22 Schepkina Str., Office 22
Moscow, RUSSIA
129090
www.lancrypto.com
Tel.: +7 / 095 / 974.76.60 /61/62/63
Fax: +7 / 095 / 974.76.59
E-mail: info@lancrypto.com

24.04.2005
7175-B

LAN Crypto Submission to the ECRYPT Stream Cipher Project.

B. Primitive specification and supporting documentation.

1. Description of the Primitive "Yamb".

The Cipher name is "Yamb" («Ямб» in Cyrillic)
Stream cipher Yamb is a *synchronous* encryption algorithm that allows keys of any length in the range 80–256 bits and allows initial vectors IV of any length in the range 32-128 bits.

Recommended values (bits) of key length are: 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240 or 256
Recommended values of IV length (bits) are the following: 32, 64, 96 or 128.

Structure of the algorithm Yamb.

Graphic scheme of the algorithm Yamb is on Picture 1.

Algorithm Yamb consists of the following three parts:

- Linear Feedback Shift Register (LFSR) L of length 15 over $GF(2^{32})$,
- Nonlinear function F_M,(32-bit input and 32-bit output), which uses dinamic memory M of 256 Bytes,
- Shift register R of length 16 over Z_2^{32} .

All parts of the algorithm Yamb exchange data by 32-bit blocks, i.e. we can think of data blocks as of binary vectors of length 32, or as of the elements of V_{32} – linear space over GF(2).

Register L is represented over $GF(2^{32})$ in its canonic presentation $F_2[x]/g(x)$,
where $g(x) = x^{32} + x^{27} + x^{24} + x^{20} + x^{19} + x^{17} + x^{16} + x^{12} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^3 + 1$ is a primitive polynomial over a field GF(2).

Elements of a field $GF(2^{32})$ are represented by binary polynomials of the form $A_0+A_1x^1+. . .+A_{31}x^{31}$ with degree less or equal to 31 and could be represented also by binary vectors $(A_0, A_1, \ldots, A_{31}) \in V_{32}$.

Characteristic polynomial of a register L over a field $GF(2^{32})$ is $f(y) = 1*y^{15}+1*y^8+x*1$,
where the coefficient of y^0 is $x = (0,1,0,\ldots0) \in GF(2^{32})$.

Function F_M maps V_{32} to V_{32}.

Shift register R contains 16 cells with the elements of a ring Z_2^{32} represented by the integers $A = A_0+A_12^1+\ldots+A_{31}2^{31}$ from the interval $[0, 2^{32}-1]$, which also could be interpreted as binary vectors $(A_0, A_1, \ldots, A_{31}) \in V_{32}$.
Contents of the registers L and R and of memory M are formed on key schedule step of the algorithm.

Computations:
Compute the next element of LSFR L by the polynomial f(y) over GF(2^{32}).
This element put in the next cell of a register L and got as an input to the function F_M.
Output of the function F_M XOR-ed with this input and sent to the register R as an input.
This input of the register R added mod 2^{32} to output of the register R.
The last sum is a 32-bit output vector of the algorithm Yamb (enciphering Γ) .

Function F_M.
Input vector (Y_0, Y_1, …, Y_{31}) of the function F_M divided by four consequent bytes:
a = (Y_0, Y_1, …, Y_7), b = (Y_8, Y_9, …, Y_{15}), c = (Y_{16}, Y_{17},…, Y_{23}) and d = (Y_{24}, Y_{25},…, Y_{31}).
With these bytes (a, b, c, d) we make a dozen (12) operations of the form H(U,T), in accordance with the following scheme on Picture 2.

Operation H(U,T) is defined as follows: take from memory M a byte with the index T, (we denote this byte as M(T)), and on its place in memory M we put a sum

$$(U+M(T)) \bmod 256$$

modulo 256 of the bytes M(T) and U, then as a result of the operation we take byte M(T).

Byte T = (T_0, …, T_7) defines a cell with number $T_0+T_1 2^1+\ldots+T_7 2^7$.

As a sum modulo 256 of bytes X = (X_0, …, X_7) and Y = (Y_0, …, Y_7) we mean, as usually, a byte Z = (Z_0, …, Z_7), such that

$$W_0+W_1 2^1+W_2 2^2+ \ldots +W_7 2^7 \equiv U_0+U_1 2^1+ \ldots +U_7 2^7+V_0+V_1 2^1+ \ldots + V_7 2^7 \mod 256.$$

A result of computation of a function F_M consists of four bytes
A = (A_0, A_1, …, A_7), B = (B_0, B_1, …, B_7), C = (C_0, C_1, … C_7) and D = (D_0, D_1, … D_7),
which are combined in a 32-bit vector

(A_0, A_1, …, A_7, B_0, B_1, …, B_7, C_0, C_1, …, C_7, D_0, D_1, …, D_7).

If we define by M(T) a byte of memory M that is changed in operation H(U,T), then we describe computation of F_M in the following way:

b := b ⊕ M(a),	M(a) := M(a) + d mod 256
c := c ⊕ M(d),	M(d) := M(d) + b mod 256
a := a ⊕ M(b),	M(b) := M(b) + c mod 256
d := d ⊕ M(c),	M(c) := M(c) + a mod 256
c := c ⊕ M(a),	M(a) := M(a) + d mod 256
b := b ⊕ M(d),	M(d) := M(d) + c mod 256
a := a ⊕ M(c),	M(c) := M(c) + b mod 256
d := d ⊕ M(b),	M(b) := M(b) + a mod 256
b := b ⊕ M(a),	M(a) := M(a) + d mod 256
c := c ⊕ M(d),	M(d) := M(d) + b mod 256
a := a ⊕ M(b),	M(b) := M(b) + c mod 256
d := d ⊕ M(c),	M(c) := M(c) + a mod 256

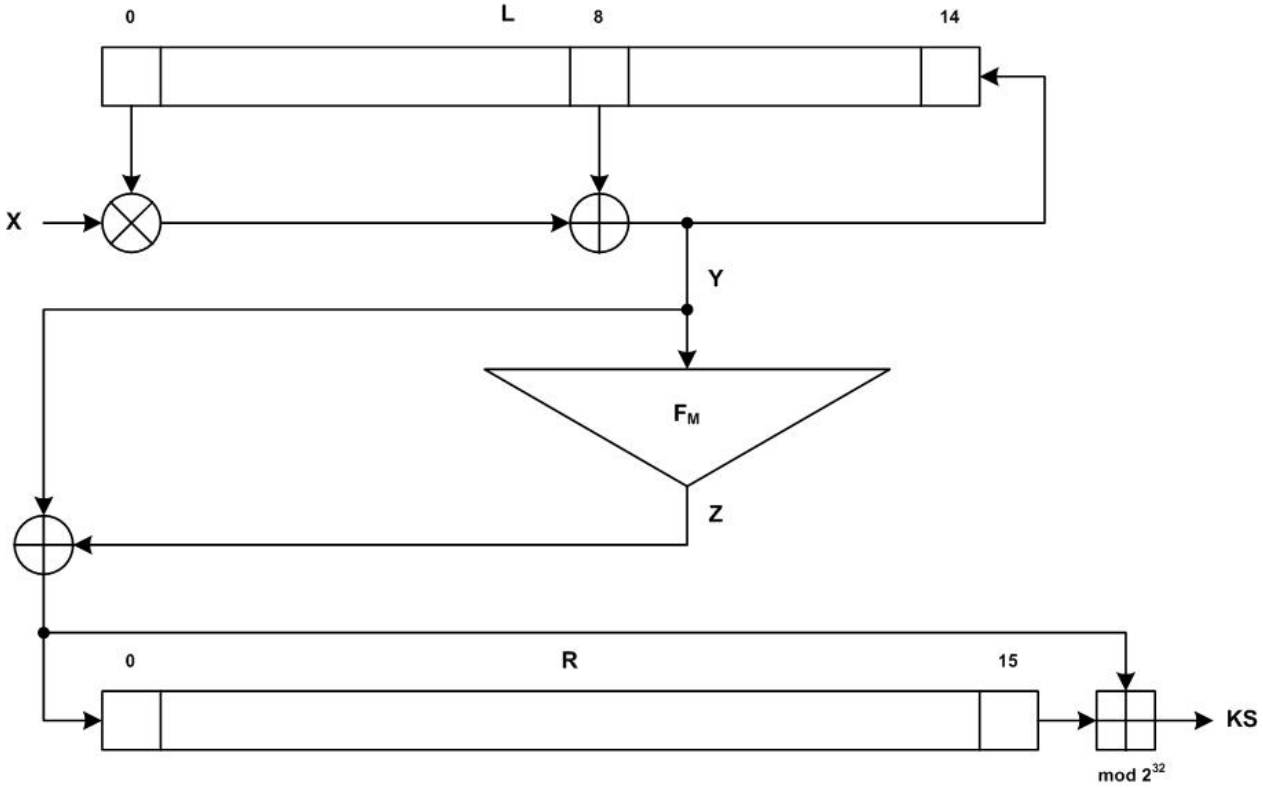
where symbol + is for arithmetic addition and ⊕ is for XOR.

Key schedule.
Initial state of LSFR L.
LSFR L consists of 15 cells with 32-bit vector each.
To fulfill register L we need 480 bits. First 256 of these bits are bits of key.
If key length is less than 256 bits, then we repeat key until the resulting length of this bit sequence will be 256. All additional bits of the last copy of a key are deleted.
Immediately right to these 256 bits we write bits of IV.
All the other bits of register L fulfilled by repeating copies of sequence of the following 72 bits

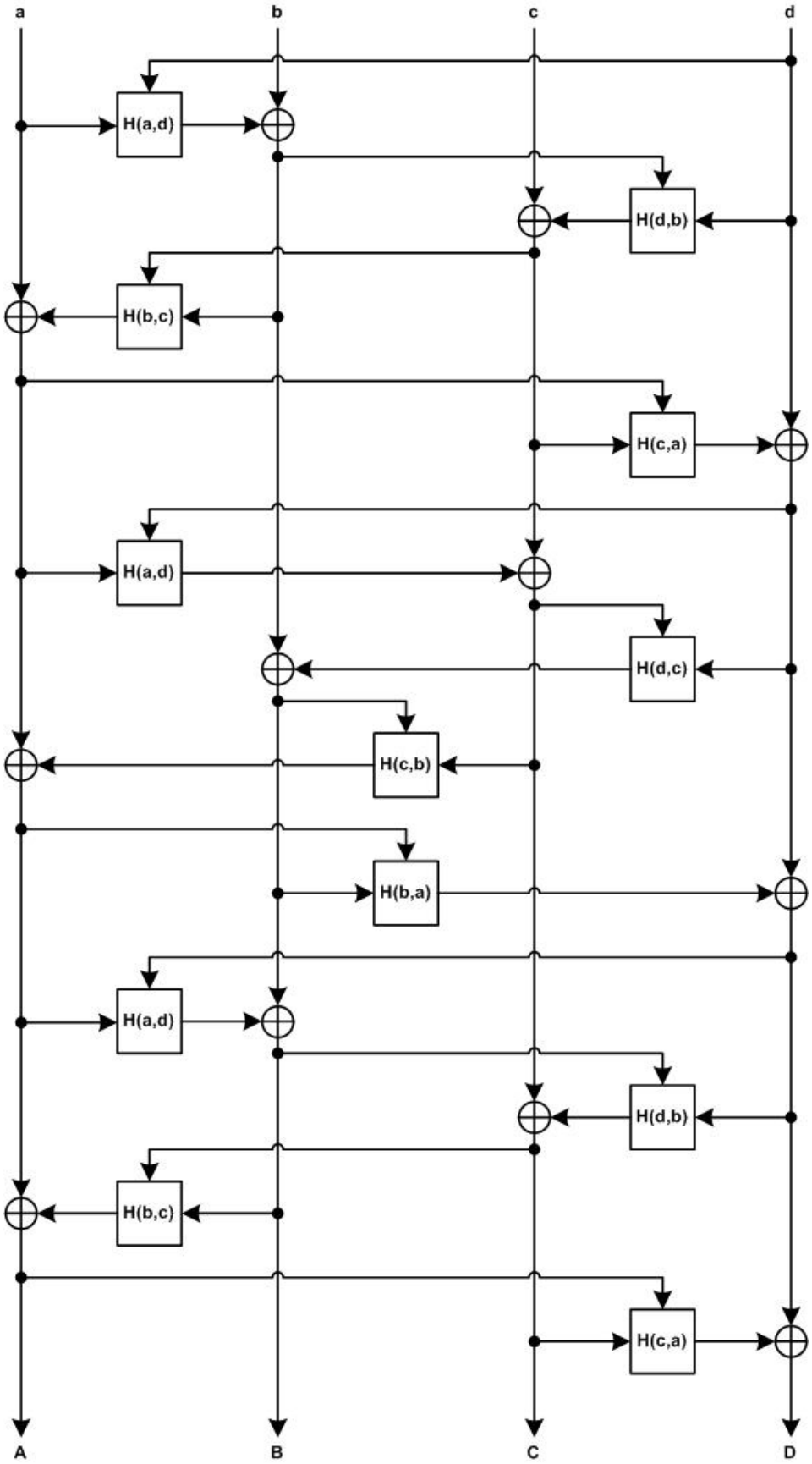
(0011 0010 1000 0010 0111 0010 1100 0010 0100 1110 1001 1110 0000 1110 0010 1110 1111 0110),

which is an ASCI code of the submitter's name LANCrypto.

Initial state of memory M is given in Table 1.
Initial state of register R may be random: in key schedule process register R will get new value.



Picture 1



Picture 2.

Table 1.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	85	2D	43	2F	A6	90	F8	1B	A9	B4	1C	58	E8	A5	D7	**56**
10	6B	03	38	67	3D	B1	7B	0B	F2	CB	29	FC	53	75	05	**CA**
20	0E	AE	D1	9C	BC	B0	DF	62	CF	3A	FE	DC	20	83	88	**68**
30	41	24	45	01	91	F0	A0	F6	65	02	B5	BE	0F	E5	13	**D4**
40	BF	A4	86	4A	63	82	4B	AC	9E	E2	7F	50	6C	EC	31	**44**
50	09	94	9A	40	06	C3	37	F4	2A	57	7C	25	99	FA	21	**3B**
60	EE	54	3C	22	B8	EB	51	8C	87	66	10	27	6D	AA	CE	**39**
70	E0	BD	8B	9B	69	B6	E7	36	AF	DE	34	93	9F	A2	60	**14**
80	7D	A7	8D	7E	76	48	72	74	23	CD	73	D9	33	D6	B2	**78**
90	9D	3F	32	8E	ED	5B	2B	4F	D3	E9	1E	4C	16	4E	B3	**C5**
A0	D8	F3	2E	26	28	8A	12	64	FB	A3	FF	AD	E1	B7	1A	**D0**
B0	F1	BA	7A	A1	00	D2	E4	C6	C0	30	81	52	92	46	61	**C1**
C0	95	1F	2C	C2	4D	42	49	07	5A	FD	0C	70	CC	84	F9	**D5**
D0	5E	18	B9	5D	C9	5C	C4	1D	6E	35	59	DB	15	79	DD	**E6**
E0	DA	A8	89	80	98	5F	EF	96	19	F7	C7	3E	47	0D	71	**EA**
F0	**04**	**BB**	**55**	**77**	**C8**	**0A**	**17**	**97**	**AB**	**8F**	**11**	**08**	**E3**	**6F**	**F5**	**6A**

Key schedule consists of the following four steps.

Step1. *Mixing contents of M and L.*
With initial states of M and L compute 225 output blocks by 32 bits each.
We do not use this output vector, but after this we get new contents of M and L.

Step 2. *Modification of M.*
Memory M consists of 64 vectors M = (M_0, M_1, …, M_{63}) of 32 bits each,
where M_i = (m_{32i}, m_{32i+1}, …, m_{32i+31}).
Compute the first 32-bit output vector, and add it to M_0 mod 2^{32}.
With this new memory M compute the next output vector and add it mod 2^{32} to M_1.
The same procedure continued up to modification of all 64 vectors of memory {M_i}.

Step 3. *Change of LFSR content.*
Compute and remember all of 15 output 32-bit vectors.
Put these output vectors into corresponding cells of the register L : first vector – into cell 0, second vector – into cell 2, etc.

Step 4. *Clearing of register R.*
With current contents of register L, memory M, and register R compute 16 output vectors.
We do not use these output vectors, but we use new contents of register L, memory M and register R to continue computations.

This key schedule scheme takes as much time as takes computation of 320 output vectors.
After this key schedule algorithm is ready for work.

2. There are no hidden weaknesses inserted by the designers.

3. Analysis of the algorithm Yamb with respect to all standard cryptanalytic attacks does not show any attack more effective than brute force method of key recovery.
There are no weak keys found by inventors.

4. The primitive requires relatively small amount of memory for its implementation and work (less than 0.5 K Byte in total).
The primitive could be implemented effectively in software as well, as in hardware:

- The algorithm has very simple and clear architecture,
- Elementary blocks are known and widely investigated in cryptography,
- All basic operations work with bytes,

5. The design rationale of the algorithm Yamb is the following.

The initial LFSR is a well known block for ciphers design. The one used in Yamb has a primitive polynomial over $GF(2^{32})$.
Its coefficients over a field $GF(2^{32})$ make this polynomial very convenient for software and hardware implementations and, on the other hand, guarantee period of the LFSR output sequence as long as $2^{480} - 1$, and very good statistical properties of this output.
This block is also very convenient for key schedule.

Function F_M, that is used to compose output of the cipher from the initial LFSR output, is based upon use of a dynamic memory to change so called S-box from round to round.
All the transformations upon input vectors work with chains of bytes, and each byte is used only once to modify memory and to indicate a memory cell to be used as output.
Each output byte depends of all the input bytes and this dependence is a polynomial of a high degree.
XORing input and output vectors of the F_M makes hidden addresses in memory that were modified on the last step.
The final shift register R protects from the attacks by trying values in memory. In the 32 clocks of the algorithm, in which input of R goes through the register, more that half of the memory M cells will change their contents.

6. Primitive setup is practically equivalent to key schedule that requires as many clocks as generating of 320 output blocks.
Encryption/decryption requires 21 cycles per byte or 84 cycles per block. The most efficient way is to generate of 16 blocks or 64 bytes simultaneously.

7. Basic techniques for implementers to avoid implementation weaknesses are the following:
- In implementing initial LFSR use masks to compute the next values of the register cells to prevent side channel analysis methods,
- Initial LFSR and final register R implement with 16 cells each to hide power supply difference in different moments.

Backup point of contact is Anatoly N. Lebedev, President of LAN Crypto,

22 Schepkina Str., Office 22,
Moscow, RUSSIA, 129090,
Tel.: +7 / 095 / 974.76.60 /61/62/63,
Fax: +7 / 095 / 974.76.59
E-mail: lan@lancrypto.com