A Simple Guide to Cryptography

Cryptography is an ancient mathematical science that was originally used for military communications, and designed to conceal the contents of a message should it fall into the hands of the enemy. Recent developments in cryptography have added additional uses, including mechanisms for authenticating users on a network, ensuring the integrity of transmitted information and preventing users from repudiating (i.e. rejecting ownership of) their transmitted messages.

In today's world of electronic commerce on the Internet, the need for secure communications is obviously crucial. Cryptographic technologies provide enterprises with the best mechanisms of protecting their information, without putting the business at risk by exposing it on the Net.

What is Encryption?

Encryption is the name given to the process of applying an algorithm to a message, which scrambles the data in it—making it very difficult and time consuming, if not practically impossible, to deduce the original given only the encoded data. Inputs to the algorithm typically involve additional secret data called **keys**, which prevents the message from being decoded—even if the algorithm is publicly known.

The safekeeping of keys, in other words their generation, storage and exchange, is of paramount importance to ensure the security of the data. There is no point applying the strongest levels of cryptographic algorithms, if your keys are stored on a scrap of paper in your in-tray.

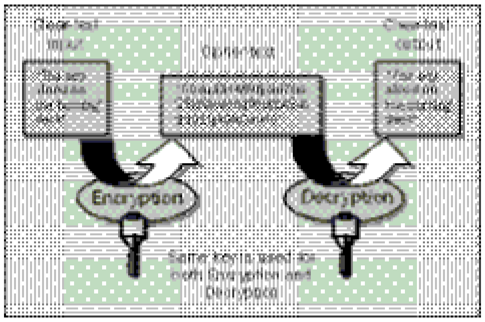
The strength of the encryption is dependent on two basic items: the nature of the mathematical algorithm and the size of the keys involved. Under US arms regulations, the length of the key that can be used in exported software is limited. However, there is no limitation on the level of encryption used *within* the US, or sold in Canada.

Unfortunately the 40-bit encryption limit, which has been in force up until recently, has been proven to provide little security from attack. Today's powerful processors, costing just a few hundred dollars, can crack such a message in a few hours by using brute force—that is, by trying every possible key until the decrypted message has been found. More expensive supercomputers can crack such messages in sub-second times! Each extra bit in the key doubles the time needed for the brute force attack, and most experts now claim that 128-bit keys are required to ensure complete confidence, and are vital for markets such as electronic commerce.

Many non-US companies have now developed add-on cryptographic products, using 128-bit key technology, to fill the vacuum left by the US software industry's inability to compete in this market. Naturally, there is a lot of discussion between concerned parties, and the future of these export restrictions is unclear.

Symmetric Cryptography Ã‚Â– Secret Keys

In **symmetric cryptography**, the encryption algorithm requires the same secret key to be used for both encryption and decryption. Because of the type of key, this is sometimes called **secret key encryption**. This diagram shows how it works:

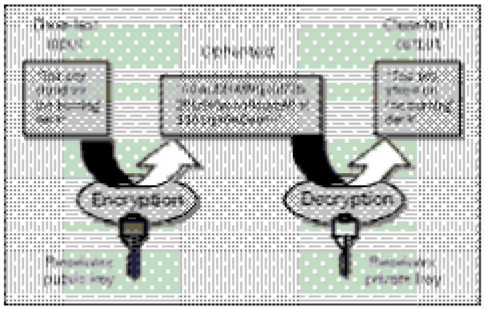


The advantage of these algorithms is that they are fast and efficient. However, the problem is that of key exchange—it is, the mechanism for safely ensuring both parties, the sender and the receiver, have the secret key. This is one of the weakest areas of symmetric cryptography. How do you send the key to your partners? You cannot just send it in an email message, because it could be intercepted and, possibly unknowingly, compromise your security. Furthermore, how can you be sure that your partners will keep your key secure?

Asymmetric Cryptography Ã‚Â– Public/Private Keys

One solution to the problem of key security is **asymmetric cryptography**. This uses two keys that are mathematically related. One key is called the **private key** and is never revealed, and the other is called the **public key** and is freely given out to all potential corespondents. The complexity of the relationship between the public key and the private key means that, provided the keys are long enough, it is practically impossible to determine one from the other. The one problem with asymmetric cryptography is that the processing required is very CPU intensive and this can cause potential performance problems when many simultaneous sessions are required.

The almost universal public/private key algorithm is named **RSA** after its creators (Ron Rivest, Adi Shamir, and Len Adleman), and patented by RSA Data Security Inc. in 1977. A sender uses the receiver's public key to encrypt the message. Only the receiver has the related private key to decrypt the message. This is shown here:



Digital Signature Encryption

An additional use of RSA is in **digital signatures**, which involves swapping the role of the private and public keys. If a sender encrypts a message using their private key, everyone can decrypt the message using the sender's public key. A successful decryption implies that the sender, who is the only person in possession of their private key, must have sent the message.

This also prevents **repudiation**, that is, the sender cannot claim that they did not actually send the message. A piece of data encrypted with a private key is called a digital signature. Common practice is to use a message digest as the item of data to be encrypted.

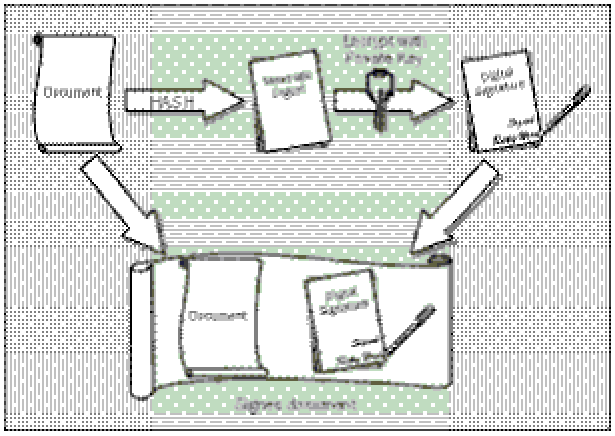
Using a Message Digest

A **message digest** is a digital fingerprint of a message, derived by applying a mathematical algorithm on a variable-length message. There are a number of suitable algorithms, called **hash functions**, each having the following special properties:

- The original message (the input) is of variable-length.
- The message digest (the output) is of a fixed-length.
- It is practically impossible to determine the original message (the input) from just the message digest (the output). This is known as being a **one-way hash function**.
- It is practically impossible to find two different messages (the inputs) that derive to the same message digest (the output)—this is known as being a **collision-free hash function**.
- The algorithm is relatively simple, so when computerized it is not CPU-intensive.
- The calculated digest is (often considerably) smaller than the item it represents.

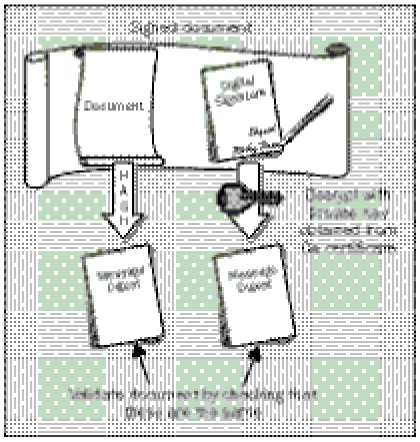
You can use message digests to guarantee that no one has tampered with a message during its transit over a network. Any amendment to the message will mean that the message and digest will not correlate. Also, message digests can also be used to supply proof that an item of information, such as a password, is known—without actually sending the password or information in the clear.

The most common message digest algorithms designed for 32-bit computer systems are **MD4**, **MD5** and **Secure Hash Algorithm** (SHA, which offer—in that order—increasing levels of security, and therefore CPU usage. The next diagram illustrates how a message digest is used to digitally sign a document:



Note that this document is **signed** (its integrity and origin are assured) but it is not **encrypted** Ã‚Â– anyone could look at the original document included with the signed digest. This does not imply that the document cannot be encrypted as well, however.

This diagram shows how the same document is validated:


Signed document
Document
Digital Signature
HASH
Decrypt with Private Key obtained from the certificate
Message Digest
Message Digest
Validate document by checking that these are the same

Digital Certificates and Certificate Authorities

A digital certificate is an item of information that binds the details of an individual or organization to their public key. The most widely accepted format for digital certificates is the X.509 standard, and is relevant to both clients and servers. If we can obtain access to someone's certificate we have their public key, and can therefore get involved in the secure communications already discussed.

But what is there to stop anybody just creating a false certificate, and pretending that they are someone else? The solution is **Certificate Authorities** (CAs), who are responsible for the issuing of digital certificates. A CA is a commonly known **trusted third party**, responsible for verifying both the contents and ownership of a certificate.

There is an ever-increasing number of CAs. Different CAs will employ different amounts of effort in their verification processes, and they must publicly divulge what checks they perform. Then users can apply the appropriate levels of trust for each CA they encounter. Also, different classes of certificates are available, which reflect the level of assurance given by the CA—a certificate for users who just surf the web requires less verification than a certificate for a business server. If two entities trust the same CA, they can swap digital certificates to obtain access to each other's public key, and from then onwards they can undertake secure transmissions between themselves.

Digital certificates include the CA's digital signature (i.e. information encrypted with the CA's private key). This means that no one can create a false certificate. The public keys of trusted CA's are stored for use by applications like Internet Explorer as we will see in a while.