Cryptography Standards Policy

Document Number	ISMS-ENGINEERING-041
Date	23/06/2022
Revision No	2.0
Information Classification	Internal
Prepared By	Gaurav Agarwal
Approved By	Vinay Rai

Revision and Approval

Rev.	Date	Changes	Approved By
1	23-03-2021	Original document	Vinay
2	23-06-2022	Added: 1. Key rotation requirement 2. AES-256 is recommended for symmetric encryption	Vinay

Main policy rules

This policy consists of the following general rules. You must follow these rules to avoid the risks of not using cryptography where it is needed and the risk of using cryptography incorrectly.

1. Consider the use of encryption whenever the confidentiality of an asset is important. Cryptography can be used when sending data and when storing data, both can be relevant.
2. Confidential and privacy sensitive information that need to be shared, must be protected via encryption. The password must be share through an alternative trusted channel such as official email or messaging app.
3. Devices such as laptops and mobile phones must be protected with a password. Data encryption must be enabled where available.
4. When considering encryption, consider the use of digital signatures or hash functions as well. Encryption only helps with confidentiality. In most cases where confidentiality is important, integrity is also important and digital signatures and hash functions are important tools to ensure integrity.
5. Make sure that you only use strong cryptographic algorithms. The difference between weak and strong algorithms is explained further on in this document.
6. Only use algorithms that have been published and have been scrutinized by researchers. Never invent your own algorithms or use non-public algorithms.
7. Make sure that you understand the risks of short key lengths and set minimum key lengths for algorithms that let you choose the key size.
8. Make sure that key management is in place. You need to make sure keys are generated securely, stored in a secure way, and destroyed when no longer needed. Also make sure that multiple people have access to keys to avoid loss of keys when people leave the organisation.
9. Symmetric algorithm keys and private keys are like passwords. For instance, these keys should never be re-used and should also be changed at least yearly. Public keys are typically not confidential and can be changed less frequently.
10. Make sure that all keys are randomly generated using a secure random number generator. You cannot use common words as keys. Keys should also never be stored in source code
11. When you are adding or changing features that rely on cryptography during software development, a second developer must review the source code and check against the rules in this policy. Note that you should never design your own algorithms (see note below), this policy is intended for cases where you invoke an existing algorithm for a specific purpose.

12. When deciding to use certain products with cryptographic features (e.g. encryption software), you must check that the product uses a strong cryptographic algorithm and you must do a google search to check if this product has known weaknesses.
13. Note that exporting cryptography to certain countries is illegal because encryption technology has historically been classified as military technology. Most of these restrictions have been lifted, especially for common cryptography in products available to consumers. Check relevant cryptography export rules before travelling with devices containing cryptography or before exporting software containing cryptography.

These are all the main rules. Below we have given some additional, often technical details. Everyone involved in the information security team or in secure software development must be aware of these main rules. The details need to be understood by IT staff involved in the application of cryptography.

Key management

Using encryption is like putting a lock on a room. Instead of having to guard the room, you only must guard a key to prevent other people accessing your data. When using encryption, protecting the digital keys is therefore very important.

The first aspect of key management is generating the keys. Any key you use should not be predictable. One can typically use a computer program for generating keys if this program has been designed for generating hard to predict keys. E.g. the programming language Java has a strong random number generator that can be used for generating keys, but it also has a faster but insecure random function that should not be used.

Another important aspect is secure storage of keys. One must store keys in such a way that access is restricted. It is not acceptable to store keys in a public place in a company, or to place keys in source code that is accessible to all developers. Key rotation should be used to reduce attack surface.

Types of encryptions

There are different types of cryptography that can be used for different purposes. Before using cryptography, it is important to understand all these types and use the right type. which type should be applied depends on the properties you are interested in i.e. confidentiality, integrity, and availability.

The main types are:

- **Symmetric encryption**. A symmetric encryption algorithm uses a key (small string of data) to scramble a plaintext into a ciphertext. The ciphertext is not readable to anyone who does not have access to the key.
- **Asymmetric encryption** (also known as **public key encryption**). An asymmetric encryption algorithm uses a key pair consisting of a public key part and a private key part. The algorithm takes the public key part and a plaintext to create a ciphertext. The ciphertext is only comprehensible for parties those have the private key part.
- **Hash functions (also known as one-way functions)**: A hash function takes a large plaintext and computes a small hash or fingerprint. Anyone with the plaintext can compute the same fingerprint. If you do not have the plaintext, it is not possible to discover anything about the plaintext from the fingerprint.
- **Digital signatures**. Like asymmetric encryption, digital signature algorithms use a key pair consisting of a public key part and a private key part. The algorithm takes the private key part and a plaintext to create a digital signature file. Anyone with the public key and the plaintext can check the validity of the signature. It is not possible to create a valid signature without the private key part.

Encryption (symmetric and asymmetric) helps with confidentiality. Hash functions and digital signatures help with integrity. When both properties are important, a combination is often needed.

Protection in transit and at rest

When analysing whether data needs to be protected with cryptography, it is important to realize that data needs to be protected when in transit and when at rest. In transit means when data is transported from one location/system to another. For example, when it is sent over the Internet or over a network. At rest means when the data is stored for later use, for instance on a disk or in a database in the cloud. In both the situation, security is important. Whether encryption is needed should be decided based on the importance of the information and the risks for the type of storage or transit. Here are some guidelines:

- You should have an information asset inventory that should tell you whether information should be kept confidential. Use this register and your risk management process for any decisions
- The Internet itself is an open network and any information that is sent over the internet must be protected. It is possible to encrypt all information at the network level (using SSL). You can also encrypt individual files.

- Encryption comes with a performance penalty and is often not used within systems where it is not needed. E.g. a database that resides on a local server and cannot be accessed from the outside does not have to be encrypted.
- Devices that can be lost or stolen (e.g. phones and laptops) are an important security risk. Encryption is highly recommended for such devices as an additional security measure.

Selecting strong cryptographic algorithms

When building your own software or when using a product that offers multiple algorithms, it is important to select a strong algorithm. Weak algorithms can help to makes sure that data cannot be read by casual users but does not provide real protection against determined hackers. We strongly recommend against the use of weak cryptography because it provides a false sense of security. If it is worth using cryptography, it is worth doing well.

Strong algorithms have been tested by cryptographic experts and are hard to break even for determined attackers. Below we have listed several strong algorithms that you should use. Note that you should refrain from creating your own algorithms or even write your own implementation of these algorithms, as this introduces additional risks. Please use a library that is well-tested.

Recommended strong algorithms:

- Symmetric encryption: AES (256 bit recommended)
- Asymmetric encryption: RSA (2048 bit recommended, at least 1200 bits required). Also suitable according to ENISA is Elliptic Curve cryptography with at least 256 bits key.
- Hash functions: SHA2 (four sizes, 256 bits is recommended).
- Digital signatures: RSA (good 2048 bits, ok 1200 bits). Alternatives are DSA, ECDSA.

Known weak algorithms

The list of weak algorithms is almost endless. Here we list some known algorithms

- DES. This is one of the oldest algorithms. It is well designed, but it has a key length of only 56 effective bits. This is no longer sufficient because computers have become much faster. You need at least 150 effective bits.
  - TripleDes or 3DES. This algorithm applies DES three times with different keys to increase the effective key size. It is slow and the effective key size is only 128 bits, still less that the now recommended 150 bits.

- IDEA. This algorithm has been designed for optimal speed while still being probably secure. It can be the right choice if your hardware is limited.
- RC1, RC2, RC3, RC4. These are all algorithms designed by well-known cryptologist Ron Rivest. (RC stands for Ron's code). RC5 and RC6 are strong algorithms, the other algorithms are interesting from a research viewpoint but not recommended for actual use.
- Blowfish. This algorithm is designed by Bruce Schneier and is quite fast. It is however weak because it has a small block size. You should use Twofish or AES instead h
- CAST or CAST-128. This algorithm also has a 64-bit block size, not enough for current standards. There is a better variant CAST-256.
- MD5. MD5 is a well-known hash function. It is well-designed but quite old and the fingerprint that it produces only contains 64 bits. This is not enough against modern computers. Collisions can be found with a birthday attack. MD5 can be used in non-cryptographic settings, e.g. as a quick check if files are different.
- SHA-1. This is a well-designed hash function with a 160 bits fingerprint. This is less than the now recommended 256 bits. The SHA2 algorithm with a 256 bits output is a better choice.