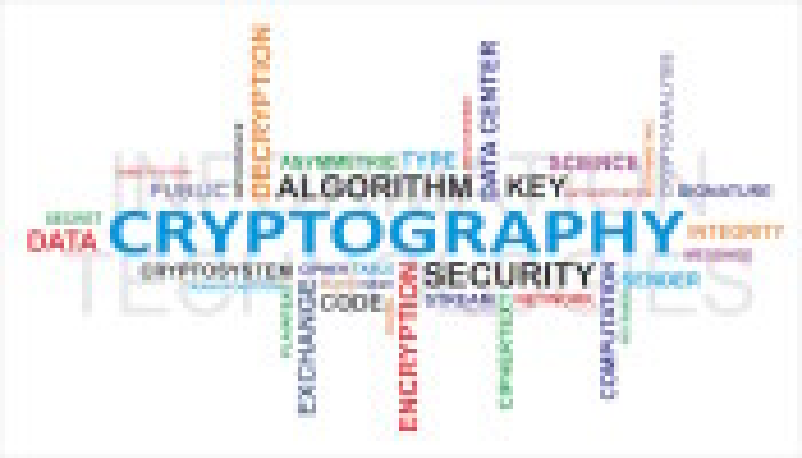
One of the basic questions in considering encryption is to understand the differences between **symmetric** and **asymmetric encryption** methods, and where to apply each method to best protect your data.

Asymmetric encryption

Asymmetric encryption, also known as public-key encryption, utilizes a pair of keys – a public key and a private key. If you encrypt data with the public key, only the holder of the corresponding private key can decrypt the data, hence ensuring confidentiality.



The first practical algorithm for asymmetric encryption was proposed by Diffie and Hellman in 1976. Subsequently, RSA became the most widely deployed asymmetric encryption algorithm.

Many “secure” online transaction systems rely on asymmetric encryption to establish a secure channel. SSL, for example, is a protocol that utilizes asymmetric encryption to provide communication security on the Internet.

An asymmetric encryption algorithms typically involve exponential operations, they are not lightweight in terms of performance. For that reason, asymmetric algorithms are often used to secure key exchanges rather than used for bulk data encryption.

Symmetric encryption

Symmetric encryption, as the name suggests, means that the encryption and decryption operations utilize the same key. For two communicating parties using symmetric encryption for secure communication, the key represents a shared secret between the two.

There exist many symmetric encryption algorithms. A few of the well-known ones include AES, DES, Blowfish, and Skipjack.

Symmetric encryption is typically more efficient than asymmetric encryption, and is often used for bulk data encryption.

Attack a cryptosystem

Given enough computing resources, both symmetric and asymmetric encryption can be broken.

The most basic way to attack a symmetric cryptosystem is brute-force attacks, where you essentially try every combination of a key. For a 128-bit key, there are 2^128 combinations to attempt, which requires extensive computing resources. Other cryptanalysis attacks, including chosen-ciphertext and chosen-plaintext attacks, can be more efficient than brute-force, but they require a priori knowledge to work.

To guard against brute-force attacks, the key length of a symmetric cryptosystem needs to be sufficiently long. The **Advanced Encryption Standard (AES)** algorithm

with 256-bit key is considered secure enough for most purposes. And the implementation can be made relatively efficient.

The best way to attack a well-designed RSA implementation is through factoring of RSA's public modulus, which is a large number. Factoring large numbers, with today's best known factoring techniques, is a compute-intensive problem.

RSA (the company), ran a factoring challenge from 1991 to 2007, during which an RSA 768-bit modulus was factored successfully. In 2010, a 1024-bit RSA modulus was factored with relatively low cost.

Today, RSA implementations typically require a 2048-bit key to be secure. For ultra sensitive operations, you would want 4096-bit keys. Of course the longer the key length, the more expensive it is to run the encryption and decryption operations.

Check Out the Cloud Encryption Resource Center for Tons of Free Resources

Which Method Is Right For You?

How to choose symmetric vs. asymmetric cryptosystems? Here are a few tips:

The case for symmetric-key cryptography

- Symmetric key cryptosystems have been shown to be more efficient and can handle high rates of data throughput
- Keys for symmetric-key cryptosystems are shorter, compared to public key algorithms
- Symmetric key ciphers can be composed together to produce a stronger cryptosystem.

The case for asymmetric-key cryptography

- In a large network, asymmetric key cryptography yields a more efficient system for key management, as you don't have to manage pair-wise keys for every communicating pair.
- Asymmetric key cryptosystems are good for digital signatures and key exchange use cases
- In many cases, the public and private key pairs in an asymmetric-key cryptosystem can remain intact for many years without compromising the security of the system. SSL certificates are one such example.

One of the most interesting facts about asymmetric key cryptosystems is that the security of these systems is based on a small set of number-theory problems that are presumed difficult but were never mathematically proven to be difficult. Factoring, for instance, is one such problem. Advances in number theory could one day render factoring a much easier problem hence diminishing security of many asymmetric key cryptosystems.