Public-Key Cryptography

Grace Howard

December 2023

Symmetric key encryption schemes

In *symmetric key encryption*, there is a secret key k shared between the two parties. Symmetric key encryption schemes consist of 3 algorithms: one which generates the key, one which encrypts the message using that key, and one which decrypts the message using that key. It also uses 3 sets: the set K of keys, the set P of plaintext, and the set C of ciphertext.

The algorithm which generates the key works as follows: Given some security parameter 1^κ, the function G outputs the secret key $k \in K$, or

$$G(1^\kappa) \to k.$$

Similarly, the encryption algorithm takes the key k and a message $p \in P$ and outputs ciphertext $c \in C$, or

$$E(k, p) \to c.$$

The decryption algorithm does the same, but using the ciphertext c as input and outputting a message $p \in P$, or

$$D(k, c) \to p.$$

This scheme is said to have *correctness* if for all $p \in P, k \in K$,

$$D(k, E(k, p)) = p.$$

An example of this is a shift cipher, which encrypts a message by shifting each letter some number of positions forward or backward in the alphabet. Let $K = \{1, .., 26\}$, $P = \{a, b, .., z\}$, and $C = \{A, B, .., Z\}$. Then, informally, if the shift is 5 then the message "five" is "KNAJ" if shifted to the left and "ADQZ" if shifted to the right. Using the formal definition, $G()$ is some $k \in K$. The function $E(k, p)$ is defined based on the length l of the message. Then, for some message

$$p = p_1 p_2 \dots p_l,$$

$$c_i = p_i + k \pmod{26}$$

for all $i \in \{1, \dots l\}$. Similarly, the algorithm $D(k, p)$ is defined in terms of the length of c (or p). For some ciphertext

$$c = c_1 c_2 \dots c_l,$$

$$p_i = c_i - k \pmod{26}.$$

This is not particularly secure. Firstly, it is not terribly difficult to use brute force to find the key. There would not be many words which are meaningful when guessing, so the options would be narrowed down very fast. Secondly, symmetric encryption in general is problematic when it comes to security. At some point, a key must be created and conveyed to another party. In transit it could be observed and copied. Given that anyone with the key can decrypt the message, this is quite bad.

Asymmetric key encryption

Public key cryptography aims to resolve such shortfalls. It relies on an asymmetric scheme, which uses a pair of keys: a *public* key (for encryption) and a *private* key (for decryption). Anyone can use the public to encrypt information, and the private key is kept secret. This eliminates the possibility of the key being observed in transit, as the private key is never being conveyed.

Public key encryption schemes make use of *one-way functions*. A function $f : X \to Y$ is a one-way function if it is "easy" to compute $f(x)$ for all $x \in X$, but "hard" to compute f^{-1}.

Example

Consider two primes $p = 11$ and $q = 3$. Then, let $n = pq = 33$ and define $X = \{1, 2, \ldots, 32\}$. Additionally, define the function $f : X \to \mathbb{N}$ by $f(x) = x^3 \pmod{n}$. Then, for instance, $f(31) = 25$. Without x, inverting f is challenging. If p and q are given, it is quite simple, though. A *trapdoor one-way function* is a one-way function $f_k : X \to Y$, where, given k, one can find an $x \in X$ for which $f(x) = y$ for all $y \in \mathrm{Im} f$.

RSA

A number n is known by both parties. Then, split the plaintext into $\log_2(n)$ bits. Then each block represents some number $M < n$. Then encryption is defined as

$$E \equiv M^e \pmod{n}$$

and decryption is defined as

$$D = C^d \equiv (M^e)^d \pmod{n}$$

where e and d are chosen values. The public key is $\{e, n\}$ and the private key is d. This is dependent on a number of things. Firstly, it must be possible to find values of e, d, and n such that

$$M^{ed} \pmod{n} \equiv M$$

for all $M < N$. Additionally, it must be unreasonable to find d given e and n.

RSA correctness

The goal is to show that there exist $e, d, n \in \mathbb{N}$ such that

$$M^{ed} \pmod{n} \equiv M$$

for all $M < N$. This equation holds if e and d are multiplicative inverses $\mod \phi(n)$.

Proof. Consider when

$$d \equiv e^{-1} \pmod{\phi(n)}$$

or

$$de \equiv 1 \pmod{\phi(n)}.$$

This is true if and only if d and e are such that $\gcd(\phi(n), d) = 1$. □

Example

Firstly, use two (generally large) distinct primes p and q. Here let $p = 47$ and $q = 71$. Then compute $n = pq$, which is 3337 in this example, and $\phi = (p-1)(q-1)$, which is 3220 here. After this, e must be selected, where $\gcd(e, \phi) = 1$ and $1 < e < \phi$. Suppose $e = 79$ was selected for this example. Then, compute $d = e^{-1} \pmod{\phi}$. Here, $d = 79^{-1} \pmod{3220} = 1019$. Now, the public key is $\{79, 3337\}$ and the private key is $\{1019, 3337\}$. The message will be encrypted with the public key. First, break the message M into blocks. For example,

$$688 \quad 232 \quad 687 \quad 966 \quad \cdots .$$

Then, compute

$$C_i = M_i^e \pmod{n}.$$

Here, $C_1 \equiv 688^{79} \equiv 1570 \pmod{3337}$ and so on. The message will be decrypted with the private key. This will mean calculating

$$M_i = C_i^d \pmod{n}.$$

Here, $M_1 = 1570^{1019} \equiv 688 \pmod{3337}$ and so on.

Diffie-Hellman key exchange

The Diffie-Hellman key exchange is an algorithm for exchanging keys over an insecure channel.

Definition 0.1. A *primitive root* g of a prime number p is a number whose powers generate $1, \ldots, p-1$.

A *discrete logarithm* is given g, $g^m \pmod{p}$, and p find m. Computing discrete logarithms is very difficult. The Diffie-Hellman key exchange is as follows: Firstly, a very large prime p is selected as well as a primitive root g of p. These numbers are public. Then, one party generates a random number n_1 and conveys

$$m_1 = g^{n_1} \pmod{p}$$

to the other party. Similarly, the other party generates a random number n_2 and conveys

$$m_2 = g^{n_2} \pmod{p}$$

to the other party. Then, the first party's secret key,

$$K = (m_2)^{n_1} \pmod{p}$$

is calculated. Similarly, the second party's secret key

$$K = (m_1)^{n_2} \pmod{p}$$

is calculated.

Example

Suppose $p = 11$ and $g = 2$. Then, suppose $n_1 = 4$, so

$$m_1 = 2^4 \equiv 5 \pmod{11}.$$

Similarly, suppose $n_2 = 6$, so

$$m_2 = 2^6 \pmod{11} = 9.$$

Then,

$$K = 9^4 \equiv \pmod{11}$$

and

$$5^6 = 5 \pmod{11}.$$

Correctness

Proof. Consider $K = (m_2)^{n_1}$

$$\begin{aligned} &\equiv (g^{n_2})^{n_1} \\ &\equiv (g^{n_2})^{n_1} \\ &\equiv (g^{n_1})^{n_2} \\ &\equiv (m_1)^{n_2} \pmod{p}. \end{aligned}$$

□

Conclusion

Adjacently, there is a field of cryptography which is concerned with quantum mechanics. *Quantum key distribution* involves quantum bits, as opposed to bits, to encode information. Similarly, *Shor's algorithm* uses quantum bits to find prime factors of an integer. It is currently inhibited by a number of problems with quantum computers, but were it to work, it could be used to break public-key crytopgraphy schemes, such as the ones discussed above.