Digital Signatures and Message Authentication Codes

Message Authentication

Conventional Encryption

The use of conventional (secret key) encryption algorithms such as the Data Encryption Standard, provides confidentiality as well as authentication of a message M since only the sender and the intended recipient can decrypt the ciphertext C with the secret key K as shown in Fig. 1.

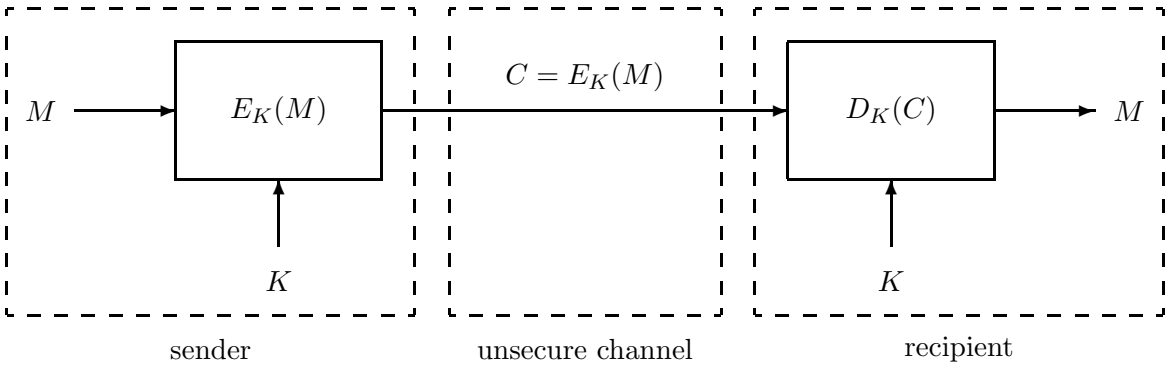

Figure 1: Conventional secret encryption.

Public-key Encryption

With public-key encryption like the RSA algorithm, user A, Alice, can provide confidentiality (see Fig. 2) using Bob's public-key K_B, authentication and signature (Fig. 3) using her own private key k_A, or both, that is, confidentiality, authentication and signature of a message M (Fig. 4) using Alice's private key k_A and Bob's public-key K_B.

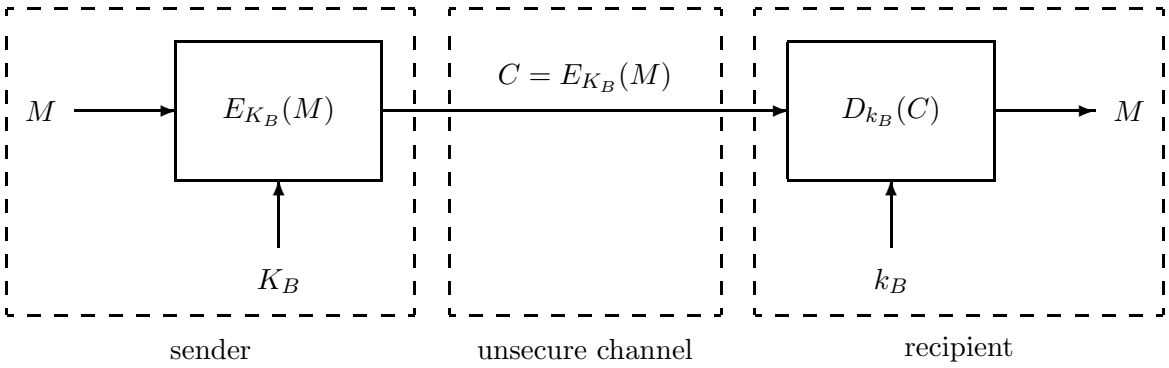

Figure 2: Public-key encryption (confidentiality).

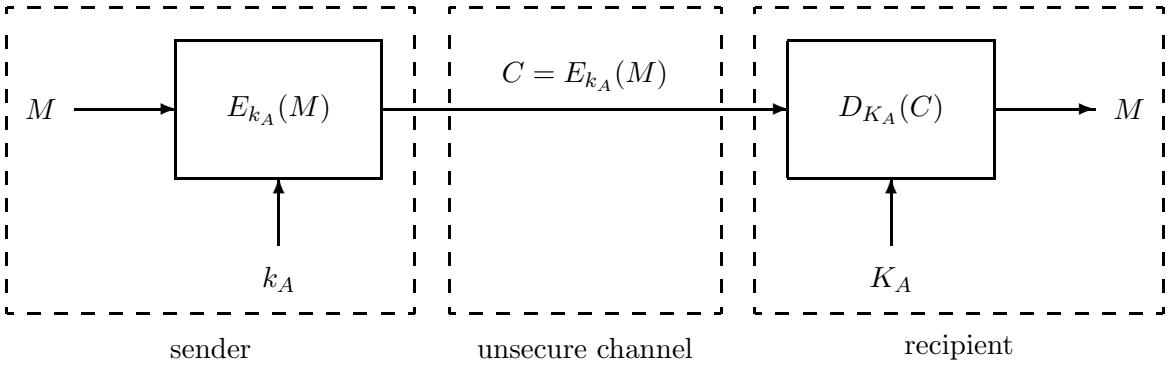

Figure 3: Public-key encryption (authenticity).

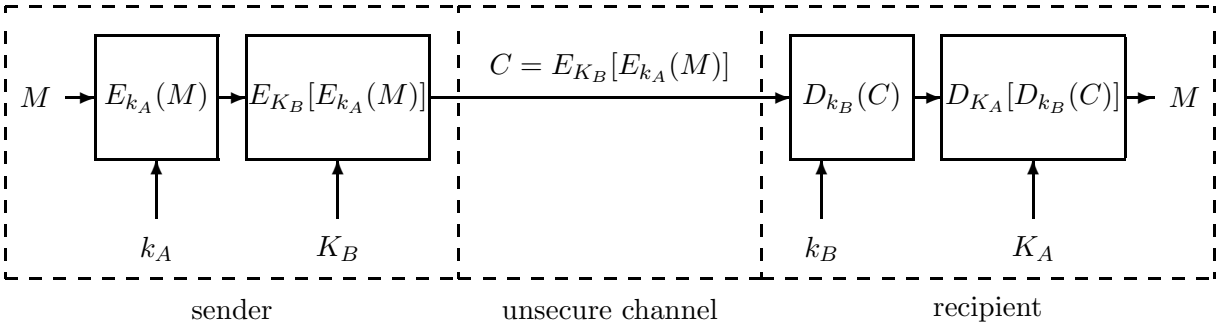

Figure 4: Public-key encryption (confidentiality, authenticity, and signature).

Hash Functions

A *hash function* $H(x)$ is a one-way function (many-to-one mapping function) which produces a fixed length vector from an input block x of arbitrary length. In other words, the function $H(x)$ can be applied to a block of data of any size and generate a fixed-length output.

For a given input x, it should be relatively easy to compute the hash function $h = H(x)$, but given the *hash value* h it must be computationally infeasible to find x, hence the term *one-way function*.

A good hash algorithm should provide *weak collison resistance*, that is for a given input x it should be computationally infeasible to find another block $y \neq x$ such that $H(y) = H(x)$.

A hash algorithm should also ensure *strong collison resistance*: it should be computationally infeasible to find any pair (x, y) such that $H(y) = H(x)$, this is to prevent a type of attack on hash function known as *birthday paradox attack*.

Simple Hash Functions

The simplest hash function that can be designed is probably a function where an input x is decomposed into L blocks of n bits. If the length of the input x is not exactly a multiple of n, then *stuffing bits* are appended at the end of the input vector to make its length of multiple of n. The hash function here consists simply into adding modulo 2 the n^{th} bit of each block:

$$\begin{aligned}
h_1 &= x_{1,1} \oplus x_{1,2} \oplus \ldots \oplus x_{1,l} \oplus \ldots \oplus x_{1,L} \\
h_2 &= x_{2,1} \oplus x_{2,2} \oplus \ldots \oplus x_{2,l} \oplus \ldots \oplus x_{2,L} \\
\vdots &= \vdots \oplus \vdots \oplus \ddots \oplus \vdots \oplus \ddots \oplus \vdots \\
h_i &= x_{i,1} \oplus x_{i,2} \oplus \ldots \oplus x_{i,l} \oplus \ldots \oplus x_{i,L} \\
\vdots &= \vdots \oplus \vdots \oplus \ddots \oplus \vdots \oplus \ddots \oplus \vdots \\
h_n &= x_{n,1} \oplus x_{n,2} \oplus \ldots \oplus x_{n,l} \oplus \ldots \oplus x_{n,L}
\end{aligned}$$

This simple hash function does provide a hash output $h = H(x)$ of length n as desired. However, with this simple hash algorithm it is very easy to find another input y which will produce the same hash output $H(y) = H(x)$, making it vulnerable to collison.

Birthday Paradox Attack

The birthday paradox can be stated as follows: given an integer random variable X uniformly distributed between 1 and n, that is: $p(x_i) = \frac{1}{n}$ for $i = 1, n$, and a selection of k instances (or outcomes) of that random variable X, waht is the probability $P(n, k)$ that there is at least one duplicate?

For instance, given an ELG-5373 class with k students (e.g. $k = 12$) what is the probability that two students have the same birthday, assuming that the $n = 365$ birthdays are uniformly distributed over the year, i.e. with a probability of $\frac{1}{365}$?

The probability of $Q(n, k)$ of having *no duplicates* is easier to determine. For a given k there are:

$$N_{(\text{no duplicates})} = n \times (n-1) \times \ldots \times (n-k+1) = \frac{n!}{(n-k)!}$$

ways to obtain no duplicates with k elements taken from the set of n elements.

If we now consider the total number of ways to pick k elements out of n without the condition of having no duplicates, then:

$$N_{(\text{with duplicates})} = \underbrace{n \times n \times \ldots \times n}_{k \text{ times}} = n^k$$

and the probability of no duplicates $Q(n, k)$ is given by:

$$Q(n,k) = \frac{N_{(\text{no duplicates})}}{N_{(\text{with duplicates})}} = \frac{\frac{n!}{(n-k)!}}{n^k} = \frac{n!}{(n-k)!n^k}$$

The probability $P(n, k)$ of having at least one duplicate is then:

$$\begin{aligned}
P(n,k) &= 1 - Q(n,k) \\
P(n,k) &= 1 - \frac{n!}{(n-k)!n^k} \\
P(n,k) &= 1 - \left[\frac{n \times (n-1) \times \ldots \times (n-k+1)}{n^k}\right] \\
P(n,k) &= 1 - \left[\left(\frac{n-1}{n}\right) \times \left(\frac{n-2}{n}\right) \times \ldots \times \left(\frac{n-k+1}{n}\right)\right] \\
P(n,k) &= 1 - \left[\left(1-\frac{1}{n}\right) \times \left(1-\frac{2}{n}\right) \times \ldots \times \left(1-\frac{k-1}{n}\right)\right]
\end{aligned}$$

However, for $x \geq 0$, we have that $(1-x) \leq e^{-x}$ as illustrated on Figure 5.

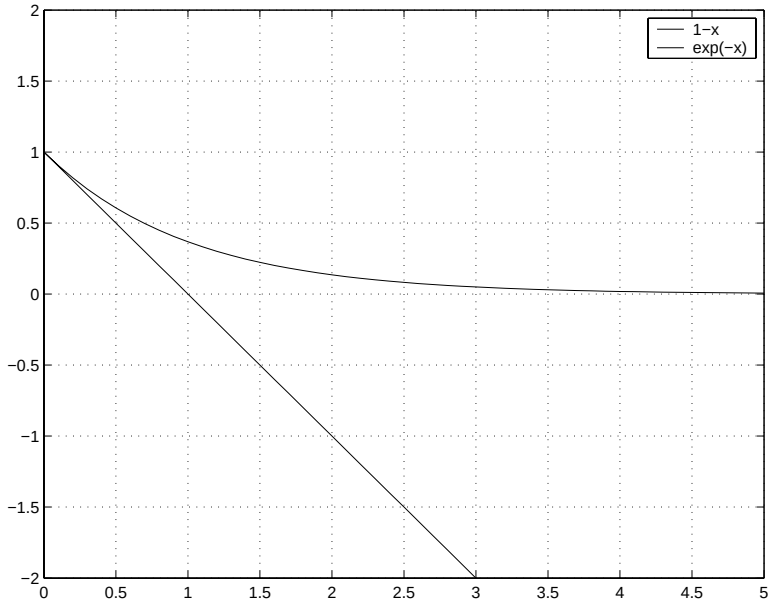

Figure 5: Inequality: $1 - x \leq e^{-x}$, for $x \geq 0$.

The probability $P(n,k)$ can be expressed as:

$$\begin{aligned}
P(n,k) &= 1 - \left[\left(1 - \frac{1}{n}\right) \times \left(1 - \frac{2}{n}\right) \times \ldots \times \left(1 - \frac{k-1}{n}\right)\right] \\
P(n,k) &\geq 1 - \left[\left(e^{-\frac{1}{n}}\right) \times \left(e^{-\frac{2}{n}}\right) \times \ldots \times \left(e^{-\frac{k-1}{n}}\right)\right] \\
P(n,k) &\geq 1 - e^{-\left[\frac{1}{n} + \frac{2}{n} + \ldots + \frac{k-1}{n}\right]} \\
P(n,k) &\geq 1 - e^{-\left[\frac{k \times (k-1)}{2n}\right]}
\end{aligned}$$

Example 1 *(Birthday paradox ($n = 365$)):*

For this first example, we want to determine the number k of people in a room such that we have a probability of 50% to find two persons having the same birthday. We assume here that there are $n = 365$ days per year. We want to determine the minimum number of persons k such that $P(n, k) = \frac{1}{2}$. We know that:

$$\begin{aligned} P(n,k) &= 1 - e^{-\left[\frac{k\times(k-1)}{2n}\right]} \\ \frac{1}{2} &= 1 - e^{-\left[\frac{k\times(k-1)}{2n}\right]} \\ -\frac{1}{2} &= -e^{-\left[\frac{k\times(k-1)}{2n}\right]} \\ \frac{1}{2} &= e^{-\left[\frac{k\times(k-1)}{2n}\right]} \end{aligned}$$

Taking the natural logarithm on both sides of the inequality leads to:

$$\begin{aligned} \ln\left(\frac{1}{2}\right) &= -\left[\frac{k\times(k-1)}{2n}\right] \\ \ln(2) &= \left[\frac{k\times(k-1)}{2n}\right] \\ 2n\ln(2) &= k\times(k-1) \leq k^2 \\ \pm\sqrt{2n\ln(2)} &= k \end{aligned}$$

Therefore, for $n = 365$ days, the minimum number of persons is $k \geq \sqrt{2\times 365\times\ln(2)}$ or $k \geq 22.4944$ as hown on Figure 6. With $k = 23$ persons in a room, the probability of finding two persons having the same birthday is greater than 50%. With, $k = 23$, this probability $P(n, k)$ is:

$$\begin{aligned} P(n,k) &\geq 1 - e^{-\left[\frac{k\times(k-1)}{2n}\right]} \\ P(365,23) &\geq 1 - e^{-\left(\frac{23\times 22}{730}\right)} \\ P(365,23) &\geq 1 - e^{-\left(\frac{23\times 22}{730}\right)} \\ P(365,23) &\geq 0.5000175218271 \end{aligned}$$

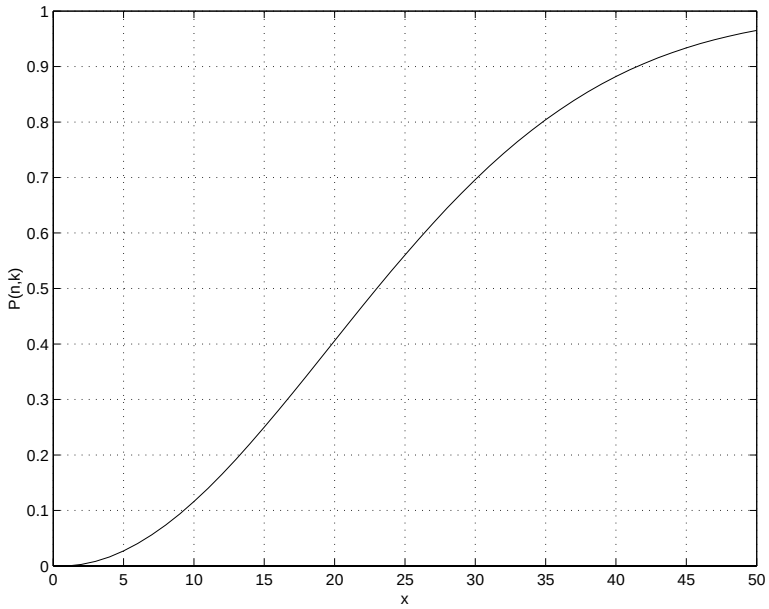

Figure 6: Birthday paradox: probability of finding two persons out of k having the same birthday (strong collision).

Example 2*(Birthday paradox ($n = 2^{128}$)):*

Consider now the probability of finding two different 128-bit vectors, x and y such that their hash values are equal, i.e., $H(x) = H(y)$. This is the problem of strong collision of any pair (x, y) into the same hash value.

The number of different vectors y to achieve equality in an exhaustive brute-force attack would be $k \leq \sqrt{2 \times 2^{128} \times \ln(2)}$, that is: $k = 2^{64} \times \sqrt{2 \times \ln(2)} = 1.1774 \times 2^{64}$ or equivalently $k = 2.1719 \times 10^{19}$.

Digital Signature Standard

Digital Signature Standard (DSS)

In 1991, the National Institute of Standard and Technology (NIST) in the United States presented the Digital Signature Standard (DSS). The DSS standard was modified and refined twice in 1993 and 1996. The block diagrams for the Digital Signature Standard message signing and signature verification are depicted on Figure 7 and Figure 8.

The DSS standard is based on the Digital Signature Algorithm (DSA) which itself uses the Secure Hash Algorithm (SHA-1) to produce a 160-bit hash value $h = H(M)$ of the message M.

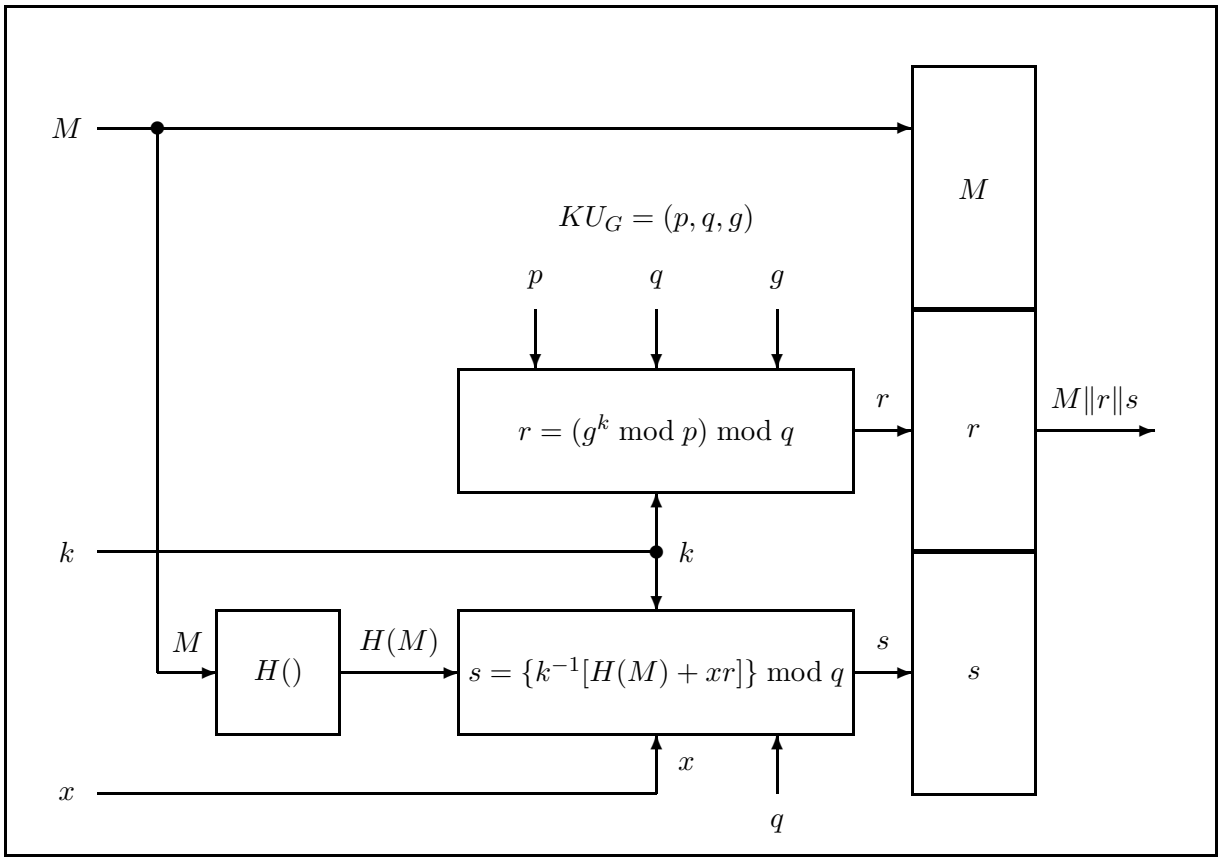

Figure 7: Digital Signature Standard (DSS): message signing.

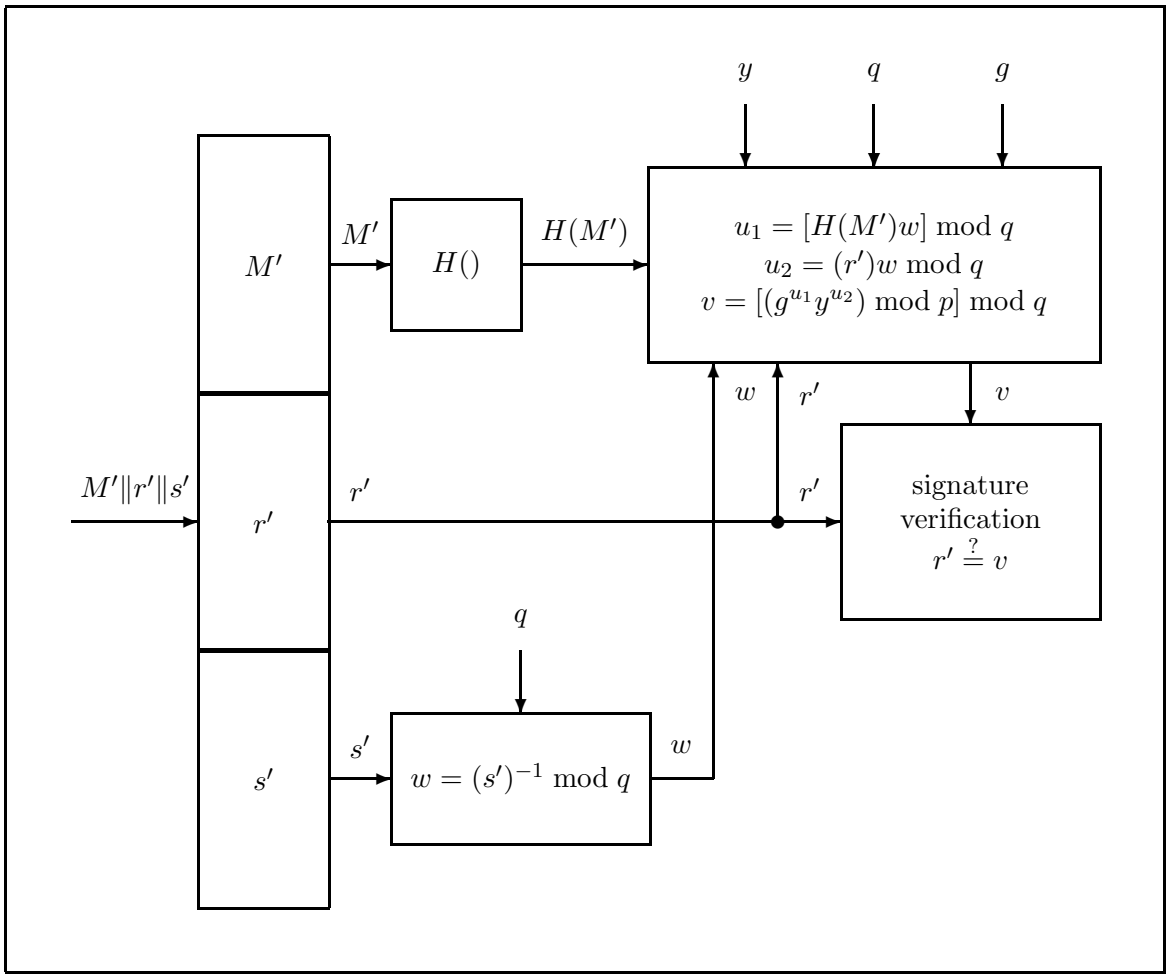

Figure 8: Digital Signature Standard (DSS): signature verification.

Digital Signature Algorithm (DSA)

The security of the Digital Signature Algorithm (DSA) is based on the computational complexity to compute discrete logarithms. The original message and signature are M and (r, s) respectively. $H(M)$ represents the hash of the message M using the Secure Hash Algorithm SHA-1.

Global Public Components: $KU_G = (p, q, g)$

The algorithm requires *global public components*, identified as the public key of a group KU_G, and consisting of global public components (p, q, g).

p is a prime number where $2^{L-1} < p < 2^L$, for $512 \leq L \leq 1024$ and where L is a multiple of 64, i.e. p is a prime number having a length L of 512 to 1024 bits.

A 160-bit prime number q (where $2^{159} < q < 2^{160}$) is chosen such that it is a divisor of $(p-1)$.

A number $g = h^{(p-1)/q} \bmod p$ is also chosen where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$.

User's Private and Public Key Pair: (KR_A, KU_A)

User A selects a pseudorandom integer x with $0 < x < q$. This is the private key KR_A. He then computes the corresponding public key KU_A as: $y = g^x \bmod p$. As we know, although it is easy to compute y by modular exponentiation it is very difficult to find the private key x from y, g, and p: this would require the solution of the discrete logarithm of y with base g modulo p: $x = \log_g(y) \bmod p$. This ensure the security of the DSA signature algorithm.

User's Per-Message Secret Number: k

The user also generates a pseudorandom integer k with $0 < k < q$.

Signature of the message

The signature of the message consists in two components: r and s:

$$\begin{aligned} r &= (g^k \bmod p) \bmod q \\ s &= \{k^{-1}[H(M) + xr]\} \bmod q \\ \text{Signature} &= (r, s) \end{aligned}$$

The signature is appended to the message, that is $M \| s \| r$, and sent over the non secure channel.

Message Signature Verification

The message and signature $M' \| r' \| s'$ are received, and potentially altered by an attacker in the unsecure network. The message signature verification is performed as follows:

$$\begin{aligned} w &= (s')^{-1} \bmod q \\ u_1 &= [H(M')w] \bmod q \\ u_2 &= (r')w \bmod q \\ v &= [(g^{u_1} y^{u_2}) \bmod p] \bmod q \end{aligned}$$

The signature verification test consists in determining whether $v \stackrel{?}{=} r'$. If the vectors v and r' match then the DSA digital signature is verified, if not it is rejected.

Digital Signature Algorithm Derivation

In this section, it is shown that if in the DSA signature verification the vectors v and r' are equal then the signature is valid. For this we have to show that $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$ equals the received signature component r. We will need intermediate results to prove that indeed $v = r$ if the signature is valid.

We first show that for any integer t:

$$g^t \bmod p = g^{(t \bmod q)} \bmod p$$

then, using this result, we show that, for non negative integers a and b:

$$g^{(a \bmod q + b \bmod q)} \bmod p = g^{[(a+b) \bmod q]} \bmod p$$

Then the relationship:

$$y^{(rw \bmod q)} \bmod p = g^{(xrw \bmod q)} \bmod p$$

is demonstrated. Then we show that, for $0 < k < q$, where q is a prime number,

$$[\{[H(M) + xr]w\}] \bmod q = k$$

leading to the proof that $v = r$ in the digital signature standard.

Lemma 1

For any integer t, if $g = h^{[(p-1)/q]} \bmod p$ then $g^t \bmod p = g^{(t \bmod q)} \bmod p$.

Proof:

The integer h is chosen such that $1 < h < (p-1)$ where p is a prime number, and hence h is relatively prime to p. By Fermat's theorem, we know that:

$$h^{p-1} \bmod p \quad = \quad 1$$

since p is prime. Consider the exponent nq of g modulo p, that is $g^{nq} \bmod p$, where n is an arbitrary positive integer. Using the principles of modular arithmetics, one obtains:

$$\begin{aligned}
g^{nq} \bmod p &= \left[h^{[(p-1)/q]} \bmod p\right]^{nq} \bmod p \\
g^{nq} \bmod p &= h^{[(p-1)/q]nq} \bmod p \\
g^{nq} \bmod p &= h^{[n(p-1)]} \bmod p \\
g^{nq} \bmod p &= \left[h^{[(p-1)]} \bmod p\right]^{n} \bmod p \\
g^{nq} \bmod p &= [1]^n \bmod p \\
g^{nq} \bmod p &= 1
\end{aligned}$$

Now express the arbitrary integer t as $t = nq + z$ where z is also a positive integer such that $0 < z < q$. Then $t \bmod q = z$ and:

$$\begin{aligned}
g^t \bmod p &= g^{nq+z} \bmod p \\
g^t \bmod p &= [g^{nq} \times g^z] \bmod p \\
g^t \bmod p &= [(g^{nq} \bmod p) \times (g^z \bmod p)] \bmod p \\
g^t \bmod p &= [(1) \times (g^z \bmod p)] \bmod p \\
g^t \bmod p &= g^z \bmod p
\end{aligned}$$

and, finally:

$$\boxed{g^t \bmod p = g^{(t \bmod q)} \bmod p}$$

QED

Lemma 2

If a and b are positive integers, then $g^{(a \bmod q + b \bmod q)} \bmod p$ is equal to $g^{[(a+b) \bmod q]} \bmod p$.

Proof:

From Lemma 1, we know that $g^t \bmod p = g^{(t \bmod q)} \bmod p$. If we set $t = a \bmod q + b \bmod q$ then:

$$\begin{aligned} g^t \bmod p &= g^{(t \bmod q)} \bmod p \\ g^{(a \bmod q + b \bmod q)} \bmod p &= g^{[(a \bmod q + b \bmod q) \bmod q]} \bmod p \\ g^{(a \bmod q + b \bmod q)} \bmod p &= g^{[(a+b) \bmod q]} \bmod p \end{aligned}$$

by the principle of modular arithmetics. Therefore, for any non negative integers a and b:

$$\boxed{g^{(a \bmod q + b \bmod q)} \bmod p = g^{[(a+b) \bmod q]} \bmod p}$$

QED

Lemma 3

Let the public key y be defined from the private key x as $y = g^x \bmod p$. Then

$$y^{(rw \bmod q)} \bmod p = g^{(xrw \bmod q)} \bmod p$$

Proof:

$$\begin{aligned} y^{(rw \bmod q)} \bmod p &= [g^x \bmod p]^{(rw \bmod q)} \bmod p \\ y^{(rw \bmod q)} \bmod p &= g^{x(rw \bmod q)} \bmod p \qquad \text{(modular arithmetics)} \\ y^{(rw \bmod q)} \bmod p &= g^{[x(rw \bmod q) \bmod q]} \bmod p \qquad \text{(Lemma 1)} \\ y^{(rw \bmod q)} \bmod p &= g^{(xrw \bmod q)} \bmod p \qquad \text{(modular arithmetics)} \end{aligned}$$

Then for the public and private key pair y and x:

$$\boxed{y^{(rw \bmod q)} \bmod p = g^{(xrw \bmod q)} \bmod p}$$

QED

Lemma 4

Let k be a randomly chosen integer such that $0 < k < q$ where q is a prime number. $H(M)$ is the hash value of the message M, x the private key. The signature components (r, s) are: $r = (g^k \bmod p) \bmod q$ and $s = \{k^{-1}[H(M) + xr]\} \bmod q$ and w is the multiplicative inverse of s modulo q, i.e. $w = s^{-1} \bmod q$. Then

$$\{[H(M) + xr]w\} \bmod q = k$$

Proof:

The signature component $s = \{k^{-1}[H(M) + xr]\} \bmod q$ by definition of the DSA. The integer k is chosen such that $0 < k < q$ and q is a prime number: this implies that any integer k has a unique multiplicative inverse k^{-1} modulo this prime number q. Thus $kk^{-1} \bmod q = 1$.

Now consider the expression $ks \bmod q$ (product of the signature component s with the random number k reduced modulo q):

$$\begin{aligned}
ks \bmod q &= k\left[\{k^{-1}[H(M) + xr]\} \bmod q\right] \bmod q \\
ks \bmod q &= \left[k\{k^{-1}[H(M) + xr]\}\right] \bmod q \\
ks \bmod q &= \left\{[(kk^{-1}) \bmod q][(H(M) + xr) \bmod q]\right\} \bmod q \\
ks \bmod q &= [(H(M) + xr) \bmod q] \bmod q
\end{aligned}$$

In the DSA algorithm w is the multiplicative inverse of s modulo q, i.e. $w = s^{-1} \bmod q$ and thus $ws \bmod q = 1$. Then

$$\begin{aligned}
\{[H(M) + xr]w\} \bmod q &= \{[(H(M) + xr) \bmod q](w \bmod q)\} \bmod q \\
\{[H(M) + xr]w\} \bmod q &= \{[ks \bmod q](w \bmod q)\} \bmod q \\
\{[H(M) + xr]w\} \bmod q &= [(ksw) \bmod q] \bmod q \\
\{[H(M) + xr]w\} \bmod q &= (ksw) \bmod q \\
\{[H(M) + xr]w\} \bmod q &= [(k \bmod q)(sw \bmod q)] \bmod q \\
\{[H(M) + xr]w\} \bmod q &= [(k \bmod q)(1)] \bmod q \\
\{[H(M) + xr]w\} \bmod q &= k \bmod q = k
\end{aligned}$$

Thus, for $0 < k < q$, where q is prime,

$$\boxed{\{[H(M) + xr]w\} \bmod q = k}$$

QED

Theorem (Digital Signature Standard)

Let the signature component $r = (g^k \bmod p) \bmod q$ and the signature verification vector $v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$ where $u_1 = [H(M')w] \bmod q$ and $u_2 = (r')w \bmod q$ with $w = (s')^{-1} \bmod q$. Then if the received version of the signed message $M'\|r'\|s' = M\|r\|s$, that is the original untampered signed message, then $v = r$ and the signature is validated:

$$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q = (g^k \bmod p) \bmod q = r$$

Proof:

$$\begin{aligned}
v &= [(g^{u_1} y^{u_2}) \bmod p] \bmod q \\
v &= \left[(g^{[H(M)w] \bmod q} y^{(rw) \bmod q}) \bmod p\right] \bmod q \\
v &= \left[(g^{[H(M)w] \bmod q} g^{(xrw) \bmod q}) \bmod p\right] \bmod q \qquad \text{(Lemma 3)} \\
v &= \left[(g^{[H(M)w] \bmod q + (xrw) \bmod q}) \bmod p\right] \bmod q \\
v &= \left[(g^{[H(M)w + (xrw)] \bmod q}) \bmod p\right] \bmod q \qquad \text{(Lemma 2)} \\
v &= \left[(g^{(H(M)+xr)w \bmod q}) \bmod p\right] \bmod q \\
v &= \left[(g^k) \bmod p\right] \bmod q \qquad \text{(Lemma 4)} \\
v &= r
\end{aligned}$$

Therefore, if the original and received signed messages $M\|r\|s$ and $M'\|r'\|s'$, then:

$$\boxed{v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q = (g^k \bmod p) \bmod q = r}$$

QED