ukm. As specified in RFC 3852 [CMS], implementations MUST support ukm message recipient processing, so interoperability is not a concern if the ukm is present or absent. When present, the ukm is used to ensure that a different key-encryption key is generated, even when the ephemeral private key is improperly used

more than once. See [RANDOM] for guidance on generation of random values.

keyEncryptionAlgorithm MUST be one of the two algorithm identifiers listed below, and the algorithm identifier parameter field MUST be present and identify the key wrap algorithm. The key wrap algorithm denotes the symmetric encryption algorithm used to encrypt the content-encryption key with the pairwise key-encryption key generated using the ephemeral-static ECDH key agreement algorithm (see Section 4.3). In Suite B, Security Level 1, the keyEncryptionAlgorithm MUST be dhSinglePass-stdDH-sha256kdf-scheme, and the keyEncryptionAlgorithm parameter MUST be a KeyWrapAlgorithm containing id-aes128-wrap (see Section 4.2). In Suite B, Security Level 2, the keyEncryptionAlgorithm MUST be dhSinglePass-stdDH-sha384kdf-scheme, and the keyEncryptionAlgorithm parameter MUST be a KeyWrapAlgorithm containing id-aes256-wrap (see Section 4.2). The algorithm identifier for dhSinglePass-stdDH-sha256kdf-scheme and dhSinglePass-stdDH-sha384kdf-scheme are:

```
dhSinglePass-stdDH-sha256kdf-scheme  OBJECT IDENTIFIER  ::=
    { iso(1) identified-organization(3) certicom(132)
      schemes(1) 11 1 }

dhSinglePass-stdDH-sha384kdf-scheme  OBJECT IDENTIFIER  ::=
    { iso(1) identified-organization(3) certicom(132)
      schemes(1) 11 2 }
```

Both of these algorithm identifiers use KeyWrapAlgorithm as the type for their parameter:

```
KeyWrapAlgorithm  ::=  AlgorithmIdentifier
```

recipientEncryptedKeys contains an identifier and an encrypted key for each recipient. The RecipientEncryptedKey KeyAgreeRecipientIdentifier MUST contain either the issuerAndSerialNumber identifying the recipient's certificate or the RecipientKeyIdentifier containing the subject key identifier from the recipient's certificate. In both cases, the recipient's certificate contains the recipient's static ECDH public key. RecipientEncryptedKey EncryptedKey MUST contain the content-encryption key encrypted with the ephemeral-static, ECDH-generated pairwise key-encryption key using the algorithm specified by the KeyWrapAlgorithm (see Section 4.3).

4.2. AES Key Wrap

CMS offers support for symmetric key-encryption key management; however, it is not used in Suite B. Rather, the AES Key Wrap key-encryption algorithm, as specified in RFC 3394 [AESWRAP, SH], is used to encrypt the content-encryption key with a pairwise key-encryption key that is generated using ephemeral-static ECDH. In Suite B, Security Level 1, AES-128 Key Wrap MUST be used as the key-encryption algorithm. In Suite B, Security Level 2, AES-256 Key Wrap MUST be used as the key-encryption algorithm.

Within the CMS enveloped-data content type, wrapped content-encryption keys are located in the EnvelopedData RecipientInfos KeyAgreeRecipientInfo RecipientEncryptedKeys encryptedKey field, and key wrap algorithm identifiers are located in the KeyWrapAlgorithm parameters within the EnvelopedData RecipientInfos KeyAgreeRecipientInfo keyEncryptionAlgorithm field.

The algorithm identifiers for AES Key Wrap are specified in RFC 3394 [SH], and the ones needed for Suite B are repeated here:

```
id-aes128-wrap  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) aes(1) 5 }

id-aes256-wrap  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) aes(1) 45 }
```

4.3. Key Derivation Functions

CMS offers support for deriving symmetric key-encryption keys from passwords; however, password-based key management is not used in Suite B. Rather, KDFs based on SHA-256 and SHA-384 are used to derive a pairwise key-encryption key from the shared secret produced by ephemeral-static ECDH. In Suite B, Security Level 1, the KDF based on SHA-256 MUST be used. In Suite B, Security Level 2, KDF based on SHA-384 MUST be used.

As specified in Section 8.2 of RFC 3278 [CMSECC], using ECDH with the CMS enveloped-data content type, the derivation of key-encryption keys makes use of the ECC-CMS-SharedInfo type, which is repeated here:

```
ECC-CMS-SharedInfo  ::=  SEQUENCE {
  keyInfo      AlgorithmIdentifier,
  entityUInfo  [0] EXPLICIT OCTET STRING OPTIONAL,
  suppPubInfo  [2] EXPLICIT OCTET STRING }
```

In Suite B, the fields of ECC-CMS-SharedInfo are used as follows:

keyInfo contains the object identifier of the key-encryption algorithm that will be used to wrap the content-encryption key and NULL parameters. In Suite B, Security Level 1, AES-128 Key Wrap MUST be used, resulting in {id-aes128-wrap, NULL}. In Suite B, Security Level 2, AES-256 Key Wrap MUST be used, resulting in {id-aes256-wrap, NULL}.

entityUInfo optionally contains a random value provided by the message originator. If the ukm is present, then the entityUInfo MUST be present, and it MUST contain the ukm value. If the ukm is not present, then the entityUInfo MUST be absent.

suppPubInfo contains the length of the generated key-encryption key, in bits, represented as a 32-bit unsigned number, as described in RFC 2631 [CMSDH]. In Suite B, Security Level 1, a 128-bit AES key MUST be used, resulting in 0x00000080. In Suite B, Security Level 2, a 256-bit AES key MUST be used, resulting in 0x00000100.

ECC-CMS-SharedInfo is DER-encoded and used as input to the key derivation function, as specified in Section 3.6.1 of [SEC1]. Note that ECC-CMS-SharedInfo differs from the OtherInfo specified in [CMSDH]. Here, a counter value is not included in the keyInfo field because the KDF specified in [SEC1] ensures that sufficient keying data is provided.

The KDF specified in [SEC1] provides an algorithm for generating an essentially arbitrary amount of keying material from the shared secret produced by ephemeral-static ECDH, which is called Z for the remainder of this discussion. The KDF can be summarized as:

```
KM = Hash ( Z || Counter || ECC-CMS-SharedInfo )
```

To generate a key-encryption key, one or more KM blocks are generated, incrementing Counter appropriately, until enough material has been generated. The KM blocks are concatenated left to right:

```
KEK = KM ( counter=1 ) || KM ( counter=2 ) ...
```

The elements of the KDF are used as follows:

Hash is the one-way hash function, and it is either SHA-256 or SHA-384 [SHA2]. In Suite B, Security Level 1, the SHA-256 MUST be used. In Suite B, Security Level 2, SHA-384 MUST be used.

Z is the shared secret value generated by ephemeral-static ECDH. Leading zero bits MUST be preserved. In Suite B, Security Level 1, Z MUST be exactly 256 bits. In Suite B, Security Level 2, Z MUST be exactly 384 bits.

Counter is a 32-bit unsigned number, represented in network byte order. Its initial value MUST be 0x00000001 for any key derivation operation. In Suite B, Security Level 1 and Security Level 2, exactly one iteration is needed; the Counter is not incremented.

ECC-CMS-SharedInfo is composed as described above. It MUST be DER encoded.

To generate a key-encryption key, one KM block is generated, with a Counter value of 0x00000001:

KEK = KM (1) = Hash (Z || Counter=1 || ECC-CMS-SharedInfo)

In Suite B, Security Level 1, the key-encryption key MUST be the most significant 128 bits of the SHA-256 output value. In Suite B, Security Level 2, the key-encryption key MUST be the most significant 256 bits of the SHA-384 output value.

Note that the only source of secret entropy in this computation is Z. The effective key space of the key-encryption key is limited by the size of Z, in addition to any security level considerations imposed by the elliptic curve that is used. However, if entityUInfo is different for each message, a different key-encryption key will be generated for each message.

5. AES CBC Content Encryption

This section specifies the conventions employed by implementations that support content encryption using AES [AES] in Cipher Block Chaining (CBC) mode [MODES]. The conventions in RFC 3565 [CMSAES] are followed. In Suite B, Security Level 1, the AES-128 in CBC mode MUST be used for content encryption. In Suite B, Security Level 2, AES-256 in CBC mode MUST be used.

Within the CMS enveloped-data content type, content encryption algorithm identifiers are located in the EnvelopedData EncryptedContentInfo contentEncryptionAlgorithm field. The content encryption algorithm is used to encipher the content located in the EnvelopedData EncryptedContentInfo encryptedContent field.

The AES CBC content-encryption algorithm is described in [AES] and [MODES]. The algorithm identifier for AES-128 in CBC mode is:

```
id-aes128-CBC  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) aes(1) 2 }
```

The algorithm identifier for AES-256 in CBC mode is:

```
id-aes256-CBC  OBJECT IDENTIFIER  ::=  { joint-iso-itu-t(2)
    country(16) us(840) organization(1) gov(101) csor(3)
    nistAlgorithm(4) aes(1) 42 }
```

The AlgorithmIdentifier parameters field MUST be present, and the parameters field must contain AES-IV:

```
AES-IV  ::=  OCTET STRING (SIZE(16))
```

The 16-octet initialization vector is generated at random by the originator. See [RANDOM] for guidance on generation of random values.

6. Security Considerations

This document specifies the conventions for using the NSA's Suite B algorithms in S/MIME. All of the algorithms have been specified in previous documents, although a few new algorithm identifiers have been assigned.

Two levels of security may be achieved using this specification. Users must consider their risk environment to determine which level is appropriate for their own use.

For signed and encrypted messages, Suite B provides a consistent level of security for confidentiality and integrity of the message content.

The security considerations in RFC 3852 [CMS] discuss the CMS as a method for digitally signing data and encrypting data.

The security considerations in RFC 3370 [CMSALG] discuss cryptographic algorithm implementation concerns in the context of the CMS.

The security considerations in RFC 3278 [CMSECC] discuss the use of elliptic curve cryptography (ECC) in the CMS.

The security considerations in RFC 3565 [CMSAES] discuss the use of AES in the CMS.

7. References

7.1. Normative References

[AES] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", FIPS PUB 197, November 2001.

[AESWRAP] National Institute of Standards and Technology, "AES Key Wrap Specification", 17 November 2001. [See http://csrc.nist.gov/encryption/kms/key-wrap.pdf]

[DSS] National Institute of Standards and Technology, "Digital Signature Standard (DSS)", FIPS PUB 186-2, January 2000.

[ECDSA] American National Standards Institute, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62-1998, 1999.

[CMS] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[CMSAES] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)", RFC 3565, July 2003.

[CMSALG] Housley, R., "Cryptographic Message Syntax (CMS) Algorithms", RFC 3370, August 2002.

[CMSDH] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[CMSECC] Blake-Wilson, S., Brown, D., and P. Lambert, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 3278, April 2002.

[IEEE1363] Institute of Electrical and Electronics Engineers, "Standard Specifications for Public Key Cryptography", IEEE Std 1363, 2000.

[MODES] National Institute of Standards and Technology, "DES Modes of Operation", FIPS Pub 81, 2 December 1980.

[MSG] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[PKIX1] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[PKIX1ALG] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[SEC1] Standards for Efficient Cryptography Group, "Elliptic Curve Cryptography", 2000. [See http://www.secg.org/collateral/sec1.pdf.]

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

[SHA2] National Institute of Standards and Technology, "Secure Hash Standard", FIPS 180-2, 1 August 2002.

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

[X.208-88] CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.

[X.209-88] CCITT. Recommendation X.209: Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1). 1988.

[X.509-88] CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

7.2. Informative References

[EH] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and HMAC-SHA)", RFC 4634, July 2006.

[RANDOM] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[SuiteB] National Security Agency, "Fact Sheet NSA Suite B Cryptography", July 2005. [See http://www.nsa.gov/ia/industry/crypto_Suite_b.cfm?MenuID=10.2.7)

Authors' Addresses

Russell Housley
Vigil Security, LLC
918 Spring Knoll Drive
Herndon, VA 20170
USA

EMail: housley@vigilsec.com

Jerome A. Solinas
National Information Assurance Laboratory
National Security Agency
9800 Savage Road
Fort George G. Meade, MD 20755
USA

EMail: jasolin@orion.ncsc.mil

Full Copyright Statement

Intellectual Property