Cryptanalysis

Lecture Block 4: Block Ciphers

John Manferdelli
jmanfer@microsoft.com
JohnManferdelli@hotmail.com

Remember, Luke

- Linear cryptanalysis can be accomplished with ~2^{43} known plaintexts, using a more sophisticated estimation 14 round approximation
  - For each 48 bit last round subkey, decrypt ciphertext backwards across last round for all sample ciphertexts
  - Increment count for all subkeys whose linear expression holds true to the penultimate round
  - This is done for the first and last round yielding 13 key bits each (total: 26)

- Here they are:

  $P_R[8,14,25] \oplus C_L[3,8,14,25] \oplus C_R[17] = K_1[26] \oplus K_3[4] \oplus K_4[26] \oplus K_6[26] \oplus K_7[4] \oplus K_8[26] \oplus K_{10}[26] \oplus K_{11}[4] \oplus K_{12}[26] \oplus K_{14}[26]$

  with probability **½ -1.19x2^{-21}**

  $C_R[8,14,25] \oplus P_L[3,8,14,25] \oplus P_R[17] = K_{13}[26] \oplus K_{12}[24] \oplus K_{11}[26] \oplus K_9[26] \oplus K_8[24] \oplus K_7[26] \oplus K_5[26] \oplus K_4[4] \oplus K_3[26] \oplus K_1[26]$

  with probability **½ -1.19x2^{-21}**

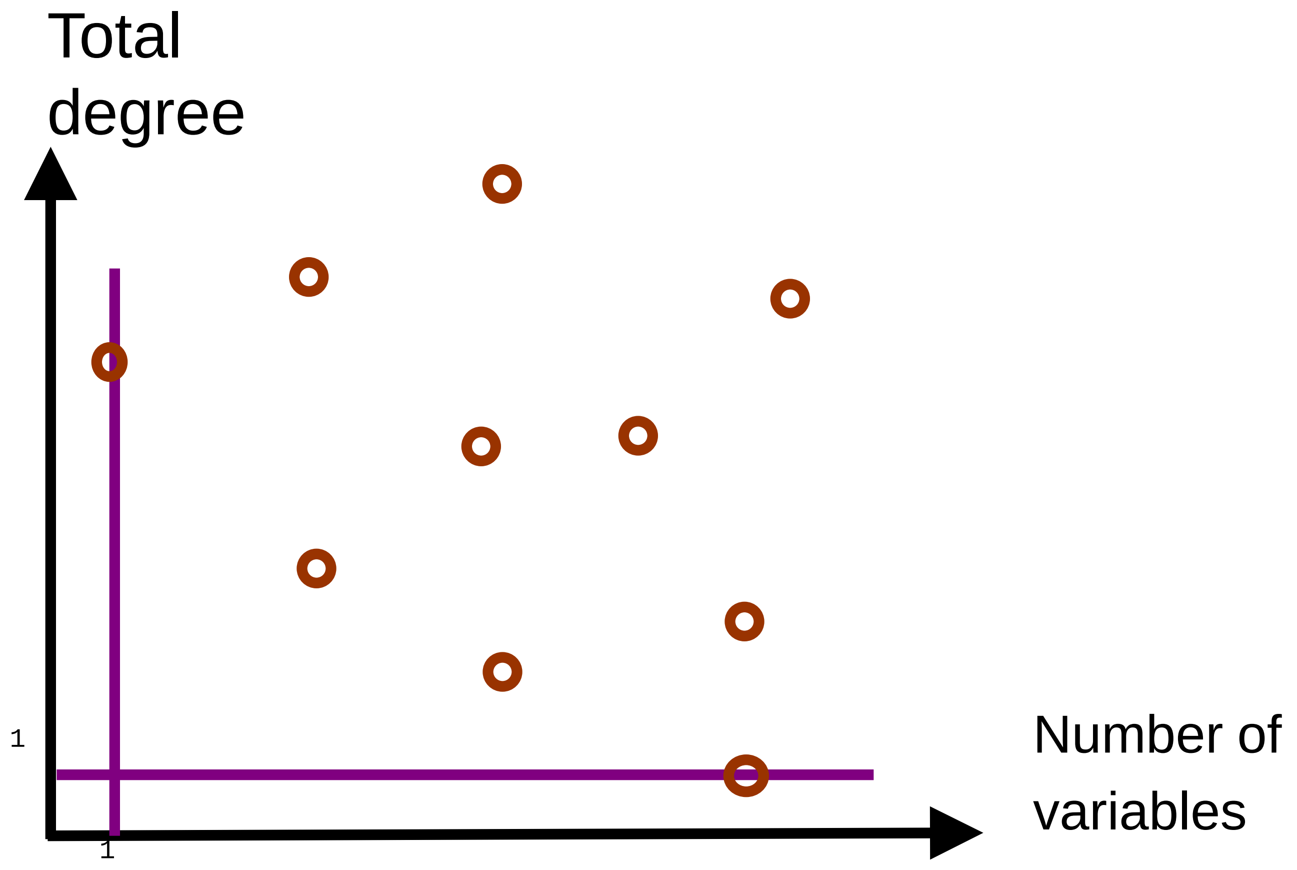
S Boxes as Polynomials over GF(2)

```
1,1:
   56+4+35+2+26+25+246+245+236+2356+16+15+156+14+146+145+13+1
   35+134+1346+1345+13456+125+1256+1245+123+12356+1234+12346

1,2:
   C+6+5+4+45+456+36+35+34+346+26+25+24+246+2456+23+236+235+2
   34+2346+1+15+156+134+13456+12+126+1256+124+1246+1245+12456
   +123+1236+1235+12356+1234+12346

1,3:
   C+6+56+46+45+3+35+356+346+3456+2+26+24+246+245+236+16+15+1
   45+13+1356+134+13456+12+126+125+12456+123+1236+1235+12356+
   1234+12346

1,4:
   C+6+5+456+3+34+346+345+2+23+234+1+15+14+146+135+134+1346+1
   345+1256+124+1246+1245+123+12356+1234+12346
```

Legend: C+6+56+46 means $1 \oplus x_6 \oplus x_5 x_6 \oplus x_4 x_6$

The only easily solvable cases of simultaneous algebraic equations

Slide from Adi Shamir



Boolean Functions

- f :$GF(2)^n \rightarrow GF(2)$ and g : $GF(2)^n \rightarrow GF(2)$,
- $C(f, g) = 2\text{Prob}(f(x) = g(x)) - 1$.
- Consider two real vectors, in R^N, $N=2^n$.
- **a**$= ((-1)^{f(0)}, (-1)^{f(1)},\ldots, (-1)^{f(N-1)})$
- **b**$= ((-1)^{g(0)}, (-1)^{g(1)},\ldots, (-1)^{g(N-1)})$
- $<f,g> = (\mathbf{a},\mathbf{b})$, $||f|| = \sqrt{(<f,f>)}$
- $C(f,g) = <f,g>/(||f||\ ||g||)$

Walsh transform and polynomials

- $\mathfrak{W}(f)(w)=F(w) = 2^{-n} \sum_x (-1)^{f(x)\oplus(w,x)}$
- $\sum_w F(w)^2 = 1$ (Parseval).
- If f(x) = g(Mx+b), M, invertible, the absolute value of the spectrums of F and G are the same.
- If * is the convolution operation, $\mathfrak{W}(f*g)(w)= \mathfrak{W}(f)(w)\ \mathfrak{W}(g)(w)$.
- If f is boolean function on n variables $x_1, x_2, \ldots, x_n$ and **a**=(a1, a2, …, an) then $f(x_1, x_2, \ldots, x_n)= \sum_{\mathbf{a}} g(\mathbf{a})\ x_1^{a1}\ x_2^{a2} \ldots, x_n^{an}$ where $g(\mathbf{a}) = \sum_{\mathbf{b<a}} f(b_1, b_2, \ldots, b_n)$. Here **b<a** means the binary representation of b does not have a 1 unless there is a corresponding 1 in the representation of a.

Walsh transform - continued

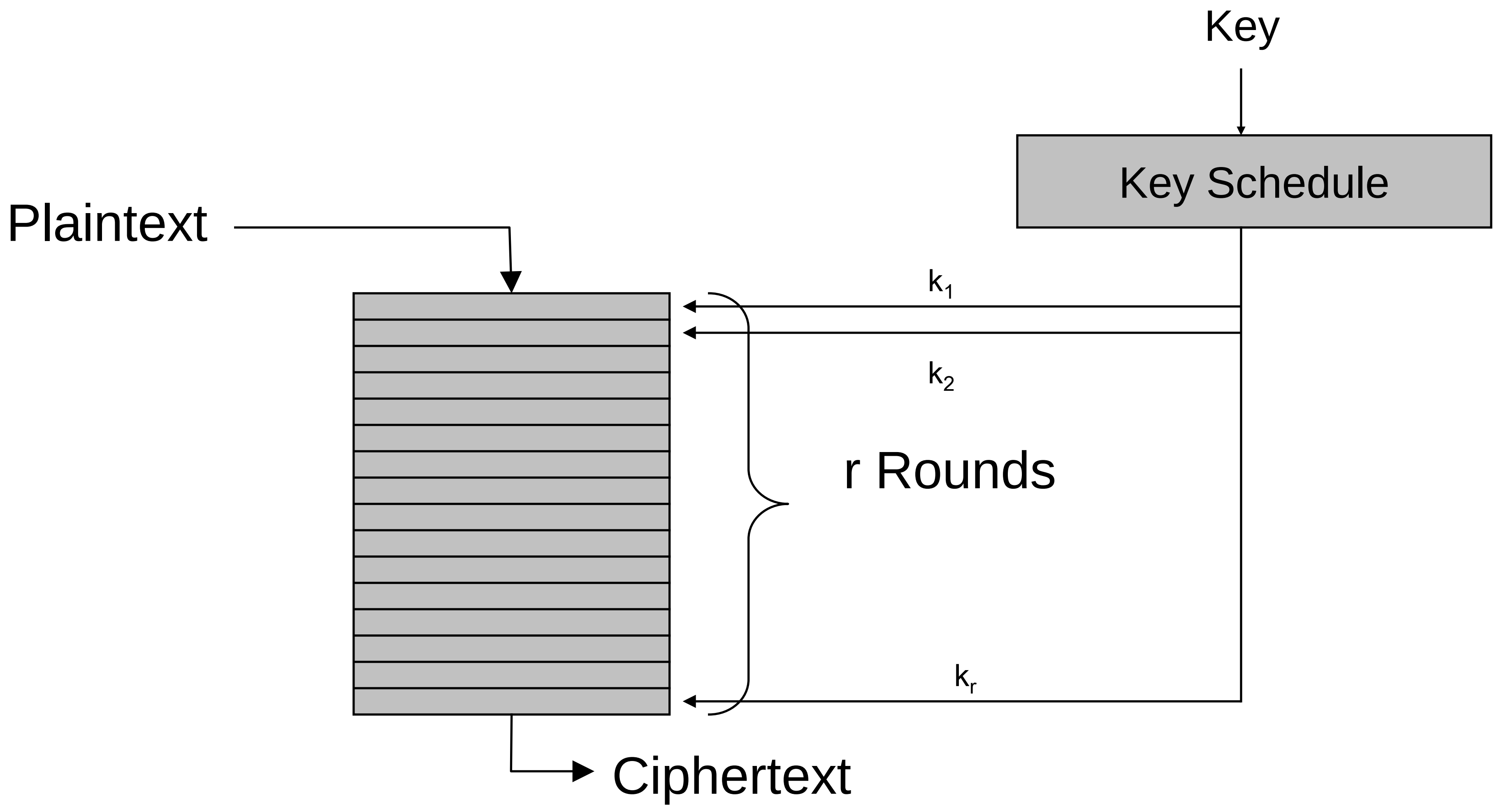
- $2^n F(w) = (a-d)$, a= # of agreements, d= # disagreements
- $a+d= 2^n$
- So $2a=2^n (F(w)+1)$
- $a= 2^{n-1} (F(w))+1)$
- Best affine approximation is the one that maximizes $|F(w)|$.

AES History

- Call for DES successor 1/97
- Nine Submissions
  - CAST-256, CRYPTON, DEAL, DFC (cipher), E2, FROG, HPC, LOKI97, MAGENTA, MARS, RC6, Rijndael, SAFER+, Serpent, and Twofish.
- Finalists
  - MARS, RC6, Rijndael, Serpent, and Twofish
- And the winner is Rijndael: FIPS 197 published 11/2001

- Good References:
  - Daemen and Rijimen, The Design of Rijndael. Springer.
  - Ferguson et. al., The Twofish Encryption Algorithm. Wiley.
  - Tons of contemporaneous material, thesis, etc. Almost all on WWW.

AES



AES Requirements

- 128, 192, 256 bit keys
- Algorithms will be judged on the following factors:
  - Actual security of the algorithm compared to other submitted algorithms (at the same key and block size).
  - The extent to which the algorithm output is indistinguishable from a random permutation on the input block.
  - Soundness of the mathematical basis for the algorithm’s security.
  - Other security factors raised by the public during the evaluation process, including any attacks which demonstrate that the actual security of the algorithm is less than the strength claimed by the submitter.
  - Claimed attacks will be evaluated for practicality.
- Key agility (NSA): “Two blocks encrypted with two different keys should not take much more time than two blocks encrypted with the same key.

DESX and whitening

- Attacks like differential and linear cryptanalysis are easier since we can direct observe the input to the first round and output of the last round directly.
- Rivest and Killian:

  $\text{DESX}(k_1,k_2,k_3,x)= k_3 \oplus \text{DES}(k_1, k_2 \oplus x)$
- Strategy adopted by almost all the AES participants.

Mars (Multiplication, Addition, Rotation and Substitution)

Basic Structure

1. Whiten
2. 8 rounds of key independent mixing
3. 16 rounds of keyed Feistel transforms (2 S-boxes)
4. 8 rounds of key independent mixing
5. Whiten

RC6 Design Philosophy

- Leverage our experience with RC5: use *data-dependent rotations* to achieve a high level of security.
- Adapt RC5 to meet AES requirements
- Take advantage of a new primitive for increased security and efficiency: *32x32 multiplication*, which executes quickly on modern processors, to compute rotation amounts.

Slide by Ron Rivest (Second AES Conference)

Description of RC6

- RC6-w/r/b parameters:
  - *Word size* in bits: w (32)(lg(w) = 5)
  - Number of *rounds*: r (20)
  - Number of *key bytes*: b (16, 24, or 32)
- Key Expansion:
  - Produces array S[0 … 2r + 3] of w-bit *round keys.*
- Encryption and Decryption:
  - Input/Output in 32-bit registers A,B,C,D

Slide by Ron Rivest (Second AES Conference)

RC6 Primitive Operations

$A + B$	Addition modulo 2^w
$A - B$	Subtraction modulo 2^w
$A \oplus B$	Exclusive-Or
$A <<< B$	Rotate A left by amount in low-order lg(w) bits of B
$A >>> B$	Rotate A right, similarly
$(A,B,C,D) = (B,C,D,A)$	Parallel assignment
$A \times B$	Multiplication modulo 2^w

Slide by Ron Rivest (Second AES Conference)

RC6 Encryption (Generic)

```
B = B + S[ 0 ]
D = D + S[ 1 ]
for  i  =  1  to  r  do
      {
          t  =  ( B  x  ( 2B  + 1 ) )  <<<  lg( w )
          u  =  ( D  x  ( 2D + 1 ) )  <<<  lg( w )
          A  =  ( ( A ⊕ t )  <<<  u )  +  S[ 2i ]
          C  =  ( ( C  ⊕ u )  <<<  t )  +  S[ 2i+1 ]
          (A, B, C, D)  =  (B, C, D, A)
       }
A = A + S[ 2r + 2 ]
C = C + S[ 2r + 3 ]
```

Slide by Ron Rivest (Second AES Conference)

RC6 Encryption (for AES)

```
 B = B + S[ 0 ]
D = D + S[ 1 ]
for  i  =  1  to  20  do
    {
        t  =  ( B  x  (2B+1) ) <<<  5
        u  =  ( D  x  (2D+1) ) <<<  5
        A  =  ( ( A ⊕ t ) <<< u ) + S[ 2i ]
        C  =  ( ( C ⊕ u ) <<< t ) +  S[ 2i+1 ]
        (A, B, C, D)  =  (B, C, D, A)
     }
A = A + S[ 42 ]
C = C + S[ 43 ]
```

Slide by Ron Rivest (Second AES Conference)

RC6 Decryption (for AES)

```
C =  C - S[ 43 ]
A =  A - S[ 42 ]
for  i  =  20  downto  1  do
    {
    (A, B, C, D)  =  (D, A, B, C)
    u  =  ( D  x  ( 2D + 1 ) )  <<<  5
    t  =  ( B  x  ( 2B  + 1 ) )  <<<  5
    C  =  ( ( C - S[ 2i + 1 ] ) >>> t ) ⊕ u
    A  =  ( ( A - S[ 2i ] ) >>> u ) ⊕ t
    }
 D = D - S[ 1 ]
 B = B - S[ 0 ]
```

Slide by Ron Rivest (Second AES Conference)

Key Expansion (Same as RC5's)

- Input: array L[0 … c-1] of input key words
- Output: array S[0 … 43] of round key words
- Procedure:

```
S[ 0 ] = 0xB7E15163
for  i = 1  to  43  do S[i] = S[i-1] + 0x9E3779B9
A = B = i = j = 0
for  s = 1  to  132  {
     A = S[ i ] = ( S[ i ] + A + B ) <<< 3
     B = L[ j ] = ( L[ j ] + A + B ) <<< ( A + B )
     i = ( i + 1 )   mod 44
     j = ( j + 1 )  mod c
     }
```

Slide by Ron Rivest (Second AES Conference)

Encryption Rate (200MHz)

Slide by Ron Rivest (Second AES Conference)

MegaBytes / second
MegaBits / second

	Java	Borland C	Assembly
Encrypt	0.197	5.19	12.6
	1.57	*41.5*	*100.8*
Decrypt	0.194	5.65	12.6
	1.55	*45.2*	*100.8*

Over 100 Megabits / second !

Linear analysis

- Find approximations for r-2 rounds.
- Two ways to approximate A = B <<< C
  - with one bit each of A, B, C (type I)
  - with one bit each of A, B only (type II)
  - each have bias 1/64; type I more useful
- Non-zero bias across f(B) only when input bit = output bit. (Best for lsb.)
- Also include effects of multiple linear approximations and linear hulls.

Slide by Ron Rivest (Second AES Conference)

Security against linear attacks

- Estimate of number of plaintext/ciphertext pairs required to mount a linear attack.
- (Only 2^{128} such pairs are available.)

Slide by Ron Rivest (Second AES Conference)

Rounds		Pairs	
8		2^{47}	
12		2^{83}	
16		2^{119}	
20	$\longleftarrow$ RC6 $\longrightarrow$	2^{155}	Infeasible
24		2^{191}	

Differential analysis

- Considers use of (iterative and non-iterative) (r-2)-round *differentials* as well as (r-2)-round *characteristics.*
- Considers two notions of “difference”:
  - exclusive-or
  - subtraction (better!)
- Combination of quadratic function and fixed rotation by 5 bits very good at thwarting differential attacks.

Slide by Ron Rivest (Second AES Conference)

An iterative RC6 differential

A	B	C	D
1<<16	1<<11	0	0
1<<11	0	0	0
0	0	0	1<<s
0	1<<26	1<<s	0
1<<26	1<<21	0	1<<v
1<<21	1<<16	1<<v	0
1<<16	1<<11	0	0

- Probability = 2^{-91}

Slide by Ron Rivest (Second AES Conference)

Security against differential attacks

Estimate of number of plaintext pairs required to mount a differential attack.

(Only 2^{128} such pairs are available.)

Rounds		Pairs	
8		2^{56}	
12		2^{117}	
16		2^{190}	
20	$\longleftarrow$ RC6 $\longrightarrow$	2^{238}	Infeasible
24		2^{299}	

TwoFish Observations

- Didn’t use multiplication unlike other candidates
- Uses same primitives for key schedule generation as basic round functions
- Key dependant S-box built from two 256 S-Boxes.
- Two non-independent S-Boxes built from 8 fixed 16 element permutations picked for statistical properties.

TwoFish

Basic Structure for 128 bit operation.

- Construct 40 32 bit round keys K_0 ,…, K_{39}
- Input Whiten
- 16 Keyed rounds
- Output Whiten (after switching left and right blocks)

- Input bytes p_0, p_1, …, p_{15}. Little Endian as 32 bit words.
  - $P_0=p_0+p_1 2^8+p_2 2^{16}+p_3 2^{24,}$ $P_1=p_4+p_5 2^8+p_6 2^{16}+p_7 2^{24}$
  - $P_2=p_8+p_9 2^8+p_{10} 2^{16}+p_{11} 2^{24,}$ $P_3=p_{12}+p_{13} 2^8+p_{14} 2^{16}+p_{15} 2^{24}$
- Same for Output c_0, …, c_{15} = C_0, C_1, C_2, C_3

- Output of round r designated R_0^r, R_1^r, R_2^r, R_3^r

- $R_0^r=P_0$, $R_1^r=P_1$, $R_2^r=P_2$, $R_3^r=P_3$

TwoFish

PHT:

$a' = a+b \pmod{2^{32}}$

$b' = a+2b \pmod{2^{32}}$

```
            0x01 0xef 0x5b 0x5b
MDS=        0x5b 0xef 0x5b 0x01
            0xef 0x5b 0x01 0xef
            0xef 0x01 0xef 0x5b
```

GF(256) calculations (MDS) use modulus

$x^8 + x^6 + x^5 + x^3 + 1$ over GF(2).

TwoFish

- Input Whiten

  $R_0^0=P_0 \oplus K_0$, $R_1^0=P_1 \oplus K_1$,

  $R_2^0=P_2 \oplus K_2$, $R_3^0=P_3 \oplus K_3$

- 16 Keyed Rounds

  $F_1(X,Y,r)$, $F_2(X,Y,r)$ defined later ($2^{32} \times 2^{32} \times I \rightarrow 2^{32}$)

  $R_0^{r+1}= \mathrm{ror}(R_1^r \oplus F_1(R_0^r, R_1^r,r+1), 1)$

  $R_1^{r+1}= \mathrm{rol}(R_1^r \oplus F_2(R_0^r, R_1^r,r+1), 1)$

  $R_2^{r+1}=R_0^r$, $R_3^{r+1}=R_1^r$

- Output Whiten (after switching left and right blocks)

  $C_0 =R_3^{16} \oplus K_{36}$, $C_1 =R_4^{16} \oplus K_{37}$,

  $C_2 =R_0^{16} \oplus K_{38}$, $C_3 =R_1^{16} \oplus K_{39}$

TwoFish Round Functions

- $F_1(X,Y,r) = g(X) + g(ror(Y,8)) + K_{2r+4} \pmod{2^{32}}$
- $F_2(X,Y,r) = g(X) + 2g(ror(Y,8)) + K_{2r+5} \pmod{2^{32}}$

- $g(x)=h(x,S)$, where h and S are defined below

TwoFish Key Schedule

```
        01 a4 55 87 5a 58 db 9e
        a4 56 82 f3 1e c6 68 e5
RS=     02 a1 fc c1 47 ae 3d 19
        a4 55 87 5a 58 db 9e 03
```

k= 2, Key M consists of 16 bytes m_0, m_1, …, m_{15} or 4 32 bit words (Little endian) M_0 , M_1, M_2, M_3.

M_e= M_0 , M_2
M_o= M_1 , M_3

$(s_{i,0}, s_{i,1}, s_{i,2}, s_{i,3})^T =$ RS $(m_{8i}, m_{8i+1}, \ldots, m_{8i+7})^T$, k= 0,1

TwoFish Key Schedule and S-Boxes

$\square = 2^{24}+2^{16}+2^{6}+1$

$A_i = h(2i\ \square,\ M_e)$
$B_i = rol(h((2i+1)\ \square,\ M_o),8)$
$K_{2i} = (A_i + B_i)\ (mod\ 2^8)$
$K_{2i+1} = rol((A_i + 2B_i)\ (mod\ 2^8),9)$

$S_i = s_{i,0} + s_{i,1}2^8+s_{i,2}2^{16}+s_{i,3}2^{24}$
$S=(S_1,\ S_0)$

The Function h

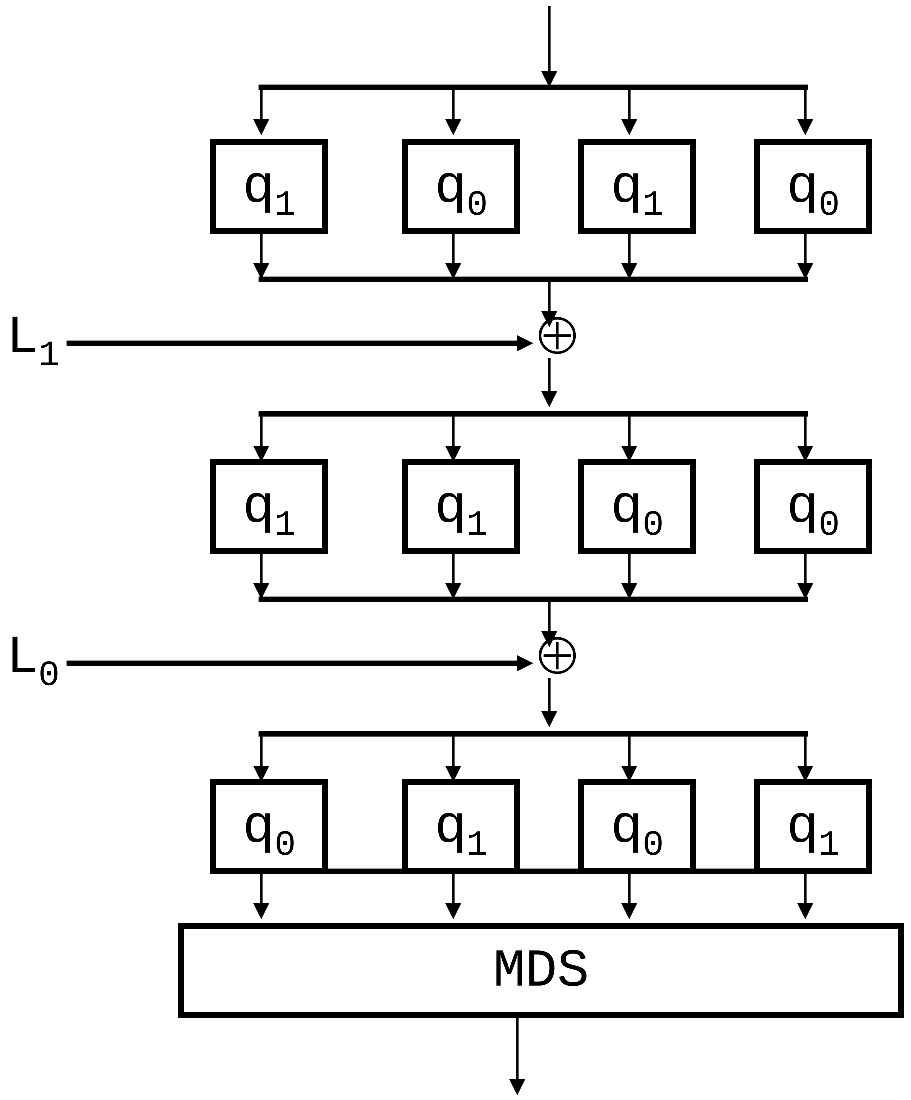
$h(X, L_0, L_1)$

$l_{i,j} = \mathrm{int}(L_i/2^{8j}) \pmod{2^8}$
$x_j = \mathrm{int}(X/2^{8j}) \pmod{2^8}$
$y_{i,j} = x_j$

$y_0 = q_1[q_0[q_0[y_{2,0}] \oplus l_{1,0}] \oplus l_{0,0}]$
$y_1 = q_0[q_0[q_1[y_{2,1}] \oplus l_{1,1}] \oplus l_{0,1}]$
$y_2 = q_1[q_1[q_0[y_{2,2}] \oplus l_{1,2}] \oplus l_{0,2}]$
$y_3 = q_0[q_1[q_1[y_{2,3}] \oplus l_{1,3}] \oplus l_{0,3}]$

$(z_0, z_1, z_2, z_3)^T = \mathrm{MDS}(y_0, y_1, y_2, y_3)^T$

The Function h



q_0, q_1

For q_0

```
   t0= [8 1 7 d 6 f 3 2 0 b 5 9 e c a 4]
   t1= [e c b 8 1 2 3 5 f 4 a 6 7 0 9 d]
   t2= [b a 5 e 6 d 9 0 c 8 f 3 2 4 7 1]
   t3= [d 7 f 4 1 2 6 e 9 b 3 0 8 5 c a]
```

For q_1

```
   t0= [2 8 b d f 7 6 e 3 1 9 4 0 a c 5]
   t1= [1 e 2 b 4 c 3 7 6 d a 5 f 9 0 8]
   t2= [4 c 7 5 1 6 9 a 0 e d 8 2 b 3 f]
   t3= [b 9 5 1 c 3 d e 6 4 7 f 2 0 8 a]
```

q_0, q_1

a_0= int(x/16), b_0= x (mod 16)

a_1= $a_0 \oplus b_0$, b_1= $a_0 \oplus \mathrm{ror}_4(b_0, 1) \oplus 8a_0$

a_2= $t_0[a_1]$, b_2=$t_1[b_1]$

a_3= $a_2 \oplus b_2$, b_3= $a_2 \oplus \mathrm{ror}_4(b_2, 1) \oplus 8a_2$

a_4= $t_2[a_3]$, b_4=$t_3[b_3]$

y= $16b_4 + a_4$

Review: Arithmetic of GF(2^n)

- Suppose m(x) is an irreducible polynomial of degree n over GF(2): $m(x)= x^n + m_{n-1} x^{n-1} + \ldots + m_{0.}$
- Let a(x) and b(x) be polynomials of degree <n. They form a vector space of dimension n over GF(2). Coefficients of like exponent “add”: $(a_{n-1} x^{n-1} + \ldots + a_0)+ (b_{n-1} x^{n-1} + \ldots + b_0)= (a_{n-1}+ b_{n-1})x^{n-1} + \ldots + a_0 + b_0)$
- Euclidean algorithm: for a(x), b(x) polynomials of degrees m□n, there are polynomials q(x), r(x), deg r(x) <n such that a(x)=q(x)b(x)+r(x)
- Polynomials over GF(2) modulo m(x) form a field (with 2^n elements). Multiplication is multiplication of polynomials mod m(x).
- Inverses exist : If a(x) and b(x) are polynomials their greatest common denominator d(x) can be written as

  d(x)= a(x)u(x)+b(x)v(x) for some u(x), v(x).

In particular if a(x) and b(x) are co-prime: 1= a(x)u(x)+b(x)v(x) for some u(x), v(x).

Example of multiplication and inverse

- $m(x)= x^2 +x +1$. $m(x)$ is irreducible (otherwise it would have a root in GF(2)
- $x+(x+1) =1$, $1+(x+1)= x$
- $(x+1)(x+1)= x^2+2x+1=x^2+1= (x) + (x^2 +x +1)= x \pmod{m(x)}$
- $(x+1)$ and $m(x)$ are co-prime in fact,

  $1= (x+1) (x) + (x^2 +x +1)(1)$
- So “x” is the multiplicative inverse of “x+1” in GF(4).
- Usually elements of $GF(2^n)$ are written in place notation so $x^5+x^3 +x^2 +1=$ 101101.

Rijndael Overview

- Input
  - p consisting of N_b words
  - k with N_k words (N_k= 4,6,8)
- State
  - 4 rows, N_b columns
- Key
  - 4 rows, columns
- Output
  - c consisting of N_b words

All tables filled first col first $s_{0,0}$, $s_{1,0}$, $s_{2,0}$, $s_{3,0}$, $s_{0,1}$, …

Rijndael Overview

- Design Philosophy
  - Wide Trails
- 32 bit word operations
- Non-linear substitution uses arithmetic over GF(2)
- Mixing uses polynomial arithmetic mod (x^4+1)

Rijndael Round Structure

$N_r = \max(N_k, N_b)+6$

N_r	**N_b=4**	**N_b= 6**	**N_b=8**
N_k=4	10	12	14
N_k=6	12	12	14
N_k=8	14	14	14

Rijndael State Layout

State: $s_{i,j}$, i= Nb (mod 4), j= [Nb/4], Nb=4j+i

For Nb= 4

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

Rijndael Key Layout

- Keys: $k_{i,j}$, i= Nk (mod 4), j= [Nk/4]
- For Nk= 4

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Rijndael Algorithm

```
Rijndael (p, k, Nb, Nk)  {
    ComputeRoundKeys(K, W[0…Nr])
    state= p
    AddRoundKey(0, state)
    for (i=1, i<=Nr, i++) {
        for each byte, b in state
            ByteSub(b)
        ShiftRow(state)
        if(i<Nr)
            MixCol(state)
        AddRoundKey(i, state)
        }
    c= state
    }
```

Inverse Rijndael Algorithm

```
InvRijndael (c, k, Nb, Nk)  {
    ComputeRoundKeys(K, W[0…Nr])
    state= c
    for (i=0, i<Nr, i++) {
        AddRoundKey(Nr-i, state)
        if(i>0)
            InvMixCol(state)
        InvShiftRow(state)
        for each byte, b in state
            InvByteSub(b)
        }
    AddRoundKey(0, state)
    p= state
    }
```

ByteSub Primitive

```
ByteSub(b)
    if b==0
        t= 0
    else
        t= $b^{-1}$
return(Mt + a)
```

M= circ(1,0,0,0,1,1,1,1)

a= $(1,1,0,0,0,1,1,0)^T$

Arithmetic over GF(2) with $m(x)= x^8+x^4+x^3+x+1$.

ByteSub Data

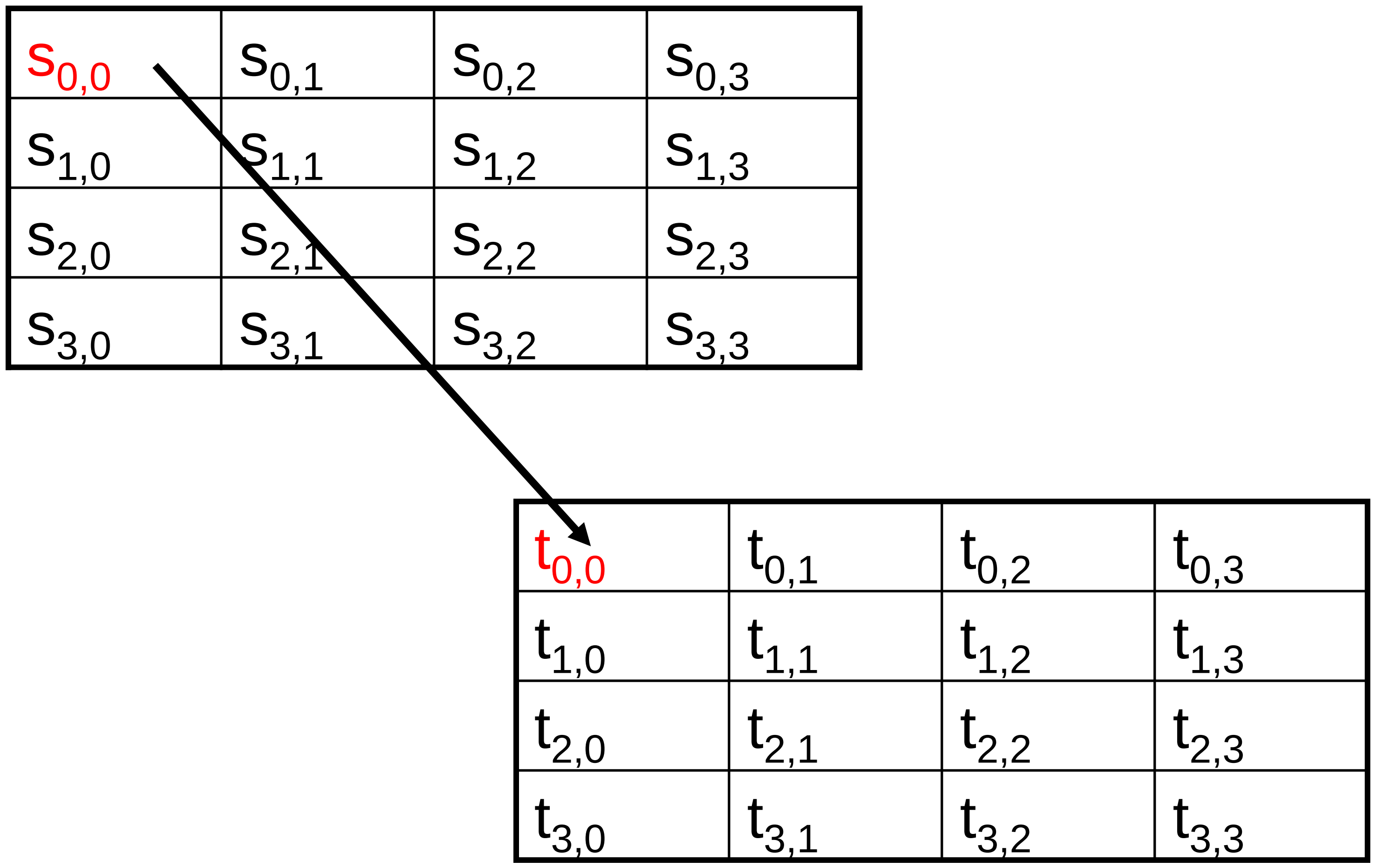
M:

1	0	0	0	1	1	1	1
1	1	0	0	0	1	1	1
1	1	1	0	0	0	1	1
1	1	1	1	0	0	0	1
1	1	1	1	1	0	0	0
0	1	1	1	1	1	0	0
0	0	1	1	1	1	1	0
0	0	0	1	1	1	1	1

a:

1
1
0
0
0
1
1
0

Bytesub



$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$t_{0,0}$	$t_{0,1}$	$t_{0,2}$	$t_{0,3}$
$t_{1,0}$	$t_{1,1}$	$t_{1,2}$	$t_{1,3}$
$t_{2,0}$	$t_{2,1}$	$t_{2,2}$	$t_{2,3}$
$t_{3,0}$	$t_{3,1}$	$t_{3,2}$	$t_{3,3}$

Rijndael Primitives

ShiftRow(state)

shift row 1 by 0.
shift row 2 by 1.
shift row 3 by 2 if Nb<8, 3 otherwise.
shift row 3 by 3 if Nb<8, 4 otherwise.

MixCol(state)

multiply each column of state by c(x) (mod x^4 +1)
c(x)= 0x03 x^3 + 0x01 x^2 + 0x01 x + 0x02

InvMixCol(state)

multiply each column of state by d(x) (mod x^4 +1)
d(x)= 0x0b x^3 + 0x0d x^2 + 0x09 x + 0x0e

AddRoundKey(i,state)

state= state + W[i]

ShiftRow

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,3}$	$s_{1,0}$	$s_{1,1}$	$s_{1,2}$
$s_{2,2}$	$s_{2,3}$	$s_{2,0}$	$s_{2,1}$
$s_{3,3}$	$s_{3,0}$	$s_{3,1}$	$s_{3,2}$

MixCol

$s_{0,0}$	$s_{0,1}$	$s_{0,3}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,3}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,3}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,3}$	$s_{3,3}$

$$t_{0,0}x^3+t_{1,0}x^2+t_{2,0}x+t_{3,0}= (0x03x^3+0x01x^2+0x01x+0x02) \times (s_{0,0}x^3+s_{1,0}x^2+s_{2,0}x+s_{3,0})(\bmod x^4+1)$$

$t_{0,0}$	$s_{0,1}$	$s_{0,3}$	$s_{0,3}$
$t_{1,0}$	$s_{1,1}$	$s_{1,3}$	$s_{1,3}$
$t_{2,0}$	$s_{2,1}$	$s_{2,3}$	$s_{2,3}$
$t_{3,0}$	$s_{3,1}$	$s_{3,3}$	$s_{3,3}$

RoundKeys

```
ComputeRoundKeys(K[4*Nk], W[Nb*(Nr+1)]) {
    for(i=0; i<Nk; i++)
        W[i]= (K[4i], K[4i+1], K[4i+2], K[4i+3])
    for(i=Nk; i<Nb*Nr+1); i++) {
        t= W[i-1]
        if((i mod Nk)==0)
            t= SubByte(RotByte(t)) + RCon(i/Nk)
        else if( (i mod Nk)==0)      // only if Nk>6
            t=SubByte(t)             // only ifNk>6
        }
    W[i]= W[i-Nk] + t
    }
```

Roundkeys Primitives

```
SubByte(w)
    w= ByteSub(w)

RotByte(w= (a,b,c,d))
    w= (b,c,d,a)

RCon[i]= (RC[i], 0x00, 0x00, 0x00);
RC[1]= 0x01
RC[i+1]=  RC[i]**(i) [multiply by “x” in polynomial
representation]
```

Cryptographic Effect

- Linear Mixing (diffusion)
  - MixCol
  - ShiftRow
- Non-Linear Mixing (confusion)
  - ByteSub
- Avalanche
  - MixCol
  - ShiftRow
  - RoundKeys

Design Criteria for ByteSub

- Invertibility
- Minimize largest non-trivial correlation between input and output (Linear resistance)
- Minimize max xor table (Differential resistance)
- Complexity of Algebraic expression in GF(2^8)
- Simplicity of description

Design Criteria for MixCol

- Invertibility (coefficient constraint)
- Linearity
- Diffusion power (coefficient constraint)
- Speed (coefficient constraint)
- Symmetry
- Simplicity

Design Criteria for Shiftrow

- Four different offsets
- Resistance against truncated differentials
- Resistance against square attack
- Simplicity

Design Criteria for KeySched

- Invertibility
- Speed
- Eliminate symmetry with round constants (weak key resistance, related key resistance)
- Diffusion of key differences
- Partial knowledge of cipher key doesn’t reveal others
- Round differences don’t reveal cipher key differences
- Don’t need to precompute entire schedule
- Simplicity

Branch Number

- Let W(a)= number of non-zero (active) bytes
- Branch Number of F= min $_{a!=0}$ W(a)+W(F(a))
- Prop ratio of differential trail appx= prop ratio of active S-boxes
- Correlation of linear trail appx = product of correlations of active S-boxes

- Wide Trail Strategy

Differential Trail

- If \beta= \rho^{(r)} \rho^{(r-1)} \ldots \rho^{(1)}, Q= (q^{(0)}, q^{(1)}, \ldots , q^{(r)}) is a differential trail whose probability is the number of a^{(0)} for which the differential tail follows the difference pattern divided by the number of possible a^{0)}.
- The weight of a differential trail is the sum of the weights of its differential steps.: w_r(Q) = \sum_i w^{\rho^{(i)}} (q^{(i-1)}, q^{(i)}).
- The differential trail imposes restrictions on the intermediate states a^{(i)}.
- Theorem: Pr(a', b')= \sum_{q^{(0)}=a', q^{(r)}=b'} Pr(Q), Pr(Q) \approx 2^{-w_r(Q)} where w_r(Q)= \sum w_r^{(\rho^{(i)}} (q^{(i-1)},q^{(i)}).

Weight Bundle

Define w_b(a) as the bundle weight of a.

B_d(\phi)= $\min_{(a, b != a)}$ (w_b (a \oplus b) + w_b (\phi(a) \oplus \phi(b))).

B_$_l$(\phi, \alpha)= min_(\alpha, \beta, C(\alpha^{T}x, \beta^T \phi(x)) != 0) (w_b (\alpha) + w_b (\beta)).

Theorem: In an alternating key block cipher with \gamma \lambda round functions, the number of active bundles in a two round trail is >= the bundle branch number of \lambda. If \psi= \gamma \Theta \gamma \lambda is a four round function, B(\psi) >= B(\lambda) \times $B^{c(\Theta)}$ where B can be either the linear or differential branch number. The linear and differential branch numbers for an AES round is 5.

Inverse provides linear/differential immunity,linear diffusion provides algebraic complexity.

Design strategy for Rijndael

- Choose number of rounds so that there is no correlation over all but a few rounds with amplitude significantly larger than 2^{- {\frac {n_b} 2}} by insuring there are no linear trails with correlation contribution above {n_k}^{-1} 2^{- {\frac {n_b} 2}} and no differential trails with weight below n_b.
- Examine round transformations \rho= \lambda \circ \gamma, where \lambda is the mixing function and \gamma is a bricklayer function that acts on bundles of n_t bits. Block size is n_b=m n_t. The correlation over \gamma is the product of correlations over different S-box positions for given input and output patterns. Define weight of correlation as -lg(Amplitude).
- If output selection pattern is \ne 0, the S-box is active. Looking for maximum amplitude of correlations and maximum difference propagation probability.
- The weight of a trail is the sum of the weights of the selection patterns or the sum of the active S-box positions ao it ia >=e number of active S-boxes times the minimum correlation weight per S-box.
- Wide trail: design round transformations so there are no trails with low bundle weight.

Rijndael Performance on 200MHz PII

(KeyLen, BlockLen)	Seed (Mb/sec)	Cycles/Blk
(128,128)	70.5	363
(192, 128)	59.3	432
(256, 128)	51.2	500

AES Finalist Bakeoff

	MARS	RC6	Rijndael (AES)	Serpent	Twofish
General Security	3	2	2	3	3
Implementation	1	1	3	3	2
SW Perf	2	2	3	1	1
Smart Card Perf	1	1	3	3	2
HW Perf	1	2	3	3	2
Design features	2	1	2	1	3

Score: 1 (low) to 3 (high). From NIST report 2 Oct 2000.

Euclidean algorithm inversion in a finite field

- Calculate (54321,9876)
  1. 54321= 5 x 9876 + 4941
  2. 9876= 1 x 4941 + 4935
  3. 4941= 4935+6
  4. 4935= 6 x 822+3
  5. 6=2 x 3

- Working Backwards:
  1. 3= (1) 4935 + (-822) 6
  2. 3= (1) 4935 + (-822)(4941-4935)= (-822) 4941 + (823) 4935
  3. 3=(-822) 4941 + (823)(9876- (1) 4941)= (823) 9876 + (-1645) 4941
  4. 3= (823) 9875 + (-1645)(54321 – (5) 9876)= (-1645) 54321 + (9048) 9876

Euclidean algorithm inversion in a finite field

- Let p=12533. What is 6^{-1} (mod p)?
- 12533= 6 x 2088 +5
- 6= (1) 5 +1
- 1= (1) 6 + (-1) 5= (1) 6 + (-1)(12533- (6) 2088)= (-1)(12533)+ (2089) (6)
- 6^{-1}= 2089 (mod 12533)

Euclidean algorithm inversion in a finite field

- Let F=GF(2), m(x)= x^2+x+1.
  1. m(x) is irreducible (why?)
  2. F[x]/(m(x)) = GF(2^2)

- What is x^{-1}?
  1. $x^2+x+1 = (x+1)x + 1$.
  2. $1 = (1)(x^2+x+1) + (x)(x+1)$
  3. $x(x+1)=1 \pmod{x^2+x+1}$

- $x^{-1} = (x+1) \pmod{x^2+x+1}$

Example: polynomial representation

- If f is boolean function on n variables x_1, x_2, …, x_n and **a**=(a1, a2, …, an) then $f(x_1, x_2, \ldots, x_n) = \sum_{\mathbf{a}} g(\mathbf{a})\, x_1^{a1}\, x_2^{a2} \ldots, x_n^{an}$ where $g(\mathbf{a}) = \sum_{\mathbf{b<a}} f(b_1, b_2, \ldots, b_n)$. Here **b<a** means the binary representation of b does not have a 1 unless there is a corresponding 1 in the representation of a.

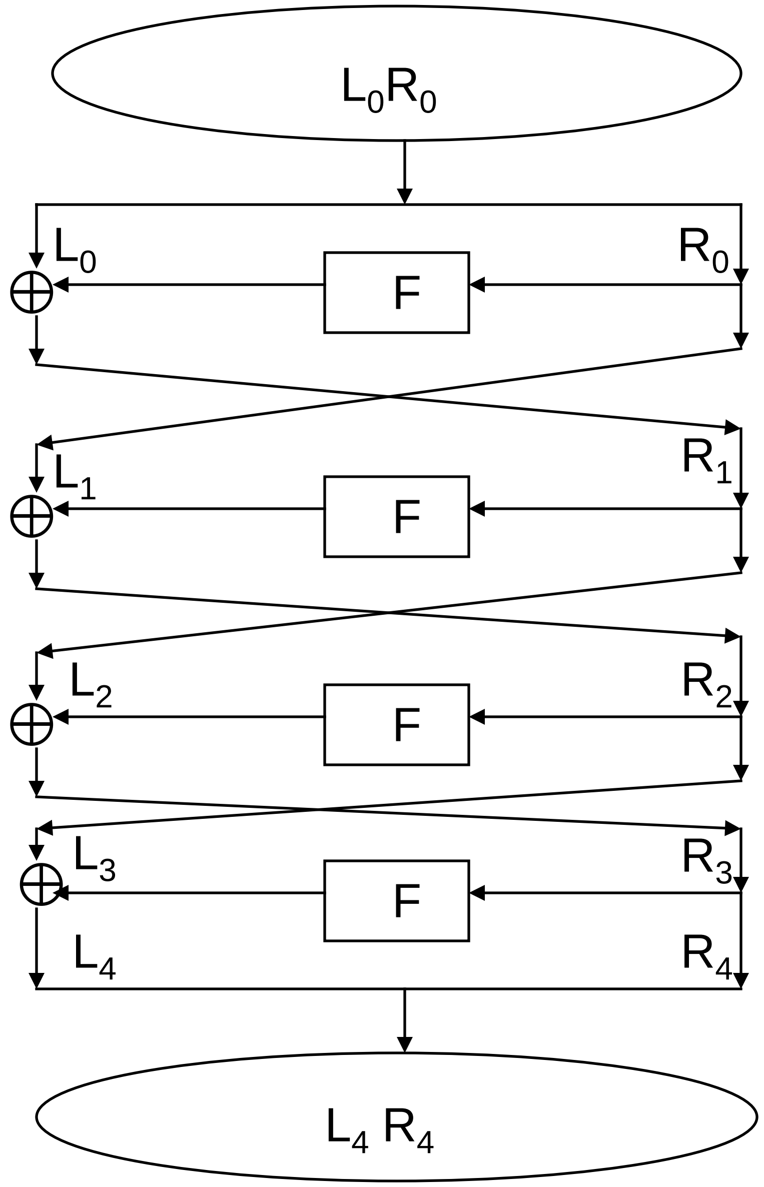
- g(0,0,0)= f(0,0,0)=1
- g(0,1,0)=f(0,0,0)+f(0,1,0)=0
- g(1,0,0)=f(0,0,0)+f(1,0,0)=1
- g(1,1,0)=f(0,0,0)+f(1,0,0))+f(0,1,0))+f(1,1,0)=0
- g(0,0,1)=f(0,0,0)+f(0,0,1)=0
- g(0,1,1)=f(0,0,0)+f(0,0,1) +f(0,1,0)+f(0,1,1)=1
- g(0,0,1)= g(1,0,1)= g(0,1,1)= g(1,1,1)= 0

- $f(x_1, x_2, x_3) = 1 + x_1 + x_2\, x_3$

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
1	0	0	0
0	1	0	1
1	1	0	0
0	0	1	1
1	0	1	0
0	1	1	0
1	1	1	1

Simplified DES

- $L_{i+1} = R_i$, each 6 bits.
- $R_{i+1} = L_i \oplus f(R_i, K_i)$
- K is 9 bits.
- $E(x) = (x_1\ x_2\ x_4\ x_3\ x_4\ x_3\ x_5\ x_6)$
- S_1
  - `101 010 001 110 011 100 111 000`
  - `001 100 110 010 000 111 101 011`
- S_2
  - `100 000 110 101 111 001 011 010`
  - `101 011 000 111 110 010 001 100`
- K_i is 8 bits of K starting at ith bit.



Six functions

- Consider the simplified DES examples
  - S_1
    - `101 010 001 110 011 100 111 000`
    - `001 100 110 010 000 111 101 011`
  - S_2
    - `100 000 110 101 111 001 011 010`
    - `101 011 000 111 110 010 001 100`

x_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
x_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
x_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
x_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
f_1	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0
f_2	0	1	0	1	1	0	1	0	0	0	1	1	0	1	0	1
f_3	1	0	1	0	1	0	1	0	1	0	0	0	0	1	1	1
f_4	1	0	1	1	1	0	0	0	1	0	0	1	1	0	0	1
f_5	0	0	1	0	1	0	1	1	0	1	0	1	1	1	0	0
f_6	0	0	0	1	1	1	1	0	1	1	0	1	0	0	1	0

Example: Walsh transform

- $\mathfrak{W}(f)(w)=F(w) = 2^{-n}$ $\Sigma_x (-1)^{f(x)\oplus(w,x)}$
- First bit of S_1:

x_1 x_2 x_3 x_4	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
f_1	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0
w	0	0	0	0												
f_1	1	0	0	1	0	1	1	0	0	1	1	0	0	1	1	0
w	0	0	1	1												
f_1+ x_3+ x_4	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0

- F1(0000)= 0
- F1(0011)= -0.50

Best affine approximation of f_1

- f_1

```
  0000 0001 0010 0011 0100 0101 0110 0111
     1    0    0    1    0    1    1    0
   1000 1001 1010 1011 1100 1101 1110 1111
     0    1    1    0    0    1    1    0
```

- As Poly: $1+x_4+x_3+x_2+x_1+x_2x_1$

- Spectrum:

```
 0000  0001  0010  0011  0100  0101  0110  0111
 0.00  0.00  0.00  0.50  0.00  0.00  0.00 -0.50
 1000  1001  1010  1011  1100  1101  1110  1111
 0.00  0.00  0.00 -0.50  0.00  0.00  0.00 -0.50
```

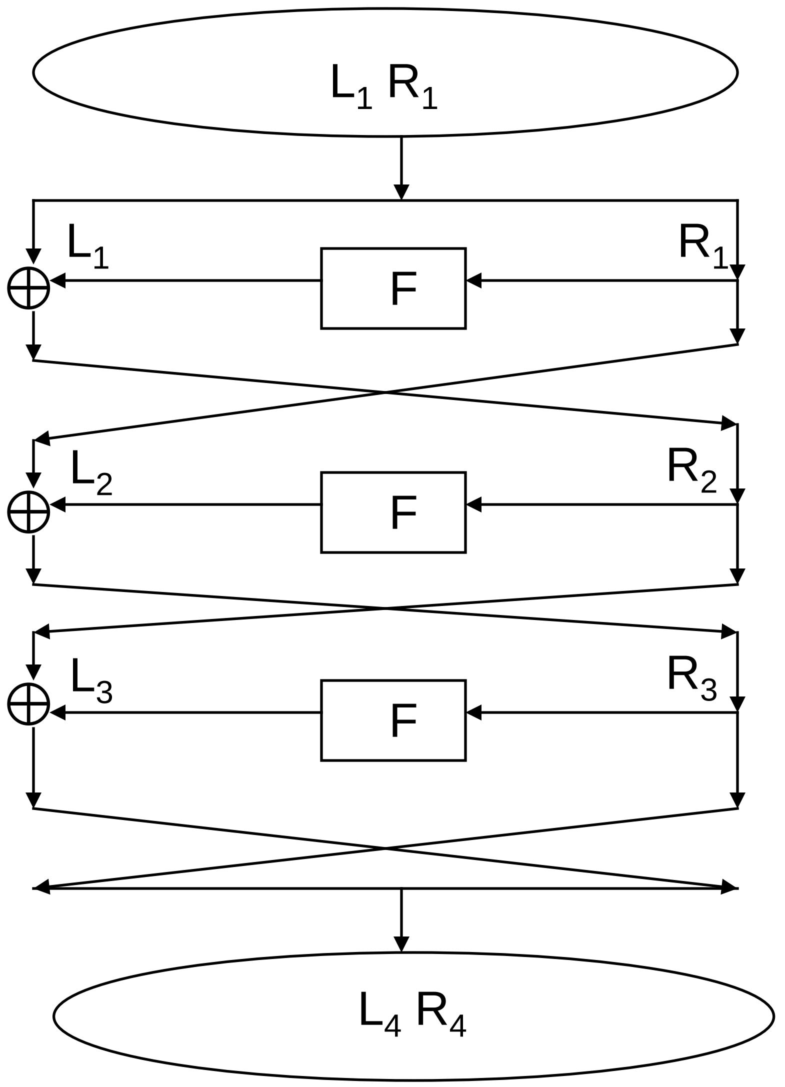
- $L(x)= x_3+x_4$ is best linear approximation. dist(f_1, L(x))= 8 (.5+1)=12

Differential Cryptanalysis – 3R

- $L_4 \oplus R_1 = f(k_3, R_2)$. .... (1)
- $R_4 \oplus L_3 = f(k_4, R_3)$. .... (2)
- $L_4 = R_3$, $L_2 = R_1$, $L_3 = R_2$.

- 1& 2 $\rightarrow$ $R_4 \oplus L_3 \oplus R_2 \oplus L_1 = f(k_2, R_1) \oplus f(k_4, R_3)$.
- $L_3 = R_2 \rightarrow R_4 \oplus L_1 = f(k_2, R_1) \oplus f(k_4, R_3)$.

- $R_4 \oplus L_1 = f(k_2, R_1) \oplus f(k_4, R_3)$. ....(3)
- $R_4^* \oplus L_1^* = f(k_2, R_1^*) \oplus f(k_4, R_3^*)$. ....(4)
- 3&4 $\rightarrow$ $R_4' \oplus L_1' = f(k_2, R_1^*) \oplus f(k_4, R_3^*) \oplus f(k_2, R_1^*) \oplus f(k_4, R_3^*)$.
- $R_1 = R_1^* \rightarrow R_4' \oplus L_1' = f(k_4, R_3) \oplus f(k_4, R_3^*)$.



Differential Cryptanalysis – 3R

L_1, R_1 : 000111 011011

L_1*, R_1*: 101110 011011

L_1', R_1': 101001 000000

L_4, R_4 : 000011 100101

L_4*, R_4*: 100100 011000

L_4', R_4': 100111 111101

$E(L_4)$: 0000 0011

$E(L_4')$: 1010 1011

$R_4' \oplus L_1'$: 111 101 $\oplus$ 101 001= 010 100.

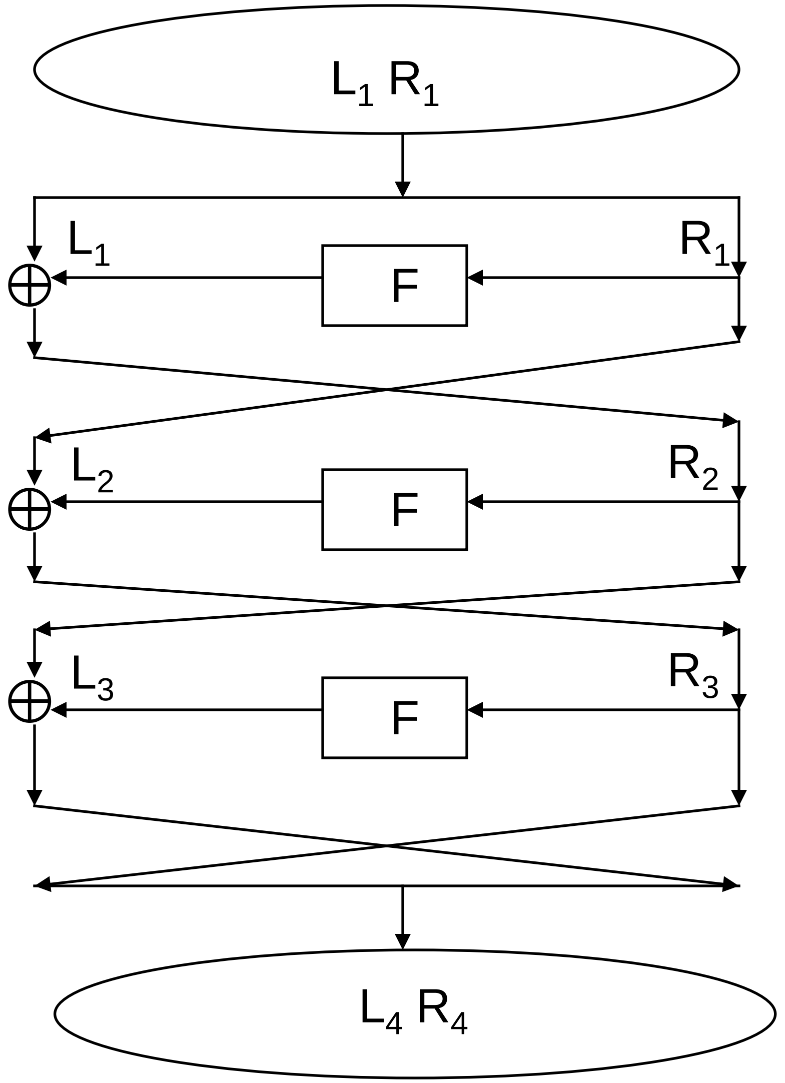
S_1': 1010 $\rightarrow$ 010(1001,0011).

S_2': 1011 $\rightarrow$ 100(1100,0111).

$(E(L_4)\oplus k_4)_{1..4}$=1001|0011, k_4= 1001|0011.

$(E(L_4)\oplus k_4)_{5..8}$= 1100|0111,$k_4$= 1111|0100.

K= 00x001101



Differential Cryptanalysis 4R

Pick

`L`$_0$`’, R`$_0$`’: 011010 001100.`

Then

`E(R`$_0$`’):   0011 1100.`

`0011 → 011 with p=3/4`

`1100 → 010 with p=1/2`

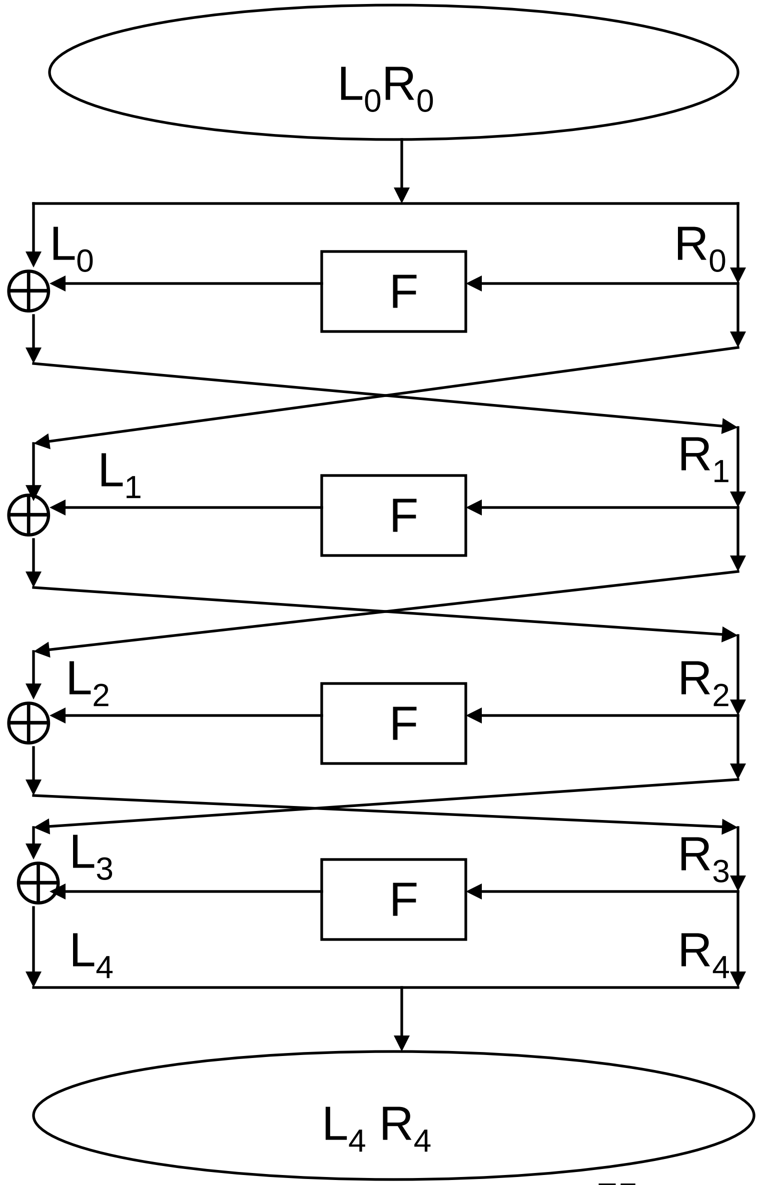
So

`f(R`$_0$`’, k`$_1$`)= 011 010, p=3/8.`

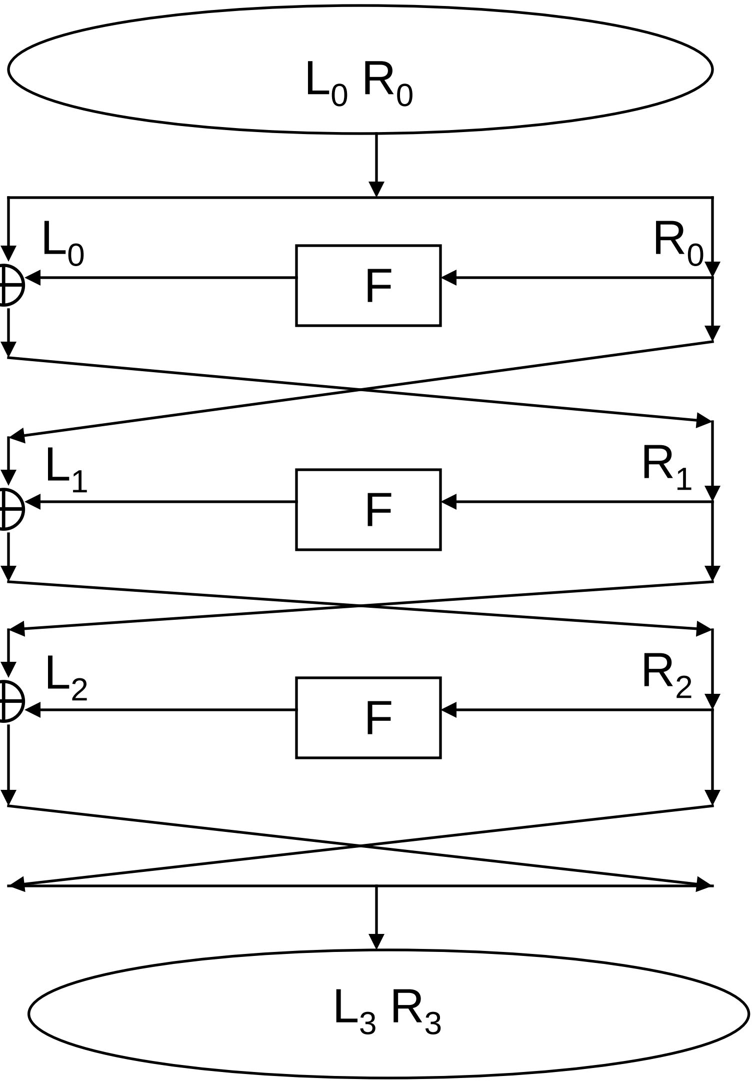
Thus

`L`$_1$`’, R`$_1$`’: 001100 000000, p=3/8.`

- 3/8 of the pairs with this differential produce this result. 5/8 scatter the output differential at random. These “vote” for 1100 and 0010.



Linear cryptanalysis - 3R (Simple DES)



- Denote $L_i = (l_1,l_2,l_3,l_4,l_5,l_6)^{(i)}$ and $R_i = (r_1,r_2,r_3,r_4,r_5,r_6)^{(i)}$ sometimes we'll drop the (i) superscript, $K_i=(k_1,k_2,\ldots,k_8)^{(i)}$ where the k_i are from the key for round I and finally, $K=(k_1,k_2,\ldots,k_9)$, where K is the master key.
- By doing the Walsh transform, we learn that $f_2(t_1,t_2,t_3,t_4)= t_1+t_3+t_4$, with p= 7/8, and $f_4(t_1,t_2,t_3,t_4)= t_3+t_4+1$, with p= 7/8.
- Note that $E(t_1,t_2,t_3,t_4,t_5,t_6)= (t_1,t_2,t_4,t_3,t_4,t_3,t_5,t_6)$.

- $f_2(E(R_0)+K_1) = f_2((r_1,r_2,r_4,r_3,r_4,r_3,r_5,r_6)^{(0)} +(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8)^{(1)}) = R_0[1,3,4]+K_1[1,3,4]$ with p=7/8.
- $f_4(E(R_0)+K_1) = f_2((r_1,r_2,r_4,r_3,r_4,r_3,r_5,r_6)^{(0)} +(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8)^{(1)}) = R_0[5,6]+K_1[7,8] +1$ with p=7/8.

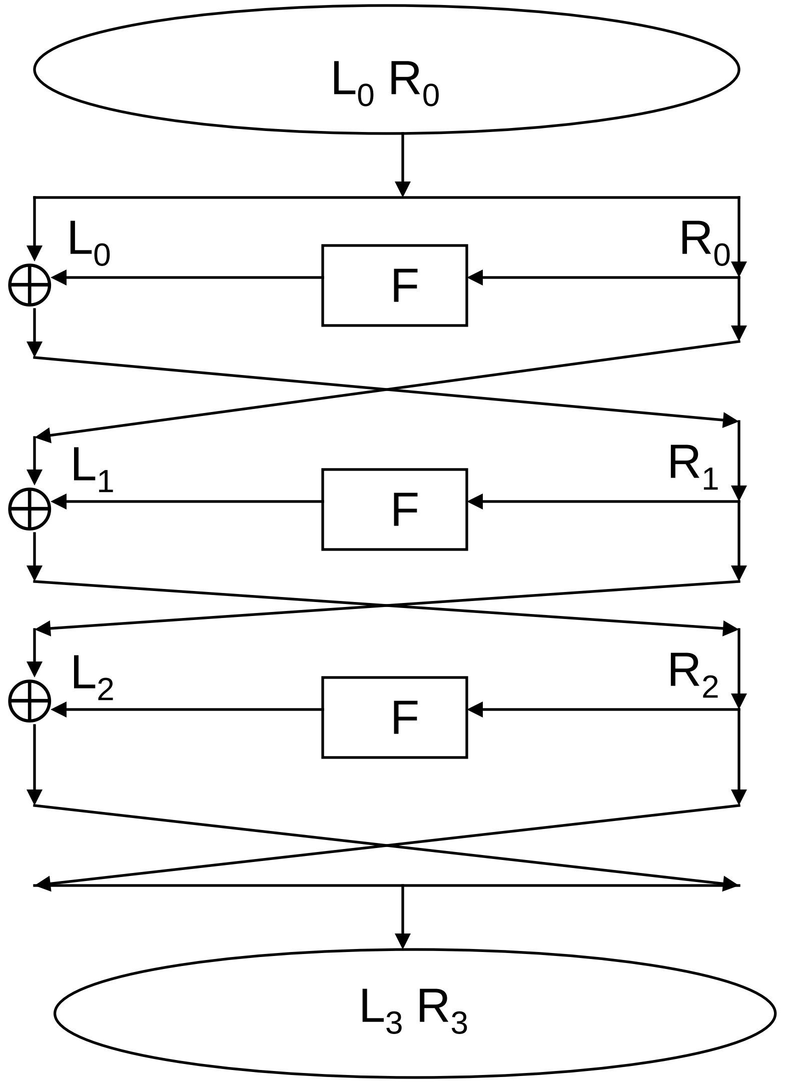
The per round constraints

- So we get
  - $R_i[2]= L_{i-1}[2]+R_{i-1}[1,3,4]+K_i[1,3,4]$
  - $R_i[4]= L_{i-1}[4]+R_{i-1}[5,6]+K_i[7,8]+1$

Writing all 3 round equations out:

1. $R_1[2]= L_0[2]+R_0[1,3,4]+K_1[1,3,4]$
2. $R_1[4]= L_0[4]+R_0[5,6]+K_1[7,8]+1$
3. $R_2[2]= L_1[2]+R_1[1,3,4]+K_2[1,3,4]$
4. $R_2[4]= L_1[4]+R_1[5,6]+K_2[7,8]+1$
5. $R_3[2]= L_2[2]+R_2[1,3,4]+K_3[1,3,4]$
6. $R_3[4]= L_2[4]+R_2[5,6]+K_3[7,8]+1$

- Note $L_1= R_0$, $L_2= R_1$, $L_3=R_2$.



Linear cryptanalysis - the payoff

- Substituting 1→5 and noting $L_2=R_1$,we get
- $R_3[2]= L_0[2]+R_0[1,3,4]+K_1[1,3,4] +R_2[1,3,4]+K_3[1,3,4]$ with $p=(7/8)^2+(1/8)^2$.
- Substituting 2→6 and noting $L_2=R_1$,we get
- $R_3[4]= L_0[4]+R_0[5,6]+K_1[7,8]+R_2[5,6]+K_3[7,8]$ with $p=(7/8)^2+(1/8)^2$.
- Finally, noting
  - $L_3=R_2$ and
  - $K_1[1,3,4]= K [1,3,4]$, $K_1[7,8]= K [7,8]$
  - $K_3[1,3,4]= K [3,5,6]$, $K_3[7,8]= K [1,9]$

1. $R_3[2]+ L_0[2]+R_0[1,3,4]+L_3[1,3,4]= K [1,3,4] + K_3[3,5,6]$ with $p=(7/8)^2+(1/8)^2$.
2. $R_3[4]+L_0[4]+R_0[5,6] +L_3 [5,6] =K [7,8]+K [1,9]$ with $p=(7/8)^2+(1/8)^2$.

- All terms on the left of 1 and 2 are known, so we get 2 linear constraints that hold with probability .78125 each constraining the key bits.

End