

COMSM2004 : Advanced Block Ciphers

B. Warinschi and N.P. Smart

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB
United Kingdom.

January 30, 2009

Outline

Overview

Differential Cryptanalysis

Linear Cryptanalysis

Design Of Rijndael

Advanced Block Ciphers

In this section of the course we consider how block ciphers are designed.

We focus on the main modern attack on block ciphers

- ▶ **Differential Cryptanalysis**

We use a simple example from a tutorial on the net by **Howard Heys**

We assume that the design of Rijndael and DES are understood

- ▶ We want to learn why they are, as they are.

Differential Cryptanalysis Basics

Differential cryptanalysis is a chosen plaintext attack.

The attacker computes lots of **pairs** of plaintexts

- ▶ (m_1, m_2)

and computes

- ▶ $(c_1, c_2) = (E_k(m_1), E_k(m_2))$

The goal is to determine the key **k**

The attacker computes

- ▶ $\Delta m = m_1 \oplus m_2$

- ▶ $\Delta c = c_1 \oplus c_2$

The pair

- ▶ $(\Delta m, \Delta c)$

is called a **differential**

Differential Cryptanalysis Basics

By careful

- ▶ Analysis of the cipher under attack
- ▶ Choice of the plaintexts/input differences
- ▶ Some statistical analysis

One tries to recover the key

In analysing the algorithm need to see how the differences **propagate** through the various stages

- ▶ By “stage” we mean a single subset of a given round

In the following

- ▶ Let s_1, s_2 denote the input to a “stage”
- ▶ Let t_1, t_2 denote the outputs from a “stage”
- ▶ Let $\Delta s = s_1 \oplus s_2, \Delta t = t_1 \oplus t_2$

Block Cipher Stages

A block cipher round usually consists of a number of stages

- ▶ Round key addition
- ▶ Permutation of bits
- ▶ Mixing Layer
- ▶ Substitution

We need to consider the propagation of differences through these stages

Round Key Addition

For round key addition we have

$$\blacktriangleright t_j = s_j \oplus k'$$

Hence,

$$\begin{aligned}\Delta t &= t_1 \oplus t_2 \\ &= (s_1 \oplus k') \oplus (s_2 \oplus k') \\ &= s_1 \oplus s_2 \\ &= \Delta s\end{aligned}$$

Hence, round key addition **does not affect** differences

Permutation Stage

Bit permutation is used in DES

- ▶ In Rijndael this is the Shift Row stage

Clearly given an input difference to a permutation stage we can compute the output difference

- ▶ Irrespective of the actual data being computed on

The DES expansion permutation clearly makes differences avalanche

- ▶ Combined with special choice of S-boxes and the exact expansion permutation this helps make DES **immune** from differential cryptanalysis

Mixing Layer

Consider the Rijndael mixing layer (MixColumn) This is a matrix multiplication

- ▶ $t_i = Ms_i$

When we consider the operation on differences we have

$$\begin{aligned}\Delta t &= t_1 \oplus t_2 \\ &= (Ms_1) \oplus (Ms_2) \\ &= M(s_1 \oplus s_2) \\ &= M(\Delta s)\end{aligned}$$

Hence, given in input difference we can compute the output difference

- ▶ Rijndaels matrix M is defined to make the differences avalanche
- ▶ See later for more details

Substitution Stage

This is where the complexity comes in and we need to look at the specific S-box under attack.

- ▶ In DES the S-boxes are designed to make differential cryptanalysis hard
- ▶ In Rijndael the $1/x$ operation in the S-box is almost “best-possible” to avoid differential cryptanalysis
 - ▶ See later for more details

We will explain this with a toy example

S-Box Example

Consider the following 4×4 S-box

▶ i.e. maps four bit inputs to four bit outputs
given by (in Hex)

$0 \longrightarrow E$	$1 \longrightarrow 4$	$2 \longrightarrow D$	$3 \longrightarrow 1$
$4 \longrightarrow 2$	$5 \longrightarrow F$	$6 \longrightarrow B$	$7 \longrightarrow 8$
$8 \longrightarrow 3$	$9 \longrightarrow A$	$A \longrightarrow 6$	$B \longrightarrow C$
$C \longrightarrow 5$	$D \longrightarrow 9$	$E \longrightarrow 0$	$F \longrightarrow 7$

We shall call this S-Box \mathbb{S} and write

▶ $t_j = \mathbb{S}(s_j)$

S-Box Example

Suppose

- ▶ $s_1 = 3 = 0b0011$ and $s_2 = 6 = 0b0110$

then

- ▶ $\Delta s = s_1 \oplus s_2 = 5 = 0b0101$

With our example S-Box we have

- ▶ $t_1 = \mathbb{S}(s_1) = 1 = 0b0001$ and $t_2 = \mathbb{S}(s_2) = B = 0b1011$

then

- ▶ $\Delta t = t_1 \oplus t_2 = A = 0b1010$

We repeat this for all $256 = 16 \times 16$ pairs of inputs (s_1, s_2) and compute the corresponding Δs and Δt . The following page gives the resulting frequency table

- ▶ Inputs differences given by rows, Outputs by columns

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	2	0	0	0	2	0	2	4	0	4	2	0	0
2	0	0	0	2	0	6	2	2	0	2	0	0	0	0	2	0
3	0	0	2	0	2	0	0	0	0	4	2	0	2	0	0	4
4	0	0	0	2	0	0	6	0	0	2	0	4	2	0	0	0
5	0	4	0	0	0	2	2	0	0	0	4	0	2	0	0	2
6	0	0	0	4	0	4	0	0	0	0	0	0	2	2	2	2
7	0	0	2	2	2	0	2	0	0	2	2	0	0	0	0	4
8	0	0	0	0	0	0	2	2	0	0	0	4	0	4	2	2
9	0	2	0	0	2	0	0	4	2	0	2	2	2	0	0	0
A	0	2	2	0	0	0	0	0	6	0	0	2	0	0	4	0
B	0	0	8	0	0	2	0	2	0	0	0	0	0	2	0	2
C	0	2	0	0	2	2	2	0	0	0	0	2	0	6	0	0
D	0	4	0	0	0	0	0	4	2	0	2	0	2	0	2	0
E	0	0	2	4	2	0	0	0	6	0	0	0	0	0	2	0
F	0	2	0	0	6	0	0	0	0	4	0	2	0	0	2	0

S-Box Example

If we ignore the 16 values where $s_1 = s_2$ then

- ▶ There are $225 = 15 \times 15$ values of (s_1, s_2)
- ▶ There are $225 = 15 \times 15$ values of $(\Delta s, \Delta t)$

Hence, if we had a **perfect** S-Box,

- ▶ i.e. one with no statistical bias,

we would have a frequency table with **one entry** in each main entry

For each input difference expect each output difference to be equally likely

- ▶ Such perfect S-Box's are impossible to create

At least would like a more **uniform** distribution in the prior table

Toy Example

We aim to use the prior table in analysing a cipher

- ▶ Use a “toy” example due to H. Heys

The cipher is an SP-Network

- ▶ A Substitution-Permutation network
- ▶ Repeated application of S-Boxes and Permutations
 - ▶ A bit like Rijndael

16-bit block size

- ▶ Clearly too small

Four rounds

- ▶ Clearly too small

Five 16-bit round keys

- ▶ Clearly too small
- ▶ We will treat this as a $5 \cdot 16 = 80$ bit single key

Permutation

The permutation part of our cipher takes a 16 bit value and produces another 16 bit value

The permutation is given by

1	→	1	2	→	5	3	→	9	4	→	13
5	→	2	6	→	6	7	→	10	8	→	14
9	→	3	10	→	7	11	→	11	12	→	15
13	→	4	14	→	8	15	→	12	16	→	16

i.e. bit 3 becomes bit 9 and bit 10 becomes bit 7 etc

Call this operation \mathbb{P} , i.e. $t_i = \mathbb{P}(s_i)$

Toy Cipher

Each round consists of the following three stages operating on the 16-bit state S

- ▶ Add round key k_j
 - ▶ $S = S \oplus k_j$
- ▶ Split S into its four constitute 4-bit nibbles
 - ▶ $S = (s_1 || s_2 || s_2 || s_3)$
- ▶ Apply the S-Box \mathbb{S} to the nibbles
 - ▶ $t_i = \mathbb{S}(s_i)$ for $i = 1, 2, 3, 4$
- ▶ Create the new state
 - ▶ $S = (t_1 || t_2 || t_3 || t_4)$
- ▶ Apply the Permutation P
 - ▶ $S = \mathbb{P}(S)$

Toy Cipher

The cipher consists of four rounds as described followed by a final fifth round key addition

- ▶ The fourth round does not have the permutation
- ▶ See code on web site for an implementation

```
block Encrypt(block m,block k[5])
{ for (int i=0; i<3; i++)
    { m^=k[i];
      m=SBox_Layer(m);
      m=P_Layer(m); // End of round i+1
    }
  m=m^k[3];
  m=SBox_Layer(m); // End of round 4
  m=m^k[4]; // Additional key addition
  return m;
}
```

Analysis - I

Look for entries in frequency table for the S-Box with high probability of giving a given output difference

For example

- ▶ If $\Delta s = B$ then $\Delta t = 2$ with probability $8/16$
- ▶ High probability due to non uniform nature of previous table

Suppose we assume that we perform the following attack

- ▶ Compute lots of plaintext pairs (m_1, m_2) such that

$$\Delta m = m_1 \oplus m_2 = 0x0B00$$

- ▶ We let S_i denote the state difference after the i 'th round

We then have that with probability $8/16$ that

- ▶ $S_1 = 0x0040$

Analysis - II

We now look in the frequency table for the input difference 4

- ▶ Find output 6 is likely with probability $6/16$

Tracing this through the permutation we find

- ▶ $S_2 = 0x0220$

with probability $(8 \cdot 6)/256 = 48/256 = 0.1875$

We now look in the frequency table for the input difference 2

- ▶ Find output 5 is likely with probability $6/16$

Tracing this through the permutation we find

- ▶ $S_3 = 0x0606$

with probability $(8 \cdot 6^3)/65536 = 1728/65536 \approx 0.02637$

Analysis - III

For the plaintext pairs we have computed we call

- ▶ “Good pairs” those such that $S_3 = 0x0606$
- ▶ “Bad pairs” those such that $S_3 \neq 0x0606$

Don't yet know which ones are good and which ones are bad

- ▶ Probability of good pair is 0.02637

Analysis - IV

Now try to compute S_3 by unwinding the encryption algorithm backwards

- ▶ Given $\Delta c = c_1 \oplus c_2 = E_k(m_1) \oplus E_k(m_2)$

Only interested in those pairs such that

- ▶ First and third nibble of S_3 are equal to zero
- ▶ Reject all values $((m_1, m_2), (c_1, c_2))$ which cannot be of form

$$S_3 = 0x0 * 0*$$

- ▶ This means we must have

$$\Delta c = 0x0 * 0*$$

The values left are highly likely to correspond to “good pairs”

Analysis - V

Let

- ▶ $S'_3 = S_3 \oplus k_3$

i.e. S'_3 is the value input into the last S-Box stage

Let $k_{i,j}$ denote the j th nibble in the $i + 1$ 'st round key

- ▶ i.e. $k_i = (k_{i,1} || k_{i,2} || k_{i,3} || k_{i,4})$

We are going to compute the value of $k_{4,2}$ and $k_{4,4}$

- ▶ There are $16 \times 16 = 256$ such values

Write

- ▶ $S'_3 = (U_1 || U_2 || U_3 || U_4)$

Analysis - VI

For each possible value of the partial subkey

- ▶ $k_{4,2}$ and $k_{4,4}$

set a counter equal to zero

For each tuple (m_1, m_2, c_1, c_2) compute ΔU_2 and ΔU_4 by

- ▶ Rewinding algorithm and using the guess $k_{4,2}$ and $k_{4,4}$
- ▶ Increase the counter if $\Delta U_2 = 0x6$ and $\Delta U_4 = 0x6$

The counter with the highest value corresponds to the most likely partial subkey

Analysis - VII

This final stage works since, considering only the second nibble,

If we have a “good pair”

- ▶ $\Delta U_2 = 0x6$

and suppose

- ▶ c_1 has second nibble equal to $0x3$

- ▶ c_2 has second nibble equal to $0x4$

Suppose we “think” $k_{4,2} = 0x2$

Then output from last S-Box on second nibble is

- ▶ $0x3 \oplus 0x2 = 0x1$ and $0x4 \oplus 0x2 = 0x6$

Which means the input to the S-Box must be

- ▶ $0x3$ and $0xA$ since $0x1 = \mathbb{S}(0x3)$ and $0x6 = \mathbb{S}(0xA)$

Analysis - VIII

This would imply

- ▶ $\Delta U_2 = 0x3 \oplus 0xA = 0x9$

This is not equal to $0x6$ hence the guess for the key is wrong

- ▶ Assuming we have a “good pair”

Since “good pairs” are likely with high probability, namely **0.02637**

- ▶ Expect correct key choice to give a high count value
- ▶ Essentially for correct key guess and a good pair we add one to the counter
- ▶ For incorrect key guess will only add to counter with low probability

Attack Summary

Collect lots of (m_1, m_2, c_1, c_2) , say 5000

Reject those which are unlikely to correspond to “good pairs”

- ▶ Expect at least 27 percent to remain

Determine most likely value for key nibbles $k_{4,2}$ and $k_{4,4}$

After having determined these key nibbles we then use other differentials to compute $k_{4,1}$ and $k_{4,3}$ and hence all of k_4

Then repeat to recover k_4 , then k_3 , and so on

In general if R rounds we trace the differences through $R - 1$ rounds and then rewind the last round using the ciphertexts corresponding to the chosen plaintexts

Complexity

Complexity measured by number of plaintext pairs needed

If p is the probability of a “good pair” then require (roughly)

$$N = (100 \cdot c)/p$$

plaintext pairs

- ▶ Where c is a “small” constant

Need to repeat this for each subkey

In our example $p \approx 0.02637$ to determine two key nibbles

- ▶ Could take $N = 1000$

Example

Did this for our toy example

- ▶ Took $N = 1000$ plaintext pairs

Found 70 (i.e. 7 percent) produced ciphertexts such that

- ▶ $\Delta c = 0x0 * 0*$

Then rewound algorithm for these 70 and found the subkey by matching the differences

- ▶ Of the 70 the correct subkey occurred in 28 of these instances
- ▶ i.e. of our 1000 plaintexts, 28 where “good pairs”

Pretty close to theoretical predication

Immunising Ciphers

Rijndael is a cipher which has been designed to be immune from differential cryptanalysis

Main strategies

- ▶ Try to minimise the difference pair probability of an S-Box
 - ▶ i.e. make the frequency table we had almost uniform
- ▶ For all small input differences try to maximise the number of active S-Box's
- ▶ Increase number of rounds to reduce the difference probabilities

Immunising Ciphers

In Rijndael

- ▶ The first strategy gives rise to the $1/x$ operation in the S-Box
- ▶ The second strategy gives rise to the MixColumn stage
 - ▶ We give more detail later on

The number of rounds is chosen to balance efficiency against making the difference probabilities small enough to withstand attacks

Linear Cryptanalysis

We now turn to Linear Cryptanalysis (again we follow the tutorial of Heys)

Here one tries to approximate the non-linear components by linear functions

- ▶ Mainly applied to S-Boxes
- ▶ **Known** plaintext attack
 - ▶ Differential Cryptanalysis was a **chosen** plaintext attack

Linear Cryptanalysis

Linear Cryptanalysis is based around the following concept

Let X be a random binary variable, with

- ▶ $\Pr[X = 0] = p$
- ▶ $\Pr[X = 1] = 1 - p$

The **bias** of X is defined to be

- ▶ $\epsilon = p - \frac{1}{2}$

The idea is to examine the bias of linear representations of the non-linear components

Piling Up Principle

If X_i are independent random binary variables with bias ϵ_i and

- ▶ $\Pr[X_i = 0] = p_i = \frac{1}{2} + \epsilon_i$

Then

$$\Pr[X_1 \oplus \dots \oplus X_n = 0] = \frac{1}{2} + 2^{n-1} \prod_{i=1}^n \epsilon_i$$

i.e. the **bias** of $X_1 \oplus \dots \oplus X_n$ is

$$2^{n-1} \prod_{i=1}^n \epsilon_i$$

Note:

- ▶ If one variable has zero bias then so does $X_1 \oplus \dots \oplus X_n$

Linear Approximation to S-Boxes

Suppose our S-Box has n input bits X_i and m -output bits Y_j . Let

- ▶ $I = \{i_1, \dots, i_u\} \subset \{1, \dots, n\}$
- ▶ $J = \{j_1, \dots, j_v\} \subset \{1, \dots, m\}$

Consider the linear function

$$L_{I,J} = (X_{i_1} \oplus \dots \oplus X_{i_u}) \oplus (Y_{j_1} \oplus \dots \oplus Y_{j_v})$$

Applying all 2^n values for the input X and examining the resulting Y s

- ▶ Determine $\Pr[L_{I,J} = 0]$ and its bias

If S-Box is “good” the bias should be close to zero

Linear Approximation to S-Boxes

Take our toy S-box \mathbb{S}

Has four input and four output bits we find (amongst others)

Function L	$Pr[L = 0]$	Bias
$L_1 = X_1 \oplus X_3 \oplus X_4 \oplus Y_2$	$\frac{3}{4}$	$\frac{1}{4}$
$L_2 = X_2 \oplus Y_2 \oplus Y_4$	$\frac{1}{4}$	$-\frac{1}{4}$
$L_3 = X_1 \oplus X_4 \oplus Y_2$	$\frac{1}{2}$	0
$L_4 = X_3 \oplus X_4 \oplus Y_1 \oplus Y_4$	$\frac{1}{8}$	$-\frac{3}{8}$

We will use L_1 and L_2 in our following analysis

We assume that linear approximations to S-Boxes are independent

- ▶ So we can apply Piling Up Principle
- ▶ Not strictly true but often OK in practice

Notation

To analyse the cipher we have to “chase” bits through the rounds

- ▶ Similar to what done in Differential Cryptanalysis

Let U_i denote the 16-bit input to the S-Box stage in Round i

Let V_i denote the 16-bit output of the S-Box stage in Round j

Label bits of U_i and V_i by $U_{i,j}$ and $V_{i,j}$

- ▶ $1 \leq j \leq 16$
- ▶ Bit 1 is the one to the left (not usual convention)

Round keys labelled as K_i

Key bits labelled by $K_{i,j}$

- ▶ $j = 1$ bit is on the left
- ▶ Note, different notation from earlier when we were nibble oriented

Let the plaintext be P and the ciphertext be C

Analysis - I

We will approximate

- ▶ The second S-Box in round one with L_1
- ▶ The second S-Box in rounds two and three with L_2
- ▶ The fourth S-Box in round three with L_2

Since assuming second S-Box in round one is approximated by L_1 we have

$$V_{1,6} = U_{1,5} \oplus U_{1,7} \oplus U_{1,8}$$

with probability $3/4$

Since $U_1 = P \oplus K_0$ this becomes simplifies to

$$I_1 = V_{1,6} \oplus (P_5 \oplus P_7 \oplus P_8) \oplus (K_{0,5} \oplus K_{0,7} \oplus K_{0,8}) = 0$$

Analysis - II

Due to the permutation stage, namely $\mathbb{P}(6) = 6$, we have that

$$\blacktriangleright U_{2,6} = V_{1,6} \oplus K_{1,6}$$

Due to approximating the second S-box in round two by L_2 we have

$$V_{2,6} \oplus V_{2,8} = U_{2,6}$$

with probability $1/4$

Hence, also with probability $1/4$ we have

$$I_2 = V_{2,6} \oplus V_{2,8} \oplus V_{1,6} \oplus K_{1,6} = 0$$

Analysis - III

By the **Piling Up Principle**

$$\Pr[l_1 \oplus l_2 = 0] = 3/8$$

Set

$$\begin{aligned} l_3 &= l_1 \oplus l_2 \\ &= V_{2,6} \oplus V_{2,8} \oplus (P_5 \oplus P_7 \oplus P_8) \oplus (K_{0,5} \oplus K_{0,7} \oplus K_{0,8} \oplus K_{1,6}) \end{aligned}$$

Notice l_3 does not depend on any $U_{1,i}$ or $V_{1,j}$

Analysis - IV

Need to trace $V_{2,6}$ and $V_{2,8}$ into round three.

Since $\mathbb{P}(6) = 6$ and $\mathbb{P}(8) = 14$ we have

$$\blacktriangleright U_{3,6} = V_{2,6} \oplus K_{2,6}$$

$$\blacktriangleright U_{3,14} = V_{2,8} \oplus K_{2,14}$$

Recall we are approximating the second and fourth S-Boxes in Round 3 with L_2

Hence, the following two equations hold each with probability $1/4$

$$\blacktriangleright V_{3,6} \oplus V_{3,8} = U_{3,6}$$

$$\blacktriangleright V_{3,14} \oplus V_{3,16} = U_{3,14}$$

Combining these facts together we obtain by the **Piling Up Principle**

$$\Pr[l_4 = 0] = 5/8$$

where

$$l_4 = (V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16} \oplus V_{2,6} \oplus V_{2,8}) \\ \oplus (K_{2,6} \oplus K_{2,14})$$

Analysis - VI

Combining $l_3 = 0$ and $l_4 = 0$ we obtain, via the **Piling Up Principle**

$$\Pr[l_3 \oplus l_4 = 0] = 15/32$$

Set

$$\begin{aligned} l_5 &= l_3 \oplus l_4 \\ &= (V_{3,6} \oplus V_{3,8} \oplus V_{3,14} \oplus V_{3,16}) \\ &\quad \oplus (P_5 \oplus P_7 \oplus P_8) \\ &\quad \oplus (K_{0,5} \oplus K_{0,7} \oplus K_{0,8} \oplus K_{1,6} \oplus K_{2,6} \oplus K_{2,14}) \end{aligned}$$

Notice l_5 does not depend on any $U_{1,i}$, $U_{2,i}$, $V_{1,j}$ and $V_{2,j}$

Analysis - VII

Now need to trace $V_{3,6}$, $V_{3,8}$, $V_{3,14}$ and $V_{3,16}$

We have $\mathbb{P}(6) = 6$, $\mathbb{P}(8) = 14$, $\mathbb{P}(14) = 8$ and $\mathbb{P}(16) = 16$

Hence,

- ▶ $U_{4,6} = V_{3,6} \oplus K_{3,6}$
- ▶ $U_{4,14} = V_{3,8} \oplus K_{3,14}$
- ▶ $U_{4,8} = V_{3,14} \oplus K_{3,8}$
- ▶ $U_{4,16} = V_{3,16} \oplus K_{3,16}$

Analysis - VIII

Plugging these into l_5 we obtain

$$l_6 = (U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16}) \oplus (P_5 \oplus P_7 \oplus P_8) \oplus \sigma_K$$

where

$$\sigma_K = K_{0,5} \oplus K_{0,7} \oplus K_{0,8} \oplus K_{1,6} \oplus K_{2,6} \oplus K_{2,14} \oplus K_{3,6} \oplus K_{3,8} \oplus K_{3,14} \oplus K_{3,16}$$

and

► $\Pr[l_6 = 0] = 15/32$

Analysis - IX

We don't actually care what σ_K is equal to

- ▶ We simply note that for any fixed key, either $\sigma_K = 0$ or $\sigma_K = 1$

Hence, if we set

$$I_7 = (U_{4,6} \oplus U_{4,8} \oplus U_{4,14} \oplus U_{4,16}) \oplus (P_5 \oplus P_7 \oplus P_8)$$

Then

- ▶ $\Pr[I_7 = 0] = 15/32$ or $(1 - 15/32) = 17/32$

Note

- ▶ I_7 involves plaintext bits and inputs to the fourth round S-Boxes

Analysis - X

Now able to determine some of the key

Compute all 256 possible values for the second and fourth nibbles of the final (post-whitening) round key

- ▶ i.e. $K_{4,5}, K_{4,6}, K_{4,7}, K_{4,8}, K_{4,13}, K_{4,14}, K_{4,15}, K_{4,16}$

For each ciphertext/plaintext pair rewind the final round and invert the final S-Box (only on second and fourth nibble)

- ▶ Allows us to compute $U_{4,6}, U_{4,8}, U_{4,14}, U_{4,16}$
- ▶ Check whether equation l_7 holds

For the correct choice of fifth round subkey l_7 should hold over all ciphertext/plaintext pairs with probability $15/32$ or $17/32$ (as opposed to $16/32$) for non-correct choices

Complexity

Complexity measured by number of known plaintext/ciphertext pairs needed

If ϵ is the bias of the linear expression for the whole cipher

- ▶ Our h_7 expression

Number of known plaintexts needed is roughly

$$N \approx 1/\epsilon^2$$

In our example $\epsilon \approx 0.03125$ to determine two key nibbles

- ▶ Could take $N = 1000$

Example

Did this for our toy example

- ▶ Took $N = 10000$ plaintexts

Found correct key

- ▶ Note very unstable
- ▶ Sometimes outputs the wrong value
 - ▶ Depends on rest of key bits
 - ▶ Correct value output if far more plaintexts are taken

Immunising Ciphers

The final bias is dependent on

- ▶ Bias in the linear approximation to the S-Boxes
- ▶ Number of active S-Boxes used in the attack (we used four)

Rijndael

- ▶ Minimizes the bias in the S-Box (via the x^{-1} operation)
- ▶ Maximises the number of active S-Boxes (via the MixColumn operation)
 - ▶ We give more detail later on

Rijndael - Recap

Block + key length: varies between **128 - 256** bits (per 64 bits).

Number of **rounds** is **10/12/14** depending on block and key length.

Uniform and parallel **round transformation**, composed of:

- ▶ **Byte substitution.**
- ▶ **Shift rows.**
- ▶ **Mix columns.**
- ▶ **Round key addition.**

AES uses key and block size of 128 bits and ten rounds

Rijndael - Data Representation

Plaintext block normally is 128 bits or **16 bytes** m_0, \dots, m_{15} .

Key normally is 128/192/256 bits or **16/24/32 bytes** k_0, \dots, k_{31} .

Both are represented as **rectangular array of bytes**.

$$\begin{pmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{pmatrix} \quad \begin{pmatrix} k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ k_8 & k_9 & k_{10} & k_{11} \\ k_{12} & k_{13} & k_{14} & k_{15} \end{pmatrix}$$

Rijndael - Byte Substitution

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \longrightarrow \begin{pmatrix} S(s_{0,0}) & S(s_{0,1}) & S(s_{0,2}) & S(s_{0,3}) \\ S(s_{1,0}) & S(s_{1,1}) & S(s_{1,2}) & S(s_{1,3}) \\ S(s_{2,0}) & S(s_{2,1}) & S(s_{2,2}) & S(s_{2,3}) \\ S(s_{3,0}) & S(s_{3,1}) & S(s_{3,2}) & S(s_{3,3}) \end{pmatrix}$$

State matrix is transformed **byte by byte**. **S-box is invertible**, else decryption would not work. Only **1 S-box** for the whole cipher (simplicity).

Rijndael - Shift Rows

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix} \longrightarrow \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{pmatrix}$$

Rows shifted over different offsets (depending on block length).

Purpose: **diffusion over the columns.**

Rijndael - Mix Columns

$$\begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3} \end{pmatrix} \longrightarrow$$

$$\begin{pmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{pmatrix} \cdot \begin{pmatrix} S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\ S_{1,1} & S_{1,2} & S_{1,3} & S_{1,0} \\ S_{2,2} & S_{2,3} & S_{2,0} & S_{2,1} \\ S_{3,3} & S_{3,0} & S_{3,1} & S_{3,2} \end{pmatrix}$$

- ▶ Operations over **finite field $GF(2^8)$** .
- ▶ Bytes in columns are combined linearly.
- ▶ Good **diffusion** properties **over rows**.
- ▶ Based on **maximal distance codes**.

Rijndael - Round Key Addition

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \longrightarrow$$

$$\begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,1} & k_{1,2} & k_{1,3} & k_{1,0} \\ k_{2,2} & k_{2,3} & k_{2,0} & k_{2,1} \\ k_{3,3} & k_{3,0} & k_{3,1} & k_{3,2} \end{pmatrix} \oplus \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}$$

Round key is simply XOR-ed with state matrix.

Rijndael - Pseudo-code

Rijndael with 10 rounds is described by the following code:

```
AddRoundKey (S, K[0]);
for (i = 1; i <= 9; i++)
    {
        SubBytes (S);
        ShiftRows (S);
        MixColumns (S);
        AddRoundKey (S, K[i]);
    }
SubBytes (S);
ShiftRows (S);
AddRoundKey (S, K[10]);
}
```

Rijndael - More Details

Each byte of the Rijndael state is considered to be an element of the finite field of degree 8

- ▶ Defining polynomial $f = x^8 + x^4 + x^3 + x + 1$

ByteSub consists of two stages

- ▶ Inversion in the finite field
- ▶ Linear map over \mathbb{F}_2

Often write this as

$$x \longrightarrow Ax^{-1} + c$$

although this does not make mathematical sense

Rijndael - ByteSub

Given a byte s one computes $x = s^{-1}$

- ▶ Thinking of $s \in \mathbb{F}_{2^8}$
- ▶ The zero byte gets mapped to the zero byte by convention

Then treat t as an 8-bit bit-vector and compute

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Output of ByteSub is the byte y

Rijndael - ByteSub

The effect of the linear transform is to remove fixed points,

- ▶ $S(a) \oplus a \neq 0x00$

for all a

It also removes all opposite fixed points

- ▶ $S(a) \oplus a \neq 0xFF$

although this has no known cryptographic justification

Inverse of SubByte performed by

- ▶ Inverting the linear \mathbb{F}_2 transform
 - ▶ Can do since matrix used is invertible modulo 2

Then perform x^{-1} again

Rijndael - ByteSub

Why the use of $x \longrightarrow x^{-1}$?

Firstly:

- ▶ ByteSub is the only non-linear component of Rijndael
- ▶ The only non-linear component of ByteSub is the map $x \longrightarrow x^{-1}$

Secondly:

- ▶ The map $x \longrightarrow x^{-1}$ is a good protection against differential cryptanalysis

Non-Linear Goal

Recall that we want a non-linear operation which produces a difference table (as earlier) which is almost uniform

Let

- ▶ The input difference be $\Delta x \neq 0$
- ▶ The output difference be Δy

Let f denote the non-linear function we aim to use

Goal : Want

$$\Delta y = f(x) \oplus f(x + \Delta x)$$

to have **roughly** the same number of solutions x

- ▶ Irrespective of choice of $\Delta x \neq 0$ and Δy

In our Toy Cipher this number ranged from 0 to 8 (out of a maximum of 16)

Non-Linear Goal

Suppose we take $f(x) = 1/x$ then we obtain the equation

$$\Delta y = \frac{1}{x} \oplus \frac{1}{x + \Delta x}$$

Suppose $x \neq 0$ and $x + \Delta x \neq 0$ Then we obtain

$$(\Delta y)x^2 + (\Delta y \cdot \Delta x)x + (\Delta x) = 0$$

hence we obtain at most two solutions in x

Now if $x = 0$ or $x + \Delta x = 0$ then $\Delta x = (\Delta y)^{-1}$

Hence number of solutions is at most

- ▶ 2 if $\Delta x \neq (\Delta y)^{-1}$
- ▶ 4 if $\Delta x = (\Delta y)^{-1}$

Difference Table - I

In the case of Rijndael we have a field of size $2^8 = 256$

Considering the difference table for the operation $f(x) = 1/x$ Obtain a table of 256 rows where each row has 256 entries such that

- ▶ Number of entries in each row adds up to 256
- ▶ Number of entries in each column adds up to 256
- ▶ Max number of entries in each cell is two
 - ▶ Except for
 - ▶ One cell which has at most four entries, when $\Delta x = (\Delta y)^{-1}$
 - ▶ One cell which has 256 entries, i.e. when $\Delta x = \Delta y = 0$

Such a table can be considered to be pretty uniformly distributed

Maximum difference propagation probability is $4/256 = 1/64$

Effect of Linear Operation Since the linear mapping (done after the $x \rightarrow 1/x$ operation) is invertible...

- ▶ There is no effect to the distribution of differences

Linear operation is only used to break up the algebraic structure

- ▶ Mixing \mathbb{F}_{2^n} operations with \mathbb{F}_2 -linear operations

Linear Cryptanalysis ByteSub provides the main defence against Linear Cryptanalysis as well

- ▶ Mainly due to the non-linear nature of the $1/x$ operation

Rijndael - MixColumn

The MixColumn operation also uses arithmetic in \mathbb{F}_{2^8}

Each column of the state is considered as a four dimensional vector with entries in \mathbb{F}_{2^8}

Perform the matrix operation

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 0x02 & 0x03 & 0x01 & 0x01 \\ 0x01 & 0x02 & 0x03 & 0x01 \\ 0x01 & 0x01 & 0x02 & 0x03 \\ 0x03 & 0x01 & 0x01 & 0x02 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Rijndael - MixColumn

The matrix is chosen for a number of reasons

- ▶ It has small coefficients
 - ▶ Makes it much easier to implement
- ▶ Invertible
 - ▶ Makes the cipher invertible (so we can decrypt)
- ▶ Very good at diffusing differences between the bytes
- ▶ Increases number of active S-Boxes

In our Toy Cipher the permutation stage was too simple

- ▶ Allowed us to easily map simple differences from the S-Box output
- ▶ to input simple differences to the next round
- ▶ Allowed us to have small number of active S-Boxes in linear cryptanalysis attack

Rijndael - MixColumn

The MixColumn step makes this mapping as complicated as possible

- ▶ Being a bit vague here,

Actually related to the fact that matrix is chosen to be related to an **Maximal Distance Separable Code**

This is a linear code with maximal distance

- ▶ For those who know about coding theory

Rijndael - Algebraic Cryptanalysis

Since much of the Rijndael structure makes use of algebraic properties

- ▶ Arithmetic in the field \mathbb{F}_{2^8}

Some people have suggested this gives it extra structure which may make it weak

Extra structure investigated by Murphy and Robshaw

At present no-one knows how to exploit the extra structure to obtain an attack