Asymmetric Cryptographic Algorithms

Today the term "asymmetric cryptography" combines a large group of mechanisms, algorithms, protocols and notions used while developing information protection systems. In particular, this group includes such notions as one-way function, cryptographic protocol and proof with zero-knoledge. The practice requirements make it necessary to develop the asymmetric cryptography. At present the main motives for developing the asymmetric cryptography can be grouped as follows:

- requirements of the developing telecommunications networks of various applications with complex topology;
- requirements of ensuring the information security in Internet;
- requirements of various banking systems (including those using smart cards).

One of the main problems of the asymmetric cryptography is hash algorithms development. The importance of this class of algorithms is determined by a wide scope of their applications. In the long run, it is the hash algorithms that figure in message authentication and electronic digital signature.

The hash algorithm developed by the our company experts has a good combination of special features and speed.

Description of the Hash Algorithm

The hash function maps the binary string of an arbitrary length into binary vector of 256 bit length:

hash: $\mathrm{V}_2^* \to \mathrm{V}_2^{256}$.

Let us assume that it is necessary to calculate the value of the hash function for the argument $M=m_1\, m_2 \ldots m_T$, where $m_i \in \{0, 1\}$ for $i \in \overline{1,T}$.

Stage 1 (pad bits appending)

We calculate the new value for the text $M' = m'_1 m'_2 \ldots m'_T m'_{T+1} \ldots m'_{T+t(T)}$, where $m'_i \in \{0, 1\}$ for $i \in \overline{1, T + t(T)}$, according to the following rules:

$m'_i = m_i$ for $i \in \overline{1,T}$;

$m'_{T+1} = 1$;

$m'_{T+j} = 0$ for $j \in \overline{2, t(T)}$; at the same time we assume $t(T)$=1024 - ($(T+1)$ mod 1024) if $(T+1)$ mod 1024 $\neq 0$; $t(T)$=1025 in the opposite case. Here $(T+1)$ mod 1024 is designation of the remainder from division of the number $(T+1)$ by 1024.

Stage 2 (length block appending)

The text resulted from Stage 1 is added by the block M_{end} of length 1024 bit. The binary record of this block is representation of the number T – initial length of the message in bits (see Stage 1 of hashing).

Stage 3 (compression function)

At this stage proceeding from the input value – digital block of length 1024*k bit, where $k \geq 2$, the hash code of length 256 bit is calculated:

$\left(V_2^{1024}\right)^* \to V_2^{256}$.

Let this procedure be applied to the block $R = R_1\, R_2 \ldots R_k$, $R_i \in V_2^{1024}$ for $i \in \overline{1,k}$. The algorithm is iterative. On the basis of the iteration with number j ($j \in \overline{1, k-1}$) the value $h_j \in V_2^{256}$ is calculated according to the rule:

$h_j = \mathrm{H}\,(h_{j-1}, R_j)$,

where $H{:}V_{256}{\times}V_{1024}{\rightarrow}V_{256}$ - elementary compressing representation. In this case h_0 is assumed to be equal to a certain constant.

The result of the hash function is announced to be the value h_k.

Elementary compressing transformation

Let V_n be a set of binary vectors with length n (n is a positive integer). Depending on the context the vectors from V_n are considered either as elements of the linear vector space above the field of two elements GF(2), or as binary representation of a certain integer (or the greatest non-negative subtraction for the module 2^n).

Let $\oplus$ be an operation of the coordinate summation of the vectors from V_n above the field GF(2); + - an operation of summation for the module 2^{32}; $<< t$ – an operation of cyclic shift towards high-order bits by t digits (t – a positive integer).

The elementary compressing transformation H operates for a set V_{1280} – binary vectors of length 1280 and take values from the set V_{256} – binary vectors of length 256, while (for convenience of description) the binary vectors of length 1280 from the field of the definition of H will be considered as concatenation of two vectors of lengths 256 and 1024 accordingly:

$\mathrm{H}{:}V_{256}{\times}V_{1024}{\rightarrow}V_{256}$.

For calculating H(x,y) let us assume the first 256-digit argument x as concatenation of 8 vectors of length 32: x=(x_0,...,x_7), and the second 1024-digit argument y as concatenation of 32 vectors of length 32: y=(y_0,...,y_{31}), x_i,$y_j{\in}V_{32}$, $i{\in}\overline{0,7}$, $j{\in}\overline{0,31}$.

The process of calculating H(x,y) is iterative. 40 iterations are performed in total. At each iteration values of some of 8 32-digit variables change:

$A_0^{(i)}, A_1^{(i)}, A_2^{(i)}, A_3^{(i)}, B_0^{(i)}, B_1^{(i)}, B_2^{(i)}, B_3^{(i)}$;

The upper index specified in the brackets denotes the number of the iteration after which the value of the corresponding variable is considered.

The procedure of calculating H(x,y) proceeds from the initial state:

$A_0^{(0)}=x_0, A_1^{(0)}=x_1, A_2^{(0)}=x_2, A_3^{(0)}=x_3, B_0^{(0)}=x_4, B_1^{(0)}=x_5, B_2^{(0)}=x_6, B_3^{(0)}=x_7$;

At each iteration the current value of the variables $A_j^{(i)}$ и $B_j^{(i)}$ ($j{\in}\overline{0,3}$) is transformed under control of the second argument y of the function H(x,y). For this end 160 32-bit numbers are formed from the 1024-bit argument

$W_0^{(1)}; W_1^{(1)}; W_2^{(1)}; W_3^{(1)}; W_0^{(2)}; W_1^{(2)}; W_2^{(2)}; W_3^{(2)};... ; W_0^{(40)}; W_1^{(40)}; W_2^{(40)}; W_3^{(40)}$

according to the following rules:

these numbers represent the first 160 members of the recursive sequence $\{X_i\}_{i=1}^{\infty}$, where X_i - 32-bit numbers.

The law of recurrence is set by the following formula:

$X_{32+i} = (X_i << s_6\,)\oplus X_{i+9}$.

The value of the cyclic shift s_6 is a parameter of the scheme and defined hereinafter. The initial filling, that is the first 32 members of the recurring sequence, is y itself (second 1024-bit argument) of the function H(x,y).

At iteration i ($i{\in}\overline{1,40}$) the following calculations are made in consecutive order:

(1) $B_0 = (B_0^{(i-1)} <<s_0)) + f(A_0^{(i-1)}, A_1^{(i-1)}, B_3^{(i-1)}) + W_0^{(i)}+C_0$;
(2) $B_1 = (B_1^{(i-1)} <<s_1)) + f(A_2^{(i-1)}, A_3^{(i-1)}, B_0) + W_1^{(i)}+C_1$;
(3) $B_2 = (B_2^{(i-1)} <<s_2)) + f(A_0^{(i-1)}, A_2^{(i-1)}, B_1) + W_2^{(i)}+C_2$;
(4) $B_3 = (B_3^{(i-1)} <<s_3)) + f(A_1^{(i-1)}, A_3^{(i-1)}, B_2) + W_3^{(i)}+C_3$;
(5) $B_3^{(i)} = B_3 + (B_0 \oplus B_2)P$;
(6) $B_1^{(i)} = B_1 + ((B_2 \oplus B_3^{(i)})+C_4\,)P$;
(7) $B_0^{(i)} = B_0 + (B_1^{(i)} <<s_4\,)P$;
(8) $B_2^{(i)} = B_2 + (B_0^{(i)} <<s_5\,)P$;

When the iteration i ends, the values of the variables $A_j^{(i)}$ and $B_j^{(i)}$ ($j{\in}\overline{0,3}$) change places:

$A_0^{(i)} \leftrightarrow B_0^{(i)}$; $A_1^{(i)} \leftrightarrow B_1^{(i)}$; $A_2^{(i)} \leftrightarrow B_2^{(i)}$; $A_3^{(i)} \leftrightarrow B_3^{(i)}$.

The description of the functions f: $V_{32}\times V_{32}\times V_{32} \to V_{32}$ и P: $V_{32} \to V_{32}$ is given below.

The result of the function H action is declared to be the concatenation of the values of the variables A_j and B_j after the 40-th iteration:

$F(x,y)=(A_0^{(40)},A_1^{(40)},A_2^{(40)},A_3^{(40)},B_0^{(40)},B_1^{(40)},B_2^{(40)},B_3^{(40)})$.

The function f:$V_{32}\times V_{32}\times V_{32} \to V_{32}$ is set by the following equation:

$f(X_1, X_2, X_3) = X_1X_2 \oplus \overline{X_2}X_3$;

In this case the logic operators are considered to be performed coordinately ($\overline{X_2}$ - denotes coordinate negation of the vector X_2).

The function P: $V_{32} \to V_{32}$ replaces the low byte of the argument: for π substitution effective for the set of numbers $\overline{0,255}$.

$(u_3\ u_2\ u_1\ u_0)P = u_3\ u_2\ u_1\ \pi(u_0)$,

where u_3 ,u_2 ,u_1 ,u_0 – bytes of the argument.

At present data protection systems use widely protocols of electronic digital signature built on the basis of the integers factorization or taking discrete logarithms procedures. The strength of the specified systems is based on complexity of solution of the corresponding mathematical problems. However, while no effective algorithms have been found yet for solving such problems, mathematicians constantly reduce the level of their strength, by developing new algorithms. The latest reports say that they managed to develop the procedure of taking logs for numbers of length more than100 decimal digits. That's why there is no absolute assurance of such system reliability.

The promising direction in this field is applying in protocols operations in the group of points of the elliptic curve over the finite field. Proceeding from these ideas, the experts of the \ our company developed an original algorithm of public distribution of keys.

System of Public Distribution of Keys on Elliptic Curve

As a group, in which the system of public distribution of keys is realized, the mathematicians use the subgroup H of order Q of the group of points on the elliptic curve G, set by a range of decisions (x,y) of the equation $y^2=x^3-x$ for the module of the prime number P. Hereinafter the first coordinate of the decision of this equation will be called x-coordinate, and the second one – y-coordinate. If U – a point of the elliptic curve, then x-coordinate U will be denoted as $(U)_x$, and y-coordinate – as $(U)_y$.

Let P be a prime number of length 256 bit, a Q - a prime number of length 250 - 255 bit, with $Q < P$.

The numbers P, Q are selected so that $Q^t \neq 1 \pmod{P}$ be for all natural $t < 1000$.

Let us select the point $W=(u,v)$ of the group H, $v \neq 0$ fixed for all the net. Each user A selects a random integer X_A , $1 < X_A < Q$-1, and calculates the point W^{X_A}. The number X_A is a secret key of the user A, and x-coordinate $PK_A = (W^{X_A})_x$ of the point W^{X_A} is his public key for public distribution of keys and is published in the directory of public keys.

Procedure of Generation of Common Key of Communication

If the user A wants to send a ciphered message to the user B, then the user A must do the following:

- find in his directory of public keys the public key $PK_B=(W^{X_B})_x$ of the user B;
- calculate x-coordinate of the point $(W^{X_B})^{X_A}$, using his private key X_A and public key PK_B (number $(W^{X_AX_B})_x$ or its part forms the common key of communication of the users A and B).

The user A, using the calculated key, develops one-time key by using initial vector, cipher a message with this key and sends the ciphered message to the user B.

If the user *B* wants to decipher the ciphered message received from the user *A*, then the user *B* must do the following:

- find in his directory of open keys the public key PK_A=$(W^{X_A})_x$ of the user *A*;
- calculate *x*-coordinate of the point $(W^{X_A})^{X_B}$, using his secret key X_B and public key PK_A (the number $(W^{X_A X_B})_x$ is the common key of the users *A* and *B*).

The user *B*, using the calculated key, develops a one-time key by using initial vector, deciphers the ciphered incoming message with this key and gets a clear message from it.