JEROBOAM

Hervé Chabanne¹ and Emmanuel Michon²

¹ SAGEM SA, Unité de Recherche et Développement
Systèmes et Terminaux Sécurisés, France
chabanne@urd32.sagem.fr

² Ecole Polytechnique, Palaiseau, France
michon@www.enst.fr***

Abstract. We introduce a new fast stream cipher, JEROBOAM, working with a key of 128 or 248 bits. JEROBOAM was designed to work with eight internal 32-bit registers called multiply-with-carry generators (*mwc*). These register are very easy to implement in software and produce a sequence of excellent statistical quality. Per contra, one *mwc* is easily cracked by a lattice reduction algorithm. Hence, we are lead to interpose a nonlinear filter between these weak registers and the pseudo-random output.

1 Introduction

The cipher JEROBOAM is designed to work efficiently on 16-bit microprocessors. The key is 128- or 248-bit long, which is quite comfortable; after a short setup equivalent in time to the encryption of a 42-byte message, JEROBOAM produces a pseudo-random stream one can use as a symmetric cipher to XOR a cleartext of any length.

JEROBOAM was designed with IDEA[8,11] as a model. It lies on a classical scheme and can be seen as a nonlinear combination of irregularly clocked pseudo-random generators. Current technical requirements bind us to use only operations directly available on all microprocessors : we use *mwc* as random generators and the nonlinear filter is obtained by a now classical alternance of $+$ and $\oplus$ [2].

The *mwc*, multiply-with-carry generators, are a new primitive in cryptography due to G.Marsaglia[5]¹. They allow fast computations in different prime finite fields ; their description and the way they can be cryptanalysed can be found in Sect. 2.

The complete description of JEROBOAM is given in Sect. 3. It is completed by a slow C implementation and test values in Sect. 7.

In Sect. 4, we discuss the statistical evaluation of the output stream as a pseudo-random sequence. We study in Sect. 5 the speed of JEROBOAM and give a microprocessor-independent evaluation.

*** current address: Ecole Nationale Supérieure des Télécommunications, Paris, France.

This work was completed during a terminal training period in SAGEM SA. The reference *Etude cryptologique du chiffreur JEROBOAM*[4] is the complete, full-documented version of this article.

¹ thanks to [10] for its pointer on the package DIEHARD due to G.Marsaglia.

2 Specification

2.1 Multiply With Carry (mwc) Generators[5]

Definition 1. Let $a, b \geq 1$, $0 \leq c_0 < a$, $0 \leq x_0 < b$. We call **mwc** of multiplier a and base b the sequence of integer pairs $(c_n, x_n)_{n \in \mathbb{N}}$ determined by the following relation:

$$\begin{array}{l} c_{n+1} \text{ is the quotient} \\ x_{n+1} \text{ is the remainder} \end{array} \quad \text{in the euclidean division of } ax_n + c_n \text{ by } b. \quad (1)$$

c_n is called the carry, and we denote with ${}^c x_n$ the pair (c_n, x_n) .

The division is just a right shift if we let b be a power of 2. In JEROBOAM, $b = 2^{16}$ and we let the carry c_n be the MSBs and x_n the LSBs of a 32-bit word $\mathbf{w}$. This way, (1) becomes in ready-to-code form

$$\mathbf{w} = a(\mathbf{w} \& 0\text{xffff}) + \mathbf{w} \gg 16$$

where $\&$ is the logical AND and $\gg$ the right shift. We return $x_n = \mathbf{w} \& 0\text{xffff}$.

We can state the following result from [5]:

Proposition 2. Let us consider the set $S = \{{}^c x \mid 0 \leq c < a, 0 \leq x < b\}$ and the integer $m = ab - 1$, called the modulus — remark that $\#S = m + 1$. The directed graph of the function f which transforms ${}^c x_n$ into ${}^{c_{n+1}} x_{n+1}$ in S consists of several orbits.

The two points ${}^0 0$ and ${}^{a-1} b - 1$ are fixed points for f . If an orbit has k points, k is the order of b in a group $(\mathbb{Z}/m'\mathbb{Z}, \times)$, where m' is a divisor of m .

If we choose m to be a safe prime, ie both m and $(m - 1)/2$ are prime, we get two non trivial orbits for this graph.

The following result shows how to switch from one orbit to another.

Proposition 3. The symmetry $g : S \rightarrow S$ which transforms ${}^c x$ into ${}^{a-1-c} b - 1 - x$ commutes to f .

Proof. Let ${}^{c'} x' = f({}^c x)$ defined by the euclidean division by b

$$bc' + x' = ax + c, \text{ with } 0 \leq x' \leq b - 1$$

then

$$\begin{aligned} ab - 1 - (bc' + x') &= ab - 1 - ax - c \\ b(a - 1 - c') + (b - 1 - x') &= a(b - 1 - x) + (a - 1 - c) \end{aligned}$$

with $0 \leq b - 1 - x' \leq b - 1$, which means $g \circ f \circ g = f$, so $f \circ g = g \circ f$. $\square$

This way, we can switch from a one-point orbit to the other, and, more interesting, from a $(m - 1)/2$ point orbit to the other.

Choice of the Multipliers Considering only safe prime modules $m = a2^{16} - 1$ ($0 < a < 2^{16}$) leave us the choice between 392 values.

We would appreciate to use every bit in the 32-bit word $\mathbf{w}$, so we impose the condition $2^{15} < a$. There are still 171 possibilities left.

To explain our final choice, let us consider the following result, which establishes a strong link between $\mathbf{mwc}$ and the well-known Lehmer generator $X_{n+1} = aX_n \bmod m$:

Proposition 4. *Let $X_0 = ax_0 + c_0$ if $ax_0 + c_0 < m$. In this case the multiplicative congruential generator*

$$X_{n+1} = aX_n \bmod m$$

verifies for all $n \in \mathbb{N}$:

$$c_n = X_n \bmod a, \quad x_n = \left\lfloor \frac{X_n}{a} \right\rfloor. \quad (2)$$

The initial conditions not satisfying $ax_0 + c_0 < m$ consist of the unique pair ${}^{c_0}x_0 = {}^{a-1}b - 1$, which is a fixed point for f .

Proof. (2) is true for $n = 0$. Suppose it is true for n . Let q, r be the quotient and the remainder of the euclidean division of X_n by b , we get

$$X_n = bq + r \text{ avec } 0 \leq r < b.$$

Multiply by a , it becomes

$$aX_n = abq + ar \text{ avec } 0 \leq ar < ab,$$

we obtain this way the euclidean division of aX_n by ab . One can also write:

$$aX_n = (ab - 1)q + ar + q$$

is this the euclidean division of aX_n by $ab - 1$? Let us check that $0 \leq ar + q < ab - 1$:

- if $0 \leq r \leq b - 2$, given $0 \leq q \leq a - 1$, one gets $ar + q = a(b - 2) + a - 1 = ab - 1 - a < ab - 1$.
- if $r = b - 1$, we know then $0 \leq q \leq a - 2$, il vient $ar + q = a(b - 1) + a - 2 = ab - 2 < ab - 1$.

This way, $X_{n+1} = ar + q$: it is the relation stated at the rank $n + 1$. □

D.E.Knuth defines the spectral test in [3]. This test is an evaluation of the quality of the geometric repartition of the successive t -uples $(X_n, X_{n+1}, \dots, X_{n+t-1})$ of a Lehmer generator ; it is clear that two generators of same module m can have very different behavior with different multipliers a .

G.S.Fishman and L.R.Moore[6,9] use this test to determine the best multipliers for given modules $m = 2^{31} - 1$, 2^{32} and 2^{48} .

We choose the biggest eight a satisfying the conditions: $m = a^{2^{16}} - 1$ safe prime, $2^{15} < a < 2^{16}$, and $\mu_3 > 0.02$ (the figure of merit μ_t and ν_t are defined in [3], p.101).

a	$a(\text{hex})$	ν_2^2	ν_3^2	ν_4^2	ν_5^2	ν_6^2	μ_2	μ_3	μ_4	μ_5	μ_6
36594	8ef2	1339120837	54273	31023	3610	1148	1.75	0.0221	1.98	1.72	3.26
36804	8fc4	1354534417	65869	29375	3287	771	1.76	0.0294	1.77	1.35	0.98
37959	9447	1440885682	54529	29468	2782	667	1.82	0.0214	1.72	0.86	0.62
38568	96a8	1487490625	75165	30978	2259	1017	1.85	0.0342	1.87	0.51	2.15
40995	a023	1680590026	77923	25142	4287	1118	1.97	0.0339	1.16	2.36	2.69
42153	a4a9	1776875410	81906	53845	1827	797	2.02	0.0355	5.18	0.27	0.95
42903	a1=a797	1840667410	65865	40036	1408	1258	2.06	0.0252	2.81	0.14	3.66
43995	a3=abdb	1935560026	58906	54315	4903	930	2.11	0.0208	5.05	3.07	1.44
47529	a5=b9a9	2259005842	68806	40203	690	686	2.28	0.0243	2.56	0.02	0.54
51813	a7=ca65	2684586970	73230	39339	5164	1055	2.48	0.0244	2.25	2.97	1.79
53130	a6=cf8a	2822796901	75169	41931	4691	889	2.55	0.0248	2.49	2.28	1.04
54564	a4=d524	2977230097	83097	39742	4283	1087	2.62	0.0281	2.18	1.77	1.86
57225	a2=df89	3274700626	74934	53947	7793	1223	2.74	0.0229	3.83	7.52	2.52
61914	a0=f1da	3833343397	88869	71443	4014	750	2.97	0.0273	6.21	1.32	0.54

Remark that the whole multiplicative congruential generator is implied in the spectral test ; the bits we use in JEROBOAM are, in fact, the most significant ones.

2.2 Cryptanalysis of a mwc

We know that the sequence produced by a multiplicative congruential generator is predictable. More precisely, A.M.Frieze, J.Hastad, R.Kannan, J.C.Lagarias and A.Shamir[1] prove the following result:

Proposition 5. *Let us consider k successive terms in the sequence $X_{n+1} = aX_n \bmod m$. If we specify at least*

$$s = 1 + \log_2 \Delta_k + \frac{1}{2} \log_2 k$$

high-order bits on each term, it is possible to recover the whole sequence $(X_n)_{n \in \mathbb{N}}$ in polynomial time. Δ_k is defined below.

Proof. Consider the matrix²

$$A = \begin{pmatrix} m & 0 & 0 & \cdots & 0 \\ a & -1 & 0 & \cdots & 0 \\ a^2 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a^{k-1} & 0 & 0 & \cdots & -1 \end{pmatrix},$$

² A detailed proof of a more complete result can be found in [1].

we want to solve the system of modular equations $AX \equiv C \pmod{m}$ where $X = (x_1, x_2, \dots, x_k)$ and $C = (0, 0, \dots, 0)$ are two column vectors.

Let us consider the lattice formed by the rows of the matrix A , and reduce it by the LLL algorithm. This way, we get a matrix $B = PA$ with a “small” norm Δ_k , defined as the biggest euclidean norm of the rows of B .

The system can then be written $BX \equiv C' \pmod{m}$. If we choose the components of C' in the integer interval $[-m/2, m/2[$, and if we know in advance that $\|X\|$ is small enough, ie, $\|BX\| < m/2$, we loose the modular aspect and we are lead to solve a trivial linear system $BX = C'$ in $\mathbb{Z}$.

Knowing some high order bits, X^h , of X allows us to change the unknown and substitute X by the small unknown vector $X - X^h$, which leads us to the situation above. $\square$

A C implementation of this proposition shows that s is roughly n/k for big m , and that a Lehmer generator can be “cracked” in a few seconds by the observation of a few MSBs. For more precise results, see [4].

3 Description of JEROBOAM

The heart of JEROBOAM consists of eight 32-bit `mwc` registers, a FIFO queue of two 16-bit words and a particular 16-bit word:

`mwc0, mwc1, mwc2, mwc3, mwc4, mwc5, mwc6, mwc7, queue1, queue2, lea` for leader.

3.1 Setup: Key Insertion

One can choose between a 248-bit key and a 128-bit key.

248-bit key The key is given by eight 32-bit words `key0, key1, key2, key3, key4, key5, key6, key7`. The 32th bit of each word must be 0, and none of these words can be 0. The initial value of `mwci` is set to `keyi`.

`queue1` and `queue2` receive any values (we chose `0xda37, 0xc07f`). `lea` is initially the LSBs of `mwc0`.

Make 21 cycles³ of the algorithm below, and prepare to encrypt.

128-bit key The key is given by eight 16-bit words `key0, key1, key2, key3, key4, key5, key6, key7`. The i th `mwc` receives the 32-bit word $(i+1)2^{16} + \text{key}_i$. One can choose any of the 2^{128} possible key, even the zero one! The following is identical.

³ This is the first rank k beyond which the probability $(k+1)/2^k$ of outputting one of the queue original setup values is less than the uniform probability 2^{-16} .

3.2 Elementary Cycle

1. Consider the bits of `lea` :

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
swi2	sup0	lea0	chop	ini1	sup2	five	fifo	ini2	lea1	ini0	sup1	lea2	swi0	cplt	swi1

2. i_0 is $4ini2 + 2ini1 + ini0$.
3. If `chop` is 1, then

$$cmb = mwc_{i_0} \oplus mwc_{i_0+1 \bmod 8} + mwc_{i_0+2 \bmod 8} \oplus mwc_{i_0+3 \bmod 8}$$

else

$$cmb = mwc_{i_0} + mwc_{i_0+1 \bmod 8} \oplus mwc_{i_0+2 \bmod 8} + mwc_{i_0+3 \bmod 8}.$$

+ denotes the modular addition in $\mathbb{Z}/2^{16}\mathbb{Z}$, $\oplus$ the bit-to-bit XOR. The evaluation of these two non commutative, non associative operations is done from left to right.

4. If `five` is 1, add $mwc_{i_0+4 \bmod 8}$ to `cmb` with the appropriate alternating operation.
5. If `fifo` is 1, `cmb` enters the queue and is replaced by the output of the queue.
6. **The output 16-bit word of this cycle is `cmb`: make a XOR between `cmb` and the corresponding word in the cleartext.**
7. Advance all the `mwc`.
8. Advance once more the `mwc` indexed by $4sup2 + 2sup1 + sup0$.
9. Switch the orbit of the `mwc` indexed by $4swi2 + 2swi1 + swi0$.
10. `newlea` is the `mwc` indexed by $4lea2 + 2lea1 + lea0$.
11. If `cplt` is 1, bit-to-bit complement `newlea`.
12. The new `lea` is `newlea`.
13. Go to step 1.

4 Statistical Evaluation

We have used the statistical tests defined by D.E.Knuth[3] to check the random behavior of the output word of JEROBOAM.

These empirical tests are: frequency test, serial test, gap test, poker test, coupon collector's test, permutation test, run test, max-of- t test, collision test and serial correlation test.

We did not notice any significant bias, every bit of the output behaves as a coin-tossing experiment does, independently from his neighbours.

A full but quite boring evaluation of these statistics can be found in [4].

5 Performances

We ciphered 1, 2, ..., 10 Mbyte files with a hundredth of second precise measure on different PCs.

β denotes the number of clocks necessary to the obtention of one 16-bit output word.

Microprocessor	Exploitation system	Speed (Mbyte/s)	β
Pentium 100	Windows 95	0.73	261
Pentium 120	Windows 95	0.88	260
Pentium 166	Windows 95	1.21	261
Pentium 166MMX	Windows NT 4	1.17	270
Pentium 200MMX	Windows 95	1.40	272

The C compiler is Microsoft Visual C++ version 4.2, using Pentium code generation.

Let us try to find a rapid estimation of the cost of a JEROBOAM cycle on a standard Pentium. With the slowest addressing mode on a Pentium[7]:

Operation	Operand	Size (bits)	Quantity	Cycles
unsigned multiplication	MUL	16.16→32	9	11
substraction	SUB	32	1	2
logical and	AND	16	16	2
addition	ADD	16	11.75	2
right shift	SHR	16	9	2
exclusive or	XOR	16	2.75	2
complement	NEG	16	0.5	2
Total cycle number C				181

This is 30% less of the above observation, but it is still the same ordre of magnitude.

Computing C for a N MHz frequency microprocessor, one can estimate the enciphering speed v in megabyte per second at

$$v \approx 1.9 \frac{N}{C}.$$

We finally note that the speed of JEROBOAM strongly depends on the speed of the multiplication of two unsigned 16-bit words. For instance, experiments on an ARM processor bring us to a 25 % speeding up.

6 Conclusion : On the Cryptanalysis of Two or Three mwc

We have seen that a mwc used alone is insecure.

We would now like to insist on the fact that, given the sequence formed by the sum in $\mathbb{Z}/2^{16}\mathbb{Z}$ of two mwc x_n and y_n in two different finite fields $\mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}/q\mathbb{Z}$, we do not know how to recover the initial terms x_0 and y_0 .

The output of JEROBOAM is far more tricky. In particular, this output always implies a third mwc with an XOR operation, ie another algebraic structure.

We invite the reader to determine how to recover the initial content of two mwc given the LSBs of their sum, then to incorporate a third mwc with $\oplus$, and finally crack JEROBOAM...

7 Appendix : C Source Code and Test Values

7.1 A Readable but Slow C Implementation

```

/* 16-bit stream cipher JEROBOAM 2.0
   Readable but slow C implementation */

#include <stdio.h>

typedef unsigned short w16 ;
typedef unsigned long  w32 ;

static w16 a[8]={61914,42903,57225,43995,54564,47529,53130,51813};
static w32 mwc[8];
static w16 lea,queue1,queue2 ;

#define nsetupcycle 21

void clockmwc(int i)
{ mwc[i]=(w32)(w16)mwc[i]*a[i]+(mwc[i]>>16); }
void switchmwc(int i)
{ mwc[i]=((w32)(a[i]-1)<<16)+0xffff-(w32)mwc[i]; }

w16 elemcycle()
{
int five,chop,fifo,cplt;
int lea2,lea1,lea0;
int ini2,ini1,ini0;
int sup2,sup1,sup0;
int swi2,swi1,swi0;
w16 newlea,cmb,save;
int i0,i;
swi1=(lea&0x0001)?1:0;
cplt=(lea&0x0002)?1:0;
swi0=(lea&0x0004)?1:0;
lea2=(lea&0x0008)?1:0;
sup1=(lea&0x0010)?1:0;
ini0=(lea&0x0020)?1:0;
lea1=(lea&0x0040)?1:0;
ini2=(lea&0x0080)?1:0;
fifo=(lea&0x0100)?1:0;
five=(lea&0x0200)?1:0;
sup2=(lea&0x0400)?1:0;
ini1=(lea&0x0800)?1:0;
chop=(lea&0x1000)?1:0;
lea0=(lea&0x2000)?1:0;
sup0=(lea&0x4000)?1:0;
swi2=(lea&0x8000)?1:0;
newlea=mwc[4*lea+2*lea1+lea0];
if (cplt) newlea=newlea;
i0=4*ini2+2*ini1+ini0;
cmb=mwc[i0];
for (i=1;i<=3+five;i++)
{

```



```

    if (chop) cmb=cmb^mwc[(i0+i)%8]; else cmb=cmb+mwc[(i0+i)%8];
    chop=!chop;
}
if (fifo)
{
    save=queue1 ;
    queue1=queue2 ;
    queue2=cmb ;
    cmb=save ;
}
for (i=0;i<8;i++) clockmwc(i);
clockmwc(4*sup2+2*sup1+sup0);
switchmwc(4*swi2+2*swi1+swi0);
lea=newlea;
return cmb;
}

void cipher(unsigned char *msg,int sizeinw16)
{
    w16 *doublemsg;
    int compt ;
    doublemsg=(w16*)msg ; /* ! pentium is little-endian */
    for (compt=0;compt<sizeinw16;compt++)
        doublemsg[compt]=doublemsg[compt]^elemcycle() ;
}

void key248(w32 key0,w32 key1,w32 key2,w32 key3,
            w32 key4,w32 key5,w32 key6,w32 key7)
{
    w16 dumb[nsetupcycle];
    if (!(key0&&key1&&key2&&key3&&key4&&key5&&key6&&key7))
        { printf("incorrect key: a 32-bit word is zero\n"); exit(1); }
    if ((key0|key1|key2|key3|key4|key5|key6|key7)&0x80000000)
        { printf("incorrect key: a 32th bit is non-zero\n"); exit(1); }
    mwc[0]=key0 ; mwc[1]=key1 ; mwc[2]=key2 ; mwc[3]=key3 ;
    mwc[4]=key4 ; mwc[5]=key5 ; mwc[6]=key6 ; mwc[7]=key7 ;
    lea=mwc[0] ;
    queue1=0xda37 ;
    queue2=0xc07f ;
    cipher((unsigned char*)dumb,nsetupcycle) ;
}

void key128(w16 key0,w16 key1,w16 key2,w16 key3,
            w16 key4,w16 key5,w16 key6,w16 key7)
{
    key248(0x00010000+key0,0x00020000+key1,0x00030000+key2,0x00040000+key3,
           0x00050000+key4,0x00060000+key5,0x00070000+key6,0x00080000+key7) ;
}

```

The fast but unreadable version, essentially written with C preprocessor macros will not be shown here. It can be found in [4].

7.2 Differences Between JEROBOAM 1.0 and JEROBOAM 2.0

JEROBOAM 1.0 is the cipher we study in [4].

This article presents an extended version : JEROBOAM 2.0. There are three differences:

- JEROBOAM 1.0 does not support 248-bit keys. He has just one 128-bit key-insertion function called `clef(...)`, and

$$\text{JEROBOAM 1.0:clef}(\dots) \equiv \text{JEROBOAM 2.0:key128}(\dots).$$

- JEROBOAM 1.0's `chiffre(...)` function only ciphers 1024 byte blocks:

$$\text{JEROBOAM 1.0:chiffre}(\dots) \equiv \text{JEROBOAM 2.0:cipher}(\dots, 512).$$

- The setup of JEROBOAM 2.0 is 21 cycle long. JEROBOAM 1.0's setup is 512 cycle long.

7.3 Test Values

Here's what we get with the following 248-bit key if we encrypt the zero message:

```
key248(7323aafc,01638ef6,20903ffa,7f8750d0,2275e0d1,36da83da,4fe33cca,38743eca)
0000-000f : e4 e3 d8 12 06 36 20 73 00 2d cc 66 29 6b 4f 9d
0010-001f : ca 6e 48 a2 a2 be 9e 61 b1 f1 f9 0d 25 0b 06 66
...
```

Acknowledgements

The first author wishes to thank the CONFIDENCE's team. Both authors would like to express their gratitude to Professor Jacques Stern for fruitful discussions about JEROBOAM and the anonymous referees for their advices.

References

1. FRIEZE A.M., HASTAD J., KANNAN R., LAGARIAS J.C., SHAMIR A., “ Reconstructing truncated integer variables satisfying linear congruences ” *SIAM Jour. Comp.*, 17 (1988) 262–280.
2. ROGAWAY P.⁴, COPPERSMITH D., “ A software-optimized encryption algorithm ” R. Anderson, editor, *Fast software encryption, Cambridge Security Workshop*, (LNCS 809) 56-63, 1994.
3. KNUTH D.E., *The Art of Computer Programming*, volume 2, Addison-Wesley, 1984.
4. MICHON E., Rapport de stage d'option scientifique: étude cryptologique du chiffreur JEROBOAM, Document L^AT_EX 2_ε, 74 ko, Ecole Polytechnique, june 1997.

⁴ SEAL has been redesigned to SEAL 3.0. A description of this new version can be found at <http://www.csif.cs.ucdavis.edu/rogaway/papers/list.html>

5. MARSAGLIA G., Diehard, [ftp stat.fsu.edu/pub/diehard/diehard.zip](ftp://stat.fsu.edu/pub/diehard/diehard.zip).
6. FISHMAN G.S., “ Multiplicative congruential random number generators with modulus 2^β ” *Math. Comp.*, 54 (1990) 331–344.
7. INTEL, *Developers’ insight CDROM*, january 1997, F. Instruction format and timing, in /design/pentium/manuals/241430_4.pdf.
8. LAI X., MASSEY J.L., “ A proposal for a new block encryption standard ” I.Darmgård, editor, *EUROCRYPT 90, Advances in Cryptology*, (LNCS 473) 389–404, 1991.
9. FISHMAN G.S., MOORE L.R. “ An exhaustive analysis of multiplicative congruential random number generators with modulus $2^{31} - 1$ ” *SIAM Jour. Comp.*, 7 (1986), 24–45.
10. JENKINS R.J., “ ISAAC ” D. Gollman, editor, *Fast Software Encryption, Third International Workshop*, (LNCS 1039) 41–50, 1996.
11. LAI X., MASSEY J.L., MURPHY S., “ Markov ciphers and differential cryptanalysis ” D.W.Davies, editor, *EUROCRYPT 91, Advances in Cryptology*, (LNCS 547) 17–38, 1991.