This is a draft version of an updated version of the NSM Cryptographic Recommendations.

This draft is circulated with the purpose of providing advance guidance and with a request for feedback. It is not a currently official recommendation, and there can be several changes before this recommendation is finalized. Version 1.0 of the NSM Cryptographic Recommendations are the currently applicable recommendations.

Contents

Part I
Introduction

Document history:

Year	Changes from the previous version
2018	
2024	Fundamental rewrite. Introduced quantum-resistant algorithms. Added more complementary text. Removed a number of options for algorithms.

This document provides recommendations and guidance for cryptographic mechanisms used to protect information and information systems. These recommendations apply to unclassified information as well as classified information up to and including BEGRENSET. Any information system which protects classified information must be evaluated and approved by NSM on a case-by-case basis. While it is the belief of NSM that the recommendations in this document provide a sufficient level of security, at least from a cryptographic point of view, any system should be thoroughly tested and its security claims evaluated by experts. Simply following these recommendations is not sufficient on its own.

This document gives simple recommendations which are intended to cover most use cases. By design, few options are presented, and choosing between them should be based on non-cryptographic factors such as software availability and interoperability concerns.

The purpose of this document is not to be an exhaustive list of reasonable cryptographic mechanisms, and there are scenarios where the appropriate mechanism is not found in this document. Anyone finding the recommendations in some way lacking or otherwise unfit for their use case is welcome to contact NSM with any questions they may have.

It is advised that anyone planning to build a system which involves cryptography contacts someone with cryptographic expertise early in the process, and this document can be used as a starting point for a discussion with such experts.

Target audience

This document addresses those who work with IT security but do not have a background in cryptography. They must know about certain aspects of cryptography but do not need to know the technical details of how or why various cryptographic mechanisms work.

One important purpose is to be a resource during an acquisition process. When acquiring a system which involves cryptography, an organization can point to this document and require that the system follows these recommendations.

This document also contains information for IT security leaders and developers about requirements for cryptography to provide actual security outside of the algorithms themselves. This includes generating random numbers, as well as managing cryptographic keys and cryptographic software.

Interoperability

Interoperability is an important constraint facing many system owners. These recommendations are primarily intended for when someone controls every part of the system. However, some comments on interoperability is in order.

Many organizations have to securely communicate with other entities, and here one must agree on which cryptographic mechanisms to use. For a number of reasons one might be forced to use certain mechanisms.

Another area where interoperability concerns are important comes to support for different mechanisms in different software/hardware platforms. It can be the case that the algorithms implemented on a particular platform for certain use cases are not listed in this document, but integrating other mechanisms is difficult or expensive.

It is crucial to minimize the security impact of such interoperability concerns. An assessment from an expert about the security of the interoperable cryptographic mechanism forms a useful starting point for further decisions. Additionally, interoperability concerns must not be used as an excuse to use insecure cryptography beyond what is strictly necessary.

The contents of this document

Using cryptography to protect information requires algorithms which give a mathematical description of how to protect information, and the implementation and integration of those algorithms in a computer system.

On the algorithm side, there are well-established solutions with a high degree of confidence in their security. However, if those algorithms are implemented or integrated without proper care, that can provide an entry point for an attacker. Implementing cryptography is difficult, and should not be done without a high level of competence. For this reason, users of cryptography should rely on well-tested libraries and not create their own implementations.

This document consists of two main parts. The first part advises about points that are important to consider when handling cryptography. This is rather generic advice about which aspects one needs to consider beyond just the choice of algorithms. The second part contains recommendations for which algorithms to use for specific purposes.

Security levels

Most cryptographic mechanisms have a certain flexibility when it comes to the choice of parameters, where different sets of parameters have different efficiency and security properties.

The metric that is typically used to quantify security is how many operations on a computer are needed for an attacker to break a security property, such as finding the message given an encryption of that message, using the best-known attack. When a cryptographic mechanism is said to have "n bits of security", this means that an attacker needs to execute approximately 2^n operations to break a security property. The security level of the mechanism is not always clear from the size of the parameters used.

To give context for what is achievable, the Bitcoin network, a large distributed collection of highly specialized computers, computes approximately 2^{94} operations per year, so schemes providing 94 bits of security could be broken by an attacker with the same resources in about a year. Each added bit of security doubles the resources needed, so such an attacker would need to work for approximately 17 billions of years to break a scheme providing 128 bits of security.

The recommendations for cryptographic mechanisms in this document come with parameter sets that all aim to provide at least 128 bits of security. NSM believes that all recommended parameter sets in this document provide an adequate security level for the scope of these recommendations. When selecting a parameter set for a particular mechanism, the advice is to select the highest parameter set that does not have a noteworthy impact on the performance of the system. Note that what is acceptable will vary from mechanism to mechanism, as different mechanisms need differently sized parameters to reach a similar security level.

It is worth pointing out that simply selecting a parameter set with a higher security level does not necessarily provide an increase in total security, as algorithmic security is only one part of secure use of cryptography.

Part II
General advice

1 Cryptographic computing

While cryptographic mechanisms are described as mathematical algorithms, they need to be implemented in the real world if they are going to protect actual information. This means that the algorithms run on a physical device (be it a regular computer or some specialized cryptographic hardware), and are often integrated into a complex piece of software. It is exceedingly rare that attackers break the algorithms themselves, and significantly more common that they exploit vulnerabilities in implementations of the algorithms or that they simply bypass the cryptography in some way. There is no recipe to follow to handle these things properly, but what follows are some key aspects to keep in mind.

1.1 Implementations

Implementing cryptography requires a lot of expertise. For this reason, users of cryptography should not create their own implementations. They should instead rely on pre-existing implementations developed and maintained by experts, where there is a lot of confidence in the security of the implementation.

It is recommended to use cryptographic software and hardware which has been evaluated and certified, such as via Common Criteria or according to ISO/IEC 19790 (FIPS 140-3). Cryptographic software should take advantage of any dedicated cryptographic hardware on a device.

1.2 Side-channel attacks

One category of attacks against cryptographic mechanisms are *side-channel attacks*. These attacks exploit information that is leaked when a computer uses a secret key to do a cryptographic operation. Some examples of information which can be exploited are the duration of an operation, the power used during the operation, and the memory accesses made during the computation.

Different attacks require differing levels of access. An attack which exploits the time it takes to perform an operation can possibly be done across the internet, but to exploit the power consumption of a device an attacker must be physically close to it.

There are several countermeasures which defend against different kinds of side-channel attacks, and an implementation must apply the countermeasures against the threats it is reasonable to protect against. There are, for example, relatively few cases where the expensive countermeasures that make it harder to exploit power consumption are necessary, but many cases where it is important to protect against attacks which exploit the duration of an operation.

1.3 Cryptographic inventory

It is recommended to have a detailed overview of the cryptography involved in a system. This overview should include which data is protected by cryptography, and how cryptography protects the data (such as whether it is protecting the confidentiality, integrity or authenticity of some data). It is also important to know which cryptographic software and hardware is involved in the system, where they come from, and which algorithms they use. If the system's cryptographic solutions are purchased from outside vendors, it is also important to keep track of which vendors delivered which solutions, and the details of the agreements with each vendor concerning maintenance and upkeep.

1.4 Cryptographic agility

An important concept to be aware of and try to follow is that of *cryptographic agility*. This means that it should be relatively easy to change the cryptographic elements of a computer system in response to newly discovered attacks, whether on an algorithmic or implementation level. If an algorithm is found to be weak, it should be possible to change to a different algorithm with the same functionality without requiring a complete redesign of the system. The same should be the case if the library one uses stops being maintained or one needs to switch to a different vendor.

Some important aspects of cryptographic agility:

- Systems should, as far as possible, not depend on properties specific to one mechanism, such as the size of various objects.
- Source code should be modular. This means that the source code should simply call functions with names named after their functionality, like `encrypt` and `decrypt`. Now changing the algorithms used just involves changing those functions instead of the entire code base.

It is worth noting that in some cases, being cryptographically agile can be infeasible due to long-lived hardware which is difficult to modify. In these cases, additional care is needed at the design stage to reduce the possibility and impact of new attacks on algorithms or implementations.

2 Key management

Key management is essential for the proper use of cryptography. If cryptographic keys are not securely generated, protected, used, and deleted after use, the cryptographic mechanisms provide no security. There are several kinds of cryptographic keys, each with their requirements. Most keys must be kept secret, this means protecting the confidentiality and authenticity of those keys, as well as protecting them against unauthorized use. Some keys are made public, but protecting their authenticity is vital.

To accomplish these goals one needs to examine the entire key life cycle, from the generation of key material through to its secure deletion or destruction. NIST SP 800-57 Part 1 [20] provides a framework for describing key management and is a relevant reference for further information.

2.1 Key generation

Cryptographic keys are generated at random. In general, a key for a symmetric mechanism is a randomly generated bit string of a specified length. For asymmetric mechanisms, a key pair is randomly generated by an algorithm specific to that mechanism. The recommendations in Section 3 should be followed when generating keys.

2.2 Certificates

Certain keys need to be associated with their owner. For example, some public keys are linked to their owner by *certificates.* Such a certificate can also include useful metadata specifying how and when a public key can be used.

By issuing a certificate, a *certificate authority* (CA) guarantees that a specific key belongs to a specific entity. However, one needs to trust the CA and its public key, which is itself authenticated by another CA. This creates a certificate chain.

At the end of the chain is a *root certificate authority.* These root certificates must be securely 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 to invoke trust.

2.3 Key distribution

Keys need to be distributed. For symmetric cryptography, two parties must possess the same key. This key needs to be transported securely at least once (using a scheme recommended in Section 5.1) or agreed upon using a key establishment scheme (using a scheme recommended in Section 6.1). For systems based on asymmetric cryptography, the private key is typically generated where it is to be used, hence no transport is needed.

Secret keys must be securely installed and stored, separate from the data they protect.

2.4 Key use

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

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. After the end of their operational lifetimes, or if they have been compromised, keys should be erased in a manner that removes all traces of the keys so that they cannot be recovered by either physical or electronic means.

3 Random number generation

Randomness is a core element of many cryptographic mechanisms. Random values are needed for generating cryptographic keys and as additional inputs in many cryptographic algorithms and protocols.

A mechanism which generates a sequence of random bits is referred to as a *random number generator* (RNG). The random values must be *cryptographically secure*, meaning that no one can accurately predict any bit in the sequence, even if they can see any other bits in the sequence.

To generate cryptographically secure randomness one first gathers a number of random bits based on a physical process. This process must have sufficient *entropy*, a measure of the amount of randomness in the process. These bits are then used as a *seed* for a suitable pseudo-random number generator (PRNG), which expands a short seed into a long sequence of cryptographically secure random bits.

Both gathering the random bits from a physical process and estimating the entropy of this process is difficult. It is therefore recommended to use a pre-existing solution. Many devices provide specialized hardware for generating cryptographically secure random numbers, which should be used when available.

If such hardware is not available, great care must be taken that the physical process does indeed generate sufficient entropy. Deciding on this process and the PRNG should be done in consultation with an expert, and NIST SP 800-90A [21] and NIST SP 800-90B [22] provide further information relevant to such a discussion.

4 Quantum computing

The development of quantum computers of sufficient capability, referred to as Cryptanalytically Relevant Quantum Computers (CRQCs), will have a significant impact on the cryptographic mechanisms in use today, particularly for asymmetric cryptography. A quantum computer uses the principles of quantum mechanics to solve certain, but not all, problems faster than what is possible on a classical computer.

The most common asymmetric algorithms in use as of the time of writing are broken by CRQCs. This can not be mitigated by simply increasing the parameters, and there is therefore a need to use new algorithms which are not vulnerable to attacks by quantum computers. For symmetric cryptography, the impact is limited. There are generic quantum attacks which give certain improvements over the best known classical attacks, but these attacks require even more powerful quantum computers, and they will not achieve a significant improvement compared to classical attacks in practice. There is no need to change the algorithms used, but one might in future need to increase the size of the parameters.

The development of a CRQC is an area of active research, and as such it is hard to predict when, or even if, a CRQC will be developed. NSM works under the conservative assumption that one should be prepared for the existence of a CRQC in the early 2030s. This is not a prediction, but is at the lower end of reasonable estimates for when a CRQC might be available.

There is a need to migrate to quantum-resistant cryptography. It is recommended to start the migration process as soon as possible, which first and foremost involves getting an overview of one's cryptographic inventory, see Section 1.3. For further information and guidance, the interested reader is invited to look at `nsm.no/kvantemigrasjon`.

When the actual change to quantum-resistant cryptography occurs will be different from organization to organization. It is important to note that the pre-quantum versions are being phased out. If one decides to use cryptography which is not quantum-resistant, it is important that one can easily and painlessly upgrade when that time comes.

The following paragraphs detail how cryptography can be vulnerable today even though a quantum computer is some time away, and where these scenarios are relevant switching to quantum-resistant cryptography is rather urgent.

Harvest now, decrypt later When information is encrypted and intended to be kept secret for a long time, it might be vulnerable to future developments in quantum computing. If a message encrypted today needs to remain secret for 30 years, an attacker can store the encrypted file now and decrypt it when a CRQC is available. If this CRQC is built within 20 years, the message has not been kept secret for as long as it should. For information which must remain confidential beyond the early 2030s, one should therefore switch to quantum-resistant algorithms as soon as possible.

Long-lived systems There are cases where one needs to deploy a system which will be operating beyond the early 2030s, but where it is impractical or impossible to modify all cryptographic algorithms in that system. In these cases, it is crucial that those algorithms which cannot be changed are resistant to quantum computers.

Part III
Algorithms

5 Symmetric cryptography

Symmetric cryptography concerns situations where two parties wish to communicate securely and they have access to the same shared information apart from any messages they wish to send. This typically includes a pre-established shared key that must be kept secret.

5.1 General purpose encryption

Applications of encryption often require more than just confidentiality of the underlying message. It is typically necessary to have guarantees of both integrity (meaning that the message was not modified in transit) and authenticity (meaning that the message came from someone with the secret key). These two properties are achieved by computing an *authentication tag*, and transmitting it alongside the ciphertext. An encryption scheme which fulfills all of these properties is called an *authenticated encryption* scheme. Since they are crucial for most applications, it is recommended to use an authenticated encryption scheme for general use. The recommended encryption scheme is:

Algorithm	Block cipher	IV length	Tag length	Standards
AES-GCM	AES-128 AES-256	96 bits	128 bits	FIPS 197 [8] NIST SP 800-38D [16]

AES-GCM is built by using a *block cipher*, AES, in the GCM *mode of operation.* A block cipher encrypts blocks of fixed size, and a mode of operation describes how one can repeatedly use a block cipher to encrypt messages of different lengths.

An important feature of encryption schemes is that even if the same message is encrypted multiple times with the same key, the resulting ciphertext should be different. This is typically achieved by adding an *initialization vector* (IV) which is different for each sent message, and using this IV to compute the ciphertext.

There are a number of important steps that any implementation of AES-GCM needs to follow to be secure:

- The ciphertext must not be decrypted before the authentication tag is verified.
- The same key and the same IV must not be used to encrypt two different messages, and great care must be taken to ensure that this does not happen. It is recommended to use the deterministic IV construction as specified in Section 8.2.1 of [16], and the key must be changed before an IV repeats. If the IVs are instead selected at random, the IV should consist of 96 cryptographically secure random bits (see Section 3), and each key should encrypt no more than 2^{32} messages.
- The message encrypted with one key and one IV must be at most $2^{32} - 2$ blocks of 128 bits, which is approximately 64 GiB.

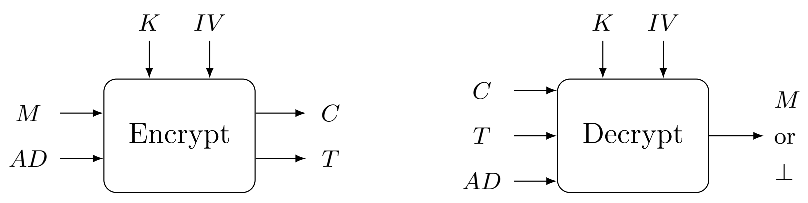

Figure 1: A schematic description of an AEAD scheme

Additional details

AES-GCM is an *authenticated encryption with associated data* (AEAD) scheme. This means that AES-GCM allows for the possibility of sending along *additional data* which will not be kept confidential, but which will be protected against modifications. This can include metadata describing the location of the recipient, which may need to be visible to ensure that the message is delivered to the correct place. If the authentication tag is valid, one can be confident that neither the ciphertext nor the additional data has been modified in transit.

Section 5.1 describes the inputs and outputs of an AEAD scheme. K refers to the secret key, and M and AD are the message and associated data, respectively. C is the ciphertext, the encryption of M. T is the authentication tag, ensuring that neither the ciphertext or associated data are modified in transit. $\perp$ is a special error symbol which indicates that the tag is not valid, and thus either the ciphertext or associated data were modified in transit.

It is possible to use AES-GCM with no message and just additional data, simply ensuring the integrity of whatever is transmitted. However, if confidentiality is never needed, a message authentication code recommended in Section 5.4 is preferred. This is because the properties of AES-GCM are not ideal in this context. Unlike the schemes in Section 5.4, it requires an IV for each message sent, and its security degrades when a lot of information is protected with the same key.

If the ciphertext is decrypted before the authentication tag is verified, this provides an opportunity for an attacker to submit invalid ciphertexts and observe what happens during decryption, and this can potentially be used to find the secret message. It is therefore crucial to verify the authentication tag before decrypting the ciphertext, and abort decryption if authentication fails.

5.2 Disk encryption

General purpose encryption is, as the name suggests, useful for most use cases where encryption is needed. However, there are cases where the features of such an encryption mode are inconvenient. In order to achieve the security properties required of an authenticated encryption scheme, the ciphertext needs to be longer than the plaintext and it is not possible to decrypt one part of the ciphertext. This makes such a scheme unsuitable for the purposes of encrypting an entire storage medium, such as a hard disk. This is due to the properties of how such a storage medium is typically organized. In order to encrypt individual files, the recommendations in Section 5.1 remain valid.

For disk encryption, it is recommended to use a mode of operation which does not provide complete integrity protection. The recommended encryption scheme is:

Algorithm	Block cipher	Standards
XTS-AES	AES-128 AES-256	FIPS 197 [8] NIST SP 800-38E [17]

A key for XTS-AES consists of two *different* keys for the underlying block cipher. XTS-AES uses a *tweak* to encrypt each 128-bit block, which serves a similar purpose to an IV in AES-GCM. The same tweak must never be used twice with the same key.

5.3 Hashing

Hash functions are functions that take a message of arbitrary length as input and produce an n-bit message digest, called a hash, as output. The recommended hash functions are:

Algorithm	Parameter sets	Standards
SHA-3	SHA3-256 SHA3-384 SHA3-512	FIPS 202 [10]
SHA-2	SHA-256 SHA-512/256 SHA-384 SHA-512	FIPS 180-4 [6]

Hash functions are often used internally in other schemes or protocols, so when a hash function is called for elsewhere, one of these hash functions should be used. The sole exception is when hashing passwords, where the guidance from Section 5.6 should be followed.

SHA-3 is a newer hash function which make several related constructions easier, and is a core part of the new quantum-resistant algorithms. Since it is relatively new, it is not very widely used and well-developed implementations of SHA-3 are comparatively rare. SHA-2 is more firmly established, and it is used a lot more and SHA-2 implementations are widely available. It will therefore often be necessary to use SHA-2 for interoperability purposes. For example, SHA-2 is the only hash function that can be used in TLS.

Additional details

Hash functions have several different purposes. One important purpose has to do with the integrity of a message. If the recipient of a message has a correct hash of that message, anyone who wants to change the message needs to find a message which hashes to the same value, which is practically impossible. Hash functions have also found applications in many different areas in cryptography, such as digital signatures and key derivation functions.

5.4 Message authentication

A *message authentication code* (MAC) is designed to provide integrity and authenticity. A MAC is a short tag computed based on some message and a secret key, for which it should be hard to come up with a different message which gets the same tag. A message authentication code does not provide any confidentiality guarantees of the message, and therefore for most purposes an authenticated encryption mode should be used. The recommended message authentication codes are:

Algorithm	Key length	Tag length	Standards
KMAC	$\geq$ 128 bits	$\geq$ 128 bits	NIST SP 800-185 [14]
HMAC			FIPS 198-1 [9]

Note that there are two versions of KMAC, KMAC128 and KMAC256, which aim for different security levels. If one uses KMAC256, the length of the key should be at least 256 bits long.

KMAC is based on the Keccak permutation which forms the basis of the SHA-3 hash function, so it is the natural choice if one uses SHA-3. If one uses SHA-2 instead, HMAC is the natural choice. HMAC makes use of a hash function internally, and it is recommended to use SHA-256.

5.5 Key derivation

Key derivation functions (KDFs) are used to derive keys for use in symmetric cryptographic schemes. KDFs take some secret values (such as a shared secret from a Diffie–Hellman style key exchange) and potentially some additional public values as input, and output one or more symmetric keys of the desired length. Key derivation functions make it possible to update key material in order to avoid using the same key for a long period of time. The recommended key derivation functions are:

Algorithm	Standard
KMAC	Section 4 of NIST SP 800-56C [19]
HKDF	RFC 5869 [2]

KMAC is the natural choice if one uses SHA-3, and HKDF is the natural choice if one uses SHA-2. HKDF uses HMAC internally, and it is recommended to use HMAC with SHA-256 as the internal hash function.

5.6 Password hashing

Passwords need to be treated carefully. The recommendation is to store passwords in a *salted and hashed* form, using a hash function specially intended for this purpose. This means that a random string, called a *salt*, which is stored in the database is added to the password, and then this combined string is passed through a hash function. This ensures that an attacker must attempt to guess each individual user's password. To verify if a password is correct, one adds the salt, hashes the combined string, and checks that the hash one gets is identical to the one already stored. The recommended password hash functions are:

Algorithm	Parameters	Sizes	Standard
Argon2id	$m = 2^{21}$, $p = 4$, $t = 1$	$\geq$ 128-bit salt	RFC 9106 [1]
PBKDF2	Iteration count $\geq$ 600 000	$\geq$ 256-bit tag	RFC 8018 [3]

PBKDF2 is most commonly used with HMAC and SHA-256 (and the suggested parameter is based on this), but any recommended MAC in Section 5.4 is recommended for this purpose as well.

Setting correct parameters is more difficult for password hash functions than for other cryptography, and is best done in consultation with an expert. The parameters should be as large as possible without making the time it takes to verify a login attempt unreasonably long.

Argon2id has better security properties than PBKDF2, and Argon2id is therefore the preferred password hash function. However, PBKDF2 is better established, and implementations of PBKDF2 are more widely available.

6 Asymmetric cryptography

Asymmetric cryptography describes the setting where parties do not have a shared secret key, but parties have a pair of mathematically related private and public keys which perform different operations in a cryptographic scheme.

As detailed in Section 4, asymmetric schemes in use today are vulnerable to the development of a CRQC, and thus there is a need to provide quantum-resistant asymmetric schemes. Because different organizations have individual timelines for migration to quantum-resistant cryptography, recommendations for both quantum-vulnerable and quantum-resistant schemes are given.

The main recommendations for quantum-resistant cryptography are based on mathematical structures known as *lattices*. While schemes based on lattices appear to be resistant to quantum computers, there is not the level of confidence in them as there is in the other cryptographic mechanisms recommended in this document. There is a concern about the possibility of future improvements in attacks against lattice cryptography, and there is a concern about potential vulnerabilities in immature implementations.

For this reason, two additional precautions are recommended. It is recommended to use both a quantum-vulnerable and a quantum-resistant algorithm in *hybrid mode*, where one uses both algorithms together and combine the results in such a way that the hybrid scheme is secure provided at least one of the two initial schemes are secure. This ensures that one does not lose any security by changing to a quantum-resistant scheme, as one still has the security of the quantum-vulnerable scheme to fall back on. Additionally, the recommended parameters are rather conservative to account for potential improvements in cryptanalysis.

Switching to quantum-resistant cryptography comes with an additional communication cost. The recommended quantum-vulnerable algorithms require around 100 bytes to be transmitted between parties, whereas the quantum-resistant algorithms require between 2 and 7 kilobytes. This additional cost can have an impact on certain systems, and is important to be aware of.

6.1 Key establishment

Asymmetric key establishment schemes allow two parties, often called Alice and Bob, to agree on a shared secret key. Using the public and private keys of Alice and/or Bob, the two can send messages back and forth, and end up with a shared symmetric key that no one besides Alice and Bob learns.

Public keys often need to be authenticated, meaning that they are associated with the correct entity. This is important to ensure that no attacker can impersonate legitimate users. If the public keys are not authenticated some other method, such as password-based or multi-factor authentication, must be used to ensure authenticity.

It is recommended to provide *forward secrecy* whenever possible. Forward secrecy means that an attacker who gains access to the current secrets of a user is unable to decrypt messages that were sent in the past. This is typically achieved by periodically agreeing on a new random secret key using a key establishment scheme (typically using long-term keys for authentication only), and deleting any previous keys.

A related notion is that of *post-compromise security*, which guarantees that an attacker who gains access to the current secrets of a user cannot use this to decrypt future messages. Both the definition, and how to achieve this, is more technical, but it is a useful property. Together with forward secrecy, post-compromise security helps reduce the value of each key, and significantly reduces the consequences of a successful attack.

Quantum-vulnerable

The recommended quantum-vulnerable key establishment scheme is:

Algorithm	Elliptic curve	Standards
ECDH	P-256 P-384	NIST SP 800-56A [18] NIST SP 800-186 [15]

It is important to choose the correct algorithm depending on whether the exchange involves static or ephemeral keys. A static key is a key that is used for a longer period of time, and is typically authenticated in some form, such as with a certificate from a third party. An ephemeral key is generated for one specific key exchange, and is typically not authenticated.

Quantum-resistant

The recommended quantum-resistant key establishment scheme is a hybrid scheme which consists of:

Algorithm	Parameter sets for ML-KEM	Elliptic curve for ECDH	Standards
ML-KEM+ECDH+KMAC	ML-KEM-768 ML-KEM-1204	P-256 P-384	FIPS 203 draft [11] NIST SP 800-56A [18] NIST SP 800-186 [15] KEM combiner draft [5]

The recommended scheme establishes one key using ML-KEM and another using ECDH. It then uses these keys, along with transmitted values during the key establishments, as inputs to KMAC, which functions as a *KEM combiner* in this context. The combiner outputs one single key, and its properties guarantees that no attacker can find this key as long as one of ML-KEM and ECDH are secure.

Pre-shared keys

There is an alternative approach to get quantum-resistant key establishment which should only be used as a stopgap measure. This relies on the two parties having a pre-shared secret key which provides the quantum-resistant element. It is crucial that this key is not shared by using a quantum-vulnerable key establishment mechanism. This key can then be updated using a quantum-vulnerable key establishment, ensuring that the same key is not reused and stored for a long period of time. This is inferior to using a quantum-resistant key establishment, but this pre-shared key provides an added level of security against quantum computers until one is able to migrate.

6.2 Digital signatures

A digital signature, much like a regular signature, is meant to certify that a specific message comes from a specific entity. An entity has a key pair consisting of a private signing key and a public verification key. A digital signature is generated using the signing key, and can be verified by anyone with the verification key. If a signature passes the verification check, one can be confident that the message indeed came from the entity associated with the public key.

Quantum-vulnerable

The recommended quantum-vulnerable digital signature schemes are:

Algorithm	Parameters	Standards
ECDSA	P-256 P-384	FIPS 186-5 [7] NIST SP 800-186 [15]
RSASSA-PSS	$n \geq 3000$ bits $2^{16} < e < 2^{256}$	RFC 8017 [4] FIPS 186-5 [7]

The *Elliptic Curve Digital Signature Algorithm* (ECDSA) requires a value k between 1 and a large number n for each signature. This value can either be generated at random, or computed deterministically from the message and the private key. If this number is generated at random, it must be generated according to Appendix A.3 of FIPS 186-5 [7] using cryptographically secure random numbers (see Section 3). If this number is computed deterministically, it must be computed according to Appendix A.3.3 of FIPS 186-5 [7]. This k must be kept secret, and also protected against any kind of leakage, such as via side-channel attacks.

The definition of the RSASSA-PSS signature scheme is found in RFC 8017 [4], but a number of additional restrictions which should be followed are found in FIPS 186-5 [7], in particular when it comes to properly generating RSA key pairs.

ECDSA is a more efficient algorithm by most metrics, and safely implementing RSA is particularly challenging. However, RSA is a widely used industry standard for digital signatures, and it is in certain cases the only supported signature scheme. For these reasons, it is included here, but ECDSA should be preferred.

Quantum-resistant

The recommended quantum-resistant digital signature schemes are:

Algorithm	Parameters		Standards
ML-DSA+ECDSA	ML-DSA	ECDSA	FIPS 204 draft [12]
	ML-DSA-65	P-256	FIPS 186-5 [7]
	ML-DSA-87	P-384	NIST SP 800-186 [15]
SLH-DSA	See discussion below		FIPS 205 draft [13]

For the first recommendation, a signature of a message must consist of *both* an ECDSA signature and an ML-DSA signature, and both component signatures need to be valid for the entire signature to be valid. It is therefore crucial that the signature verification algorithm verifies both signatures. Furthermore, keys should be generated specifically for this hybrid scheme and not used for non-hybrid signature schemes as well.

SLH-DSA is a signature scheme based on hash functions, which is secure as long as the underlying hash function is secure. Since hash function design and implementation is well-understood, it is recommended to use SLH-DSA without adding a quantum-vulnerable signature. However, SLH-DSA signatures are quite large (8-50kB depending on the parameter set) and signing messages is relatively slow. This makes SLH-DSA an option which is unsuitable in many cases, but the applications where the performance drawbacks are not a concern it is a good choice due to the confidence one has in its security. FIPS 205 contains 12 parameter sets for SLH-DSA. All parameter sets are recommended, but choosing the appropriate set for a specific use case is best done in consultation with an expert.

7 Quick reference

Below is a reference list, briefly describing each use case and the algorithm(s) recommended for that use case. A short summary of the recommended parameters are given, except when quickly summarizing the recommended parameters can not be done succinctly. Further details about all recommendations are found in the relevant sections earlier in this document.

Symmetric cryptography

Further information, including descriptions of the parameters and references to standards documents, about the following schemes can be found in Section 5.

Use case	Algorithms	Parameters
Encrypting messages between two parties who share a secret key	AES-GCM	AES-128/256 96 bit IV 128 bit tag
Encrypting data on disk	XTS-AES	AES-128/256
Hashing information into a small digest	SHA-3 SHA-2	Output size $\geq$ 256 bits
Integrity checking	KMAC HMAC	Key length $\geq$ 128 bits Tag length $\geq$ 128 bits
Deriving keys from a secret value	KMAC HKDF	
Hashing passwords	Argon2id PBKDF2	

Asymmetric cryptography

Further information, including detailed descriptions of the parameters and references to standards documents, about the following schemes can be found in Section 6.

Quantum-vulnerable

Use case	Algorithms	Parameters
Establishing a shared secret between two parties	ECDH	P-256/384
Digitally signing a message	ECDSA	P-256/384
	RSA	$n \geq 3000$ bits

Quantum-resistant

Note that the first two recommendations are hybrid schemes, so each recommendation consists of two algorithms and a way to combine them.

Use case	Algorithms	Standard
Establishing a shared secret between two parties	ML-KEM + ECDH + KMAC	ML-KEM-768/1024 + P-256/384
Digitally signing a message	ECDSA + ML-DSA	ML-DSA-65/87 + P-256/384
	SLH-DSA	

8 Protocols

A cryptographic protocol describes how multiple parties interact and use cryptographic schemes to achieve a specific security objective. Examples include secure communication between two people, between a user and a website, or between a developer and a remote server.

Designing protocols requires expertise, and it is recommended to use pre-existing and modern protocols that are well-suited for the application. Some examples are TLS 1.3 and IPSec. When using protocols which allow for the use of different algorithms, one should use the algorithms recommended in this document.

NSM will release more specific recommendations for some very common protocols at a future time.

References

[1] Alex Biryukov et al. *Argon2 Memory-Hard Function for Password Hashing and Proof-of-Work Applications.* RFC 9106. Sept. 2021. DOI: `10.17487/RFC9106`. URL: `https://www.rfc-editor.org/info/rfc9106`.

[2] Dr. Hugo Krawczyk and Pasi Eronen. *HMAC-based Extract-and-Expand Key Derivation Function (HKDF).* RFC 5869. May 2010. DOI: `10.17487/RFC5869`. URL: `https://www.rfc-editor.org/info/rfc5869`.

[3] Kathleen Moriarty, Burt Kaliski, and Andreas Rusch. *PKCS #5: Password-Based Cryptography Specification Version 2.1.* RFC 8018. Jan. 2017. DOI: `10.17487/RFC8018`. URL: `https://www.rfc-editor.org/info/rfc8018`.

[4] Kathleen Moriarty et al. *PKCS #1: RSA Cryptography Specifications Version 2.2.* RFC 8017. Nov. 2016. DOI: `10.17487/RFC8017`. URL: `https://www.rfc-editor.org/info/rfc8017`.

[5] Mike Ounsworth, Aron Wussler, and Stavros Kousidis. *Combiner function for hybrid key encapsulation mechanisms (Hybrid KEMs).* Internet-Draft draft-ounsworth-cfrg-kem-combiners-05. Work in Progress. Internet Engineering Task Force, Jan. 2024. 14 pp. URL: `https://datatracker.ietf.org/doc/draft-ounsworth-cfrg-kem-combiners/05/`.

[6] National Institute of Standards and Technology. *FIPS 180-4, Secure Hash Standard (SHS).* 2015. URL: `https://doi.org/10.6028/NIST.FIPS.180-4`.

[7] National Institute of Standards and Technology. *FIPS 186-5, Digital Signature Standard (DSS).* 2023. URL: `https://doi.org/10.6028/NIST.FIPS.186-5`.

[8] National Institute of Standards and Technology. *FIPS 197, Advanced Encryption Standard (AES).* 2001. URL: `https://doi.org/10.6028/NIST.FIPS.197-upd1`.

[9] National Institute of Standards and Technology. *FIPS 198-1, The Keyed-Hash Message Authentication Code (HMAC).* 2008. URL: `https://doi.org/10.6028/NIST.FIPS.198-1`.

[10] National Institute of Standards and Technology. *FIPS 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* 2015. URL: `https://doi.org/10.6028/NIST.FIPS.202`.

[11] National Institute of Standards and Technology. *FIPS 203 (Initial Public Draft), Module-Lattice-Based Key-Encapsulation Mechanism Standard.* 2023. URL: `https://doi.org/10.6028/NIST.FIPS.203.ipd`.

[12] National Institute of Standards and Technology. *FIPS 204 (Initial Public Draft), Module-Lattice-Based Digital Signature Standard.* 2023. URL: `https://doi.org/10.6028/NIST.FIPS.204.ipd`.

[13] National Institute of Standards and Technology. *FIPS 205 (Initial Public Draft), Stateless Hash-Based Digital Signature Standard.* 2023. URL: `https://doi.org/10.6028/NIST.FIPS.205.ipd`.

[14] National Institute of Standards and Technology. *NIST SP 800-185, SHA-3 Derived Functions: cSHAKE, KMAC, TupleHash, and ParallelHash.* 2016. URL: `https://doi.org/10.6028/NIST.SP.800-185`.

[15] National Institute of Standards and Technology. *NIST SP 800-186, Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters.* 2023. URL: `https://doi.org/10.6028/NIST.SP.800-186`.

[16] National Institute of Standards and Technology. *NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.* 2007. URL: `https://doi.org/10.6028/NIST.SP.800-38D`.

[17] National Institute of Standards and Technology. *NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices.* 2010. URL: `https://doi.org/10.6028/NIST.SP.800-38E`.

[18] National Institute of Standards and Technology. *NIST SP 800-56A Rev. 3, Recommendation for Pair-Wise Key-Establishment Schemes Using Discrete Logarithm Cryptography.* 2018. URL: `https://doi.org/10.6028/NIST.SP.800-56Ar3`.

[19] National Institute of Standards and Technology. *NIST SP 800-56C Rev. 2, Recommendation for Key-Derivation Methods in Key-Establishment Schemes.* 2020. URL: `https://doi.org/10.6028/NIST.SP.800-56Cr2`.

[20] National Institute of Standards and Technology. *NIST SP 800-57 Part 1 Rev. 5, Recommendation for Key Management: Part 1 – General.* 2020. URL: `https://doi.org/10.6028/NIST.SP.800-57pt1r5`.

[21] National Institute of Standards and Technology. *NIST SP 800-90A Rev. 1, Recommendation for Random Number Generation Using Deterministic Random Bit Generators.* 2015. URL: `https://doi.org/10.6028/NIST.SP.800-90Ar1`.

[22] National Institute of Standards and Technology. *NIST SP 800-90B, Recommendation for the Entropy Sources Used for Random Bit Generation.* 2018. URL: `https://doi.org/10.6028/NIST.SP.800-90B`.

A Standards and standards organizations

This document contains references to *standards*. These are documents which describe a certain cryptographic algorithm in detail. There are a number of *standards organizations* which produce such standards. This document refers to standards from NIST and the IETF. NIST is the National Institute of Standards and Technology, an agency of the United States Department of Commerce. The IETF is the Internet Engineering Task Force, the organization responsible for the technical standards which govern the Internet. They do not dictate what NSMs recommendations are, and their role in this document is solely to provide definitions of the cryptographic algorithms that are recommended. The standards published by these organizations are typically not a solely internal endeavour, but are based on open work by academics, engineers, governments and other interested parties.