NASJONAL
SIKKERHETSMYNDIGHET

NSM Cryptographic Recommendations

Versjon: 1.0

CONTENT

1. Introduction

The Norwegian National Security Authority (NSM) is the governmental organization authorized to approve cryptographic systems and products for the protection of information classified according to the Security Act.

This document supersedes *NSM Cryptographic Requirements* and will be reviewed on a regular basis.

For ease of reference, there is a table summarizing important information from the main text at the end of the document.

2. Scope

This document provides recommendations and guidance to cryptographic mechanisms used to protect Norwegian classified and unclassified information as well as information systems. It may be regarded as a high level guide to any user of cryptographic mechanisms. It is not intended for product implementation or evaluation purposes.

NSM expects owners of information systems handling classified information to comply with these recommendations. Any deviations should be justified and explained to the information system security accreditation authority.

3. General remarks

In reality, the security of cryptographic primitives and mechanisms is far more complex than simply following the recommendations of this document on their own. There are many things that can go wrong in the process of their implementation, so any party intending to do so should seek professional advice before using them in their own system.

In general, it is hard to predict how long a cryptographic mechanism will remain secure. It is therefore essential that system architects or owners of cryptographic systems design their systems so that cryptographic primitives may easily be modified or replaced at a later point in time.

4. Key management

Key management is essential for the proper use of cryptography. We distinguish the following objectives as fundamental:

- Protecting the confidentiality and authenticity of secret and private keys, as well as protecting secret and private keys against unauthorized use.
- Protecting the authenticity of public keys.
- Ensuring the availability of secret and public keys.

To accomplish these three goals one needs to examine the entire key life cycle, from the generation of key material through to its secure deletion or destruction.

The *NIST SP 800-57 Recommendation for Key Management, 2016,* provides a framework for describing key management and is a relevant reference for more on the elements described in this document.

4.1. Key generation

Secret and private keys need to be unpredictable. In general, symmetric primitives will not have additional requirements for their secret keys. However, there are special cases where a small fraction of weak keys should be avoided. Asymmetric primitives usually have additional requirements, both on private and public keys.

4.2. Key registration and certification

Some keys need to be associated with their owner. For example, public keys are linked to their owner by means of certificates. Through the issuing of a certificate, a certification authority guarantees that a certain key belongs to a certain user, and associated policy statements specify for what purposes the owner may use the key.

A certificate has a validity period and is usually a public document. Its authenticity is ensured by means of a digital signature, placed by the certification authority. However, one needs to trust the certificate authority and its public key, which is itself authenticated by another certificate authority creating a certificate chain.

At the root of the chain is a root certificate authority. These root certificates can be distributed to relying parties and signatories by including them in applications, or having them downloaded from an authoritative source (e.g. a designated public authority), all for the purpose of invoking trust.

4.3. Key distribution

Keys need to be distributed. For systems based on symmetric cryptography, there is need for a common key at both ends of communication. This key needs to be transported securely (protection of confidentiality and authenticity) at least once, or agreed via means of a key agreement scheme.

For systems based on asymmetric cryptography, the private key is often generated where it is to be used, so that no transport is needed.

Private keys and all copies of symmetric keys need to be installed and stored securely.

4.4. Key use

The goal of key management is to put keys in place such that they can be used for a certain period of time. During the lifetime of a key, it has to be protected against unauthorized use by attackers.

Cryptographic keys expire and are replaced. There are situations where keys have to be discarded ahead of their planned end of lifetime, e.g. if secret keys leak to outsiders or if developments in cryptanalysis make schemes insecure. This process is termed revocation.

The key must also be protected against unauthorized use by the owner of the key: The owner of the key should not be allowed to export the key or use it in an insecure environment.

After the end of their operational lifetimes, or if they have been compromised, keys should be erased in a proper manner.

5. Symmetric primitives

5.1. Block ciphers

Block ciphers are permutation functions which produce an m-bit ciphertext from an m-bit message and a k-bit key. The function should be computationally infeasible to invert without knowing the key.

Recommended block ciphers are:

- *ADVANCED ENCRYPTION STANDARD*

Advanced Encryption Standard (AES) is defined in *FIPS PUB 197 Advanced Encryption Standard (AES), 2001*.

The recommended key length is 256 bits.

5.2. Hash functions

Hash-functions are functions that take a message of arbitrary length as input, and produce a finite length (h-bit) cryptographic message digest (a hash) as output. The hash of a message is sometimes called a fingerprint.

Recommended hash functions are:

- *SHA-2*
- *SHA-3*

Secure Hash Algorithm 2 (SHA-2) is defined in *FIPS PUB 180-4 Secure Hash Standard (SHS), 2012*. SHA-3 is defined in *FIPS PUB 202 SHA-3 Standard, 2015*.

The recommended hash function output length is 384 bits or higher (SHA-384, SHA3-384, SHA-512, SHA3-512).

6. Symmetric schemes

6.1. Block cipher modes of operation

A mode of operation specifies how to extend a block cipher to encrypt messages of sizes larger than a block. It describes how nonces (a number used only once) and keys are combined with cryptographic primitives in order to provide secure encryption.

Note that none of the modes in this section provide integrity protection, a feature only achieved in combination with an approved MAC or a dedicated authenticated mode. In general, NSM recommends the use of authenticated encryption.

Recommended modes of operation are:

- *COUNTER MODE (CTR)*

CTR turns a block cipher into a stream cipher by encrypting an incremental counter starting at a unique value (nonce) defined by an initialization vector (IV). This mode does not require padding and is recommended if no integrity protection is required.

- *CIPHER BLOCK CHAINING (CBC)*

CBC is a widely used mode. It requires a separate implementation of decryption.

- *XEX TWEAKABLE BLOCK CIPHER WITH CIPHERTEXT STEALING (XTS-AES)*

XTS-AES is designed for encryption of data at rest. This mode does not provide full integrity protection, but does provide countermeasures against adversaries seeking to manipulate ciphertext.

6.2. Message authentication codes

Message authentication codes provide authentication of messages. Hash functions provide cryptographically secure integrity protection and a MAC provides authenticity in addition to this.

Recommended MACs are:

- *CIPHER-BASED MAC (CMAC)*

Cipher-based MAC (CMAC) is specified in *NIST SP 800-38B Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication,* and defines a block cipher-based message authentication code algorithm. CMAC is used to ensure the authenticity and integrity of a message. The recommended algorithms can be found in part 5.1.

- *HASH-BASED MAC (HMAC)*

HMAC is specified in *FIPS PUB 198-1 The Keyed-Hash Message Authentication Code (HMAC)* and provides authenticity and integrity protection for messages. HMAC should be used with a recommended hash function. The recommended algorithms can be found in part 5.2.

6.3. Authenticated encryption

Authenticated encryption modes provide both integrity and authenticity. These modes are typically either one-pass or two-pass. Two-pass modes are modes that separate the computation of encryption and authentication tags. Hence, one-pass modes are usually much faster.

Recommended authenticated encryption modes are:

- *AES GALOIS COUNTER MODE (AES-GCM)*

GCM is a one-pass authenticated counter mode. GCM employs universal hashing to protect against adversaries that seek to manipulate data. It is strongly advised against using short authentication tags in GCM.

- *COUNTER WITH CBC-MAC (CCM)*

CCM is a two-pass authenticated mode that combines AES-CTR with CBC-MAC. Being two-pass, this mode is slower than GCM.

6.4. Key protection

Key Wrap (KW) is intended to protect the confidentiality and integrity of cryptographic keys. It is characterized by taking a secret key (K) and a key-encryption key (KEK) as input to produce a wrapped key W.

Recommended key wrap functions are:

- *AES KEY WRAP (KW)*

AES Key Wrap (KW) is defined in *NIST SP 800-38 F Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping* specifies wrapping and ensures the confidentiality and integrity of cryptographic keys.

- *AES KEY WRAP WITH PADDING (KWP)*

AES Key Wrap with Padding (KWP) is defined in *NIST SP 800-38 F Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping* extends KW by specifying a padding scheme (adding information data to conform with the block size of AES).

6.5. Key derivation functions

Key Derivation Functions (KDFs) are used to derive keys from random strings (seeds). A KDF takes some random secret strings, possibly together with additional non-secret data, as input to allow for larger variations.

Recommended key derivation functions are:

- *EXTRACT-THEN-EXPAND*

Key derivation is defined in *NIST SP 800-56C Recommendation for Key Derivation through Extraction-then-Expansion*.

7. Asymmetric primitives

In asymmetric cryptography, security is based on a discrepancy of two mathematical problems in terms of their computational difficulty: One is easy while the other is hard. This discrepancy is needed in order to ensure the possibility of generating pairs of private and public keys for which it is computationally hard to recover the private key from the public key.

7.1. RSA / integer factorization

The security of various asymmetric cryptographic schemes, including the well-known RSA schemes, relies on the difficulty of integer factorization compared to that of integer multiplication and modular exponentiation.

The RSA primitive consists of a public permutation parametrized by a public key and the private inverse permutation parametrized by the associated private key. Note that the primitive alone cannot be considered to be a complete encryption or signature scheme since using it without extra conventions would be highly insecure.

Let p and q be prime numbers and $n = p*q$ their product, called the modulus. The public key is formed by the modulus n together with an element e, called the public exponent, which is invertible modulo (p-1)*(q-1).An inverse of e modulo $(p-1)*(q-1)$, denoted by d, is called the private exponent.

The private key is formed by this private exponent together with the modulus. The public permutation operates on integers modulo n and consists in the exponentiation of the input to the power e. The private permutation operates on integers modulo n and consists in the exponentiation of the input to the power d.

It is recommended that $\log_2(n)\geq 3072$ and that $\log_2(e)>16$.

7.2. Discrete logarithms in finite fields

The security of several asymmetric cryptographic schemes relies on the difficulty of the discrete logarithm problem versus that of exponentiation in finite fields.

In principle, there is a variety of choices for the finite field, but the only secure and widely used solution is to pick a prime field GF(p) where p is a prime number. Primitives that rely on the discrete logarithm problem in GF(p) can be used in various key exchange, signature, or (hybrid) encryption algorithms.

Let g be a generator for a subgroup of order q of the multiplicative group GF$(p)^*$, and let r be the largest prime factor of q. The primitive is based on the exponentiation function of base g in GF(p) that on the input of an integer x (typically between 1 and $q - 1$) returns $X = g^x$.

Depending on how the scheme uses the primitive, x and X may represent (a part of) a private key and the associated public key, or they may represent an ephemeral Diffie-Hellman exponent and the associated public value, etc.

It is recommended that $\log_2(p)\geq 3072$ and that $\log_2(r)\geq 250$.

7.3. Discrete logarithms over elliptic curves

The difficulty of the discrete logarithm problem can also be considered in the group of rational points of an elliptic curve over a finite field. In these groups, the discrete logarithm problem is also thought to be difficult compared to scalar point multiplication.

There are a couple of choices to be made in the case of elliptic curves: Firstly, the finite field over which the elliptic curve is defined, and secondly the elliptic curve itself.

Let E be an elliptic curve defined over GF(p), P a point in E(GF(p)) of order q and r be the largest prime factor of q.

The primitive based on the discrete logarithm problem on E relies on scalar multiplication: On the input of an integer x between 1 and $q-1$, it returns the point $Q = [x]P$. The public parameters in cryptographic schemes where the primitive is used are formed by p, E, P, and q.

Depending on the cryptographic scheme where the primitive is invoked, x and Q may represent (a part of) a private key and the associated public key, or they may represent an ephemeral Diffie-Hellman private value and the associated public value, etc.

Recommended elliptic curves are:

- *BRAINPOOL CURVES*

Brainpool Curves are defined in *ECC Brainpool Standard Curves and Curve Generation, 2010*. It is recommended that one uses the curves BrainpoolP384r1 or BrainpoolP512r1.

- *NIST CURVES*

NIST curves are defined in *FIPS PUB 186-4, Appendix D.1.2, 2013*. It is recommended that one uses the curves NIST P-384 or NIST P-521.

7.4. Quantum computation and cryptography

The advent of quantum computers would retroactively threaten the privacy of data. Thus measures have to be taken to protect data for which there is a need for long-term privacy. This may involve adopting a higher security margin for symmetric cryptographic mechanisms and using hybrid solutions for asymmetric cryptographic mechanisms: Combining a traditional cryptographic mechanism with a quantum resistant cryptographic mechanism will hopefully provide a robustness against classical attacks as well as some resistance against attacks using a quantum computer.

While there is no knowledge of a quantum computer large enough to threaten the security of state-of-the-art cryptographic mechanisms, quantum computing is a very active area of research. Some experts expect a relevant quantum computer to become a reality in the coming decades.

This document does not provide agreed asymmetric quantum resistant mechanisms. Such mechanisms will be introduced in future versions, following the ongoing process of assessing their security.

The mathematical problems mentioned in the previous sections are widely used today. However, there are other mathematical problems leading to trapdoor functions or proof of knowledge constructions for which no generic efficient solving method is known in general.

To cite a few:

- Lattice-based constructions such as the ones built on the Learning With Errors (LWE) problem.
- Code-based cryptography relying on the difficulty of decoding a random error correcting code.
- Multivariate cryptography.
- Hash-based cryptography.
- Isogeny-based cryptography.

These constructions have received less attention than the integer factorization problem and the discrete logarithm problems in finite fields or elliptic curves. As a consequence, they should only be used after careful consideration.

For the time being, no asymmetric primitive beyond the three mathematical problems handled in the previous sections has reached the needed maturity for sound recommendations. However, it is recommended that one considers opting for the utilization of such primitives for future use, thereby achieving “cryptographic agility”.

There is ongoing work to standardize quantum resistant cryptography, candidate algorithms will probably receive more attention in the coming years. Any user of cryptography should follow this process closely.

8. Asymmetric schemes

The primitives of part 7 can be used to build asymmetric schemes. In them, typically, each user is attributed a key pair, consisting of a public key *pk* that can be published, and an associated private key *sk* that should remain confidential.

The security of asymmetric schemes relies on the security of a mathematical problem, and can also rely on the security of a symmetric primitive or scheme. The security of keyed asymmetric schemes relies on the confidentiality and integrity of the private key and the integrity and data origin authentication of the public key.

8.1. Digital signatures

Digital signatures can provide integrity protection and authentication.

Recommended asymmetric signature algorithms are:

- *DIGITAL SIGNATURE ALGORITHM (DSA)*

DSA is defined in *NIST FIPS PUB 186-4 Digital Signature Standard (DSS), 2013*.

- *ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM (ECDSA)*

ECDSA is defined in *NIST FIPS PUB 186-4 Digital Signature Standard (DSS), 2013*.

- *RSA SIGNATURE SCHEME WITH APPENDIX- PROBABILISTIC SIGNATURE SCHEME (RSASSA-PSS)*

RSASSA-PSS is defined in *RSA Labs PKCS #1 v2.2, 2012*.

8.2. Key exchange

Asymmetric key establishment schemes allow two or more parties to generate a common secret without using any pre-shared secret values. They are usually combined with asymmetric or symmetric authentication based on a public/private key pair or a shared secret key.

The most widely used two-party key establishment scheme was proposed by Diffie and Hellman. It relies on the discrete logarithm problem (instantiated in any suitable group).

It should be noted that in in its basic form, the protocol is vulnerable to man-in-the-middle attacks (among other issues). In particular, additional steps should be performed and additional data should be exchanged to ensure authentication of the users and of the key establishment messages.

Recommended key exchange algorithms are:

- *DIFFIE-HELLMAN KEY EXCHANGE(DH)*

DH is defined in *NIST SP 800-56A Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm*, 2018.

- *ELLIPTIC CURVE DIFFIE-HELLMAN KEY EXCHANGE (ECDH)*

ECDH is defined in *NIST SP 800-56A Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm*, 2018.

It is recommended that one implements forward secrecy (protection of past sessions against future compromised keys) and authenticated key establishment.

9. Random number generation

Randomness is needed in almost all cryptographic systems and protocols. For example, random numbers are needed for generating asymmetric key pairs, for defining symmetric keys, for generating IVs for cryptographic modes of operation, in challenge-response protocols, as additional inputs to most standardized public key encryption and signature algorithms, and to generate ephemeral values in key exchange protocols.

Almost all formal security analysis of cryptographic schemes fails if perfect randomness assumptions are not met. There are many prominent examples of randomness failures with severe security consequences.

A random source is a probabilistic process from which random bits can be extracted. It is typically very difficult to assess the quality of the output of a random source. There are basically two approaches:

- **Perform statistical tests on the output of the source.** This approach is black box: No knowledge about the source is required to conduct the tests. It suffers from two main drawbacks: First of all, the statistical tests are generic and may only be used to detect specific shortcomings of the source relative to an ideal random source. Secondly, it does not provide any assurance on the distribution of the output of the random source. Note that despite these shortcomings, statistical tests are useful measures to detect unintentional failure of the random source.
- **Model the probabilistic process of the source.** This approach requires a deep understanding of the random source design, and tries to assess the quality of the source from the study of a theoretical model of the source. This approach provides better assurances. Some aspects like the degree of correspondence between the random source and its model may be difficult to evaluate.

There is a distinction between True Random Number Generators (TRNGs) and Pseudo-Random Number Generators (PRNGs). TRNGs usually involve the use of special-purpose hardware (e.g. electronic circuits, quantum devices) followed by suitable post-processing of the raw output data to generate random numbers.

In an ideal world, all random number requirements would be met by using TRNGs. But, typically, TRNGs operate at low output rates (relative to PRNGs) and are of moderate-to-high cost (relative to PRNGs which are usually implemented in software).

A TRNG device might be used to generate highly sensitive cryptographic keys, e.g. system master keys, in a secured environment, but would be considered overkill for general-purpose use.

PRNGs are suitable for general purpose computing environments and usually involve a software only method. Here, the approach is to deterministically generate random-looking output from an initial seed value.

For further details and recommendations on random number generation, the reader is advised to consult *NIST SP 800-90A Recommendation for Random Number Generation Using Deterministic Random Bit Generators, 2016.*

10. Quick reference

For ease of reference, we include a table summarizing important information from the main text here.

Algorithm	**Function**	**Parameters**
Advanced Encryption Standard (AES).	Block cipher used for information protection.	k=256.
Secure Hash Algorithm (SHA).	Applied to produce a condensed representation of information.	SHA-384/512, SHA3-384/512.
Digital Signature Algorithm (DSA).	Asymmetric algorithm used for digital signatures.	Minimum 3072-bit modulus.
Elliptic Curve Digital Signature Algorithm (ECDSA).	Asymmetric algorithm used for digital signatures.	BrainpoolP384r1/BrainpoolP512r1, NIST P-384/P-521.
RSA.	Asymmetric algorithm used for digital signatures.	Minimum 3072-bit modulus.
Diffie-Hellman (DH) Key Exchange.	Asymmetric algorithm used for key establishment.	Minimum 3072-bit modulus.
Elliptic Curve Diffie-Hellman (ECDH) Key Exchange.	Asymmetric algorithm used for key establishment.	BrainpoolP384r1/BrainpoolP512r1, NIST P-384/P-521.

Nasjonal sikkerhetsmyndighet

Postboks 814
1306 Sandvika

post@nsm.stat.no
www.nsm.stat.no