COMP2300/COMP6300 - Applied Cryptography

Symmetric Cryptography

Les Bell, les.bell@mq.edu.au

Department of Computing

MACQUARIE University

MACQUARIE University

Revision : 1.15

Table of Contents

1 Basic Principles
- Shannon, Feistel and Modern Cryptography

2 Data Encryption Standard
- The Trouble with DES
- Strengthening DES

3 Attacking Block Ciphers
- General Attacks
- Specialized Attacks

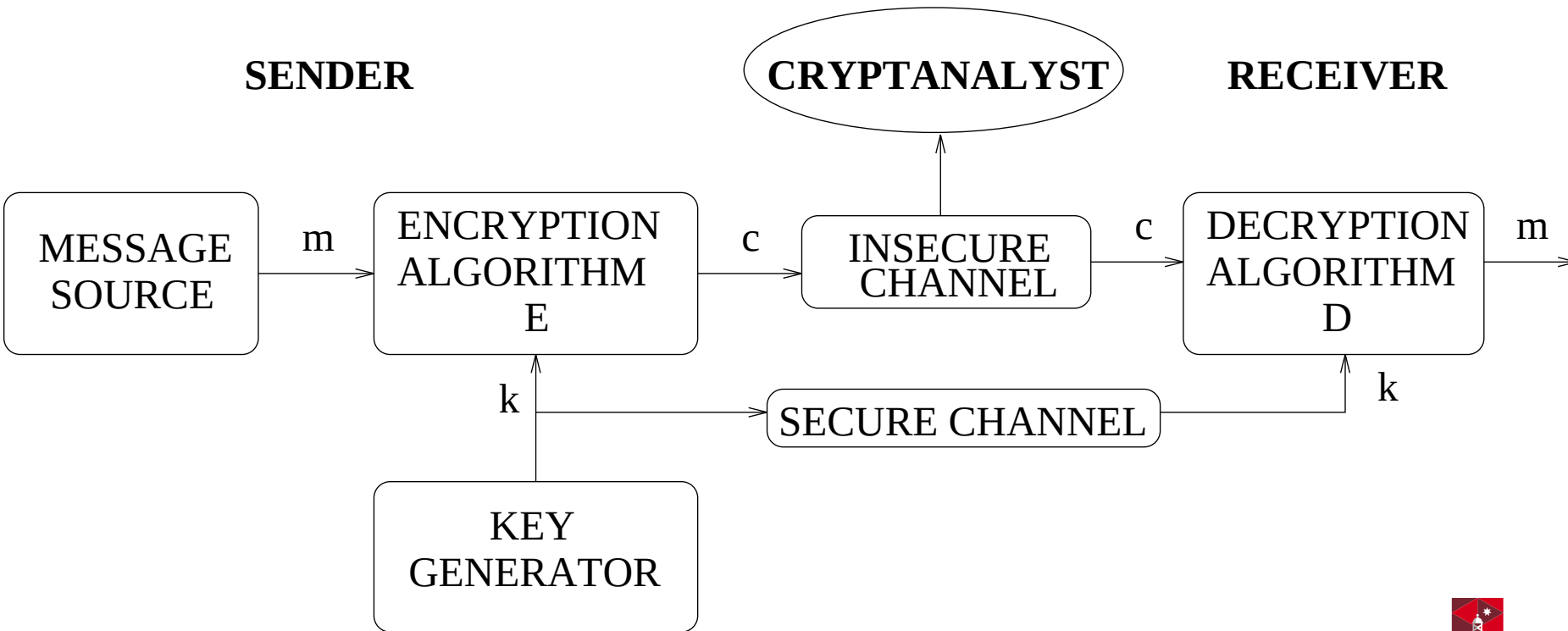
4 Other Block Ciphers

5 The AES Competition
- Rijndael

6 Key Generation and Stream Ciphers
- Sources of Randomness
- Linear Feedback Shift Registers
- Block Cipher Modes

MACQUARIE University

Secret-Key (Symmetric) Cryptography



MACQUARIE University

Kerckhoff's Maxims

From "*La Cryptographie Militaire*" (1883):

1. The system should be, if not theoretically unbreakable, unbreakable in practice.
2. Compromise of the system should not inconvenience the correspondents.
3. The method for choosing the particular member (key) of the cryptographic system to be used should be easy to memorize and change.
4. Ciphertext should be transmittable by telegraph.
5. The apparatus should be portable.
6. Use of the system should not require a long list of rules or mental strain.

Note especially 2, also rephrased by Claude Shannon: "The enemy knows the system".

Logic

You *must* be familiar with the basic logic operators, AND (conjunction, $\wedge$), OR (disjunction, $\vee$) and exclusive-OR (exclusive-disjunction, $\oplus$ or $\veebar$):

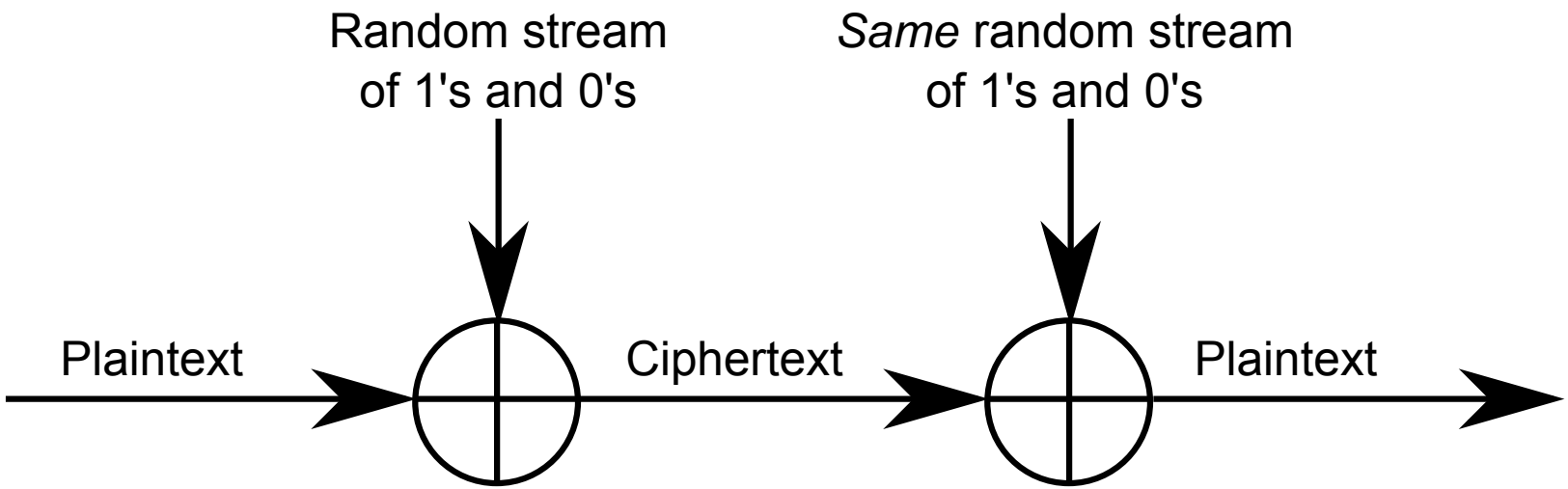
A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

You should also know how to perform these operations on both boolean values and bit-strings in your chosen programming language.

What is the relationship between B and $A \oplus B$?

Vernam Encryption

This is simple XOR'ing of the plaintext with the keystream
Is self-inverse



Shannon showed that we can achieve perfect secrecy with this scheme, *if* the keystream is genuinely random - but we need a secure channel to transfer the keystream from sender to receiver, in addition to the ciphertext (but hold that thought!).

MACQUARIE University

One-Time Pad

Message Space: $\mathcal{M} = \{0,1\}^n$ – all possible n-bit sequences

Cryptogram Space: $\mathcal{C} = \{0,1\}^n$ – all possible n-bit sequences

Key Space: $\mathcal{K} = \{0,1\}^n$, $\mid \mathcal{K} \mid = 2^n$

Encryption: $c_i = E_k(m_i) = m_i \oplus k_i$.

Decryption: $m_i = D_k(c_i) = c_i \oplus k_i$.

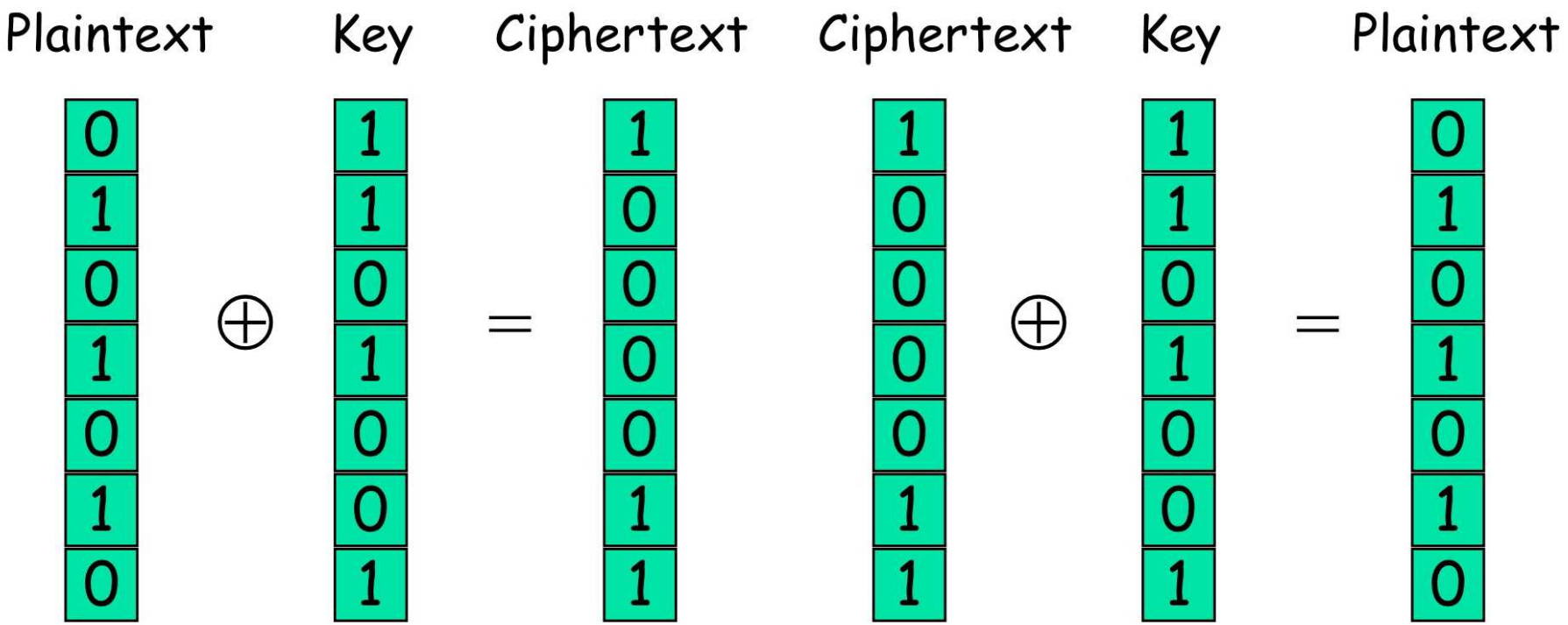
Cryptanalysis: This cipher is unconditionally secure. The only method of breaking it is by guessing!

MACQUARIE University

One-Time Pad

Encryption

Decryption



Properties of OTP

- Advantages
  - Easy to encrypt and decrypt
  - This is an information-theoretically secure cipher, i.e. given ciphertext, all possible plaintexts are equally likely, assuming that key is chosen randomly
- Disadvantage
  - Key is as long as the plaintext
  - Distribution of the cryptographic key needs a secure communication channel

MACQUARIE University

Modern Electronic Cryptography

We can trace the beginnings of modern digital cryptography to the ideas of confusion and diffusion developed by Claude Shannon at Bell Labs [Sha49], and the later work of Horst Feistel and others at IBM.

References: Menezes, et. al [MVOV97], section 7.4

Product Ciphers

Shannon's big breakthrough: *product ciphers*.

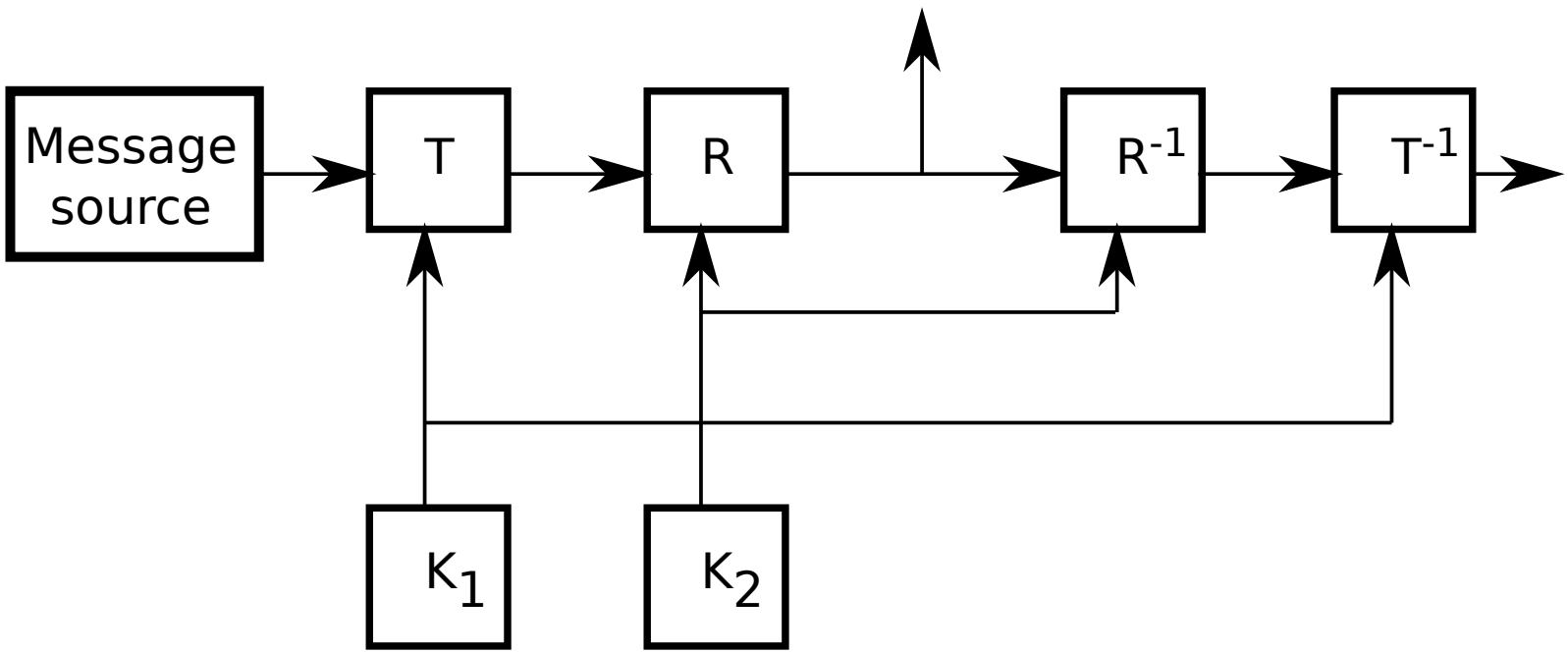
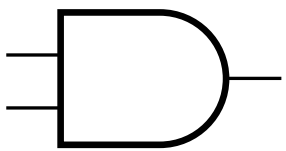


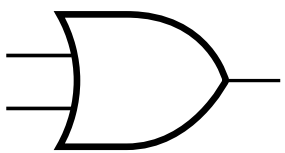
Figure 1: Product of two systems $S = RT$ (after Shannon, 1949)

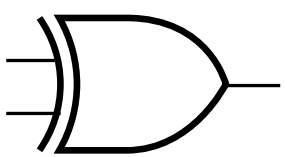
MACQUARIE University

Logical Operations

Circuit symbols:

AND 

OR 

XOR 

Truth table:

A	B	$A \wedge B$	$A \vee B$	$A \oplus B$
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

The XOR ($\oplus$) operation is probably the most important idea in symmetric cryptography.

MACQUARIE University

Incorporation of Key Material

We use the XOR operation to selectively invert some bits of a block:

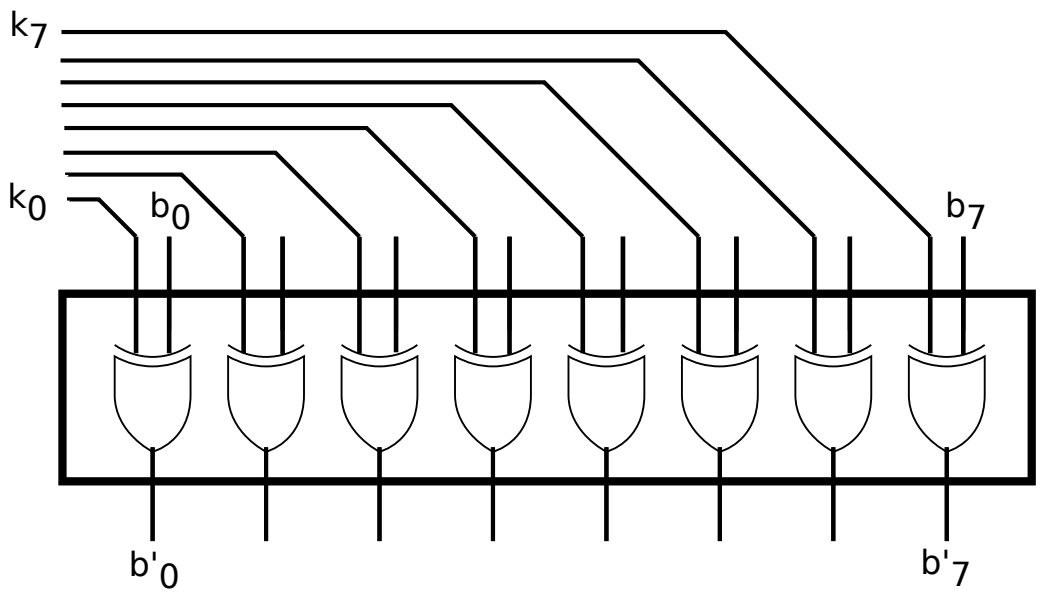
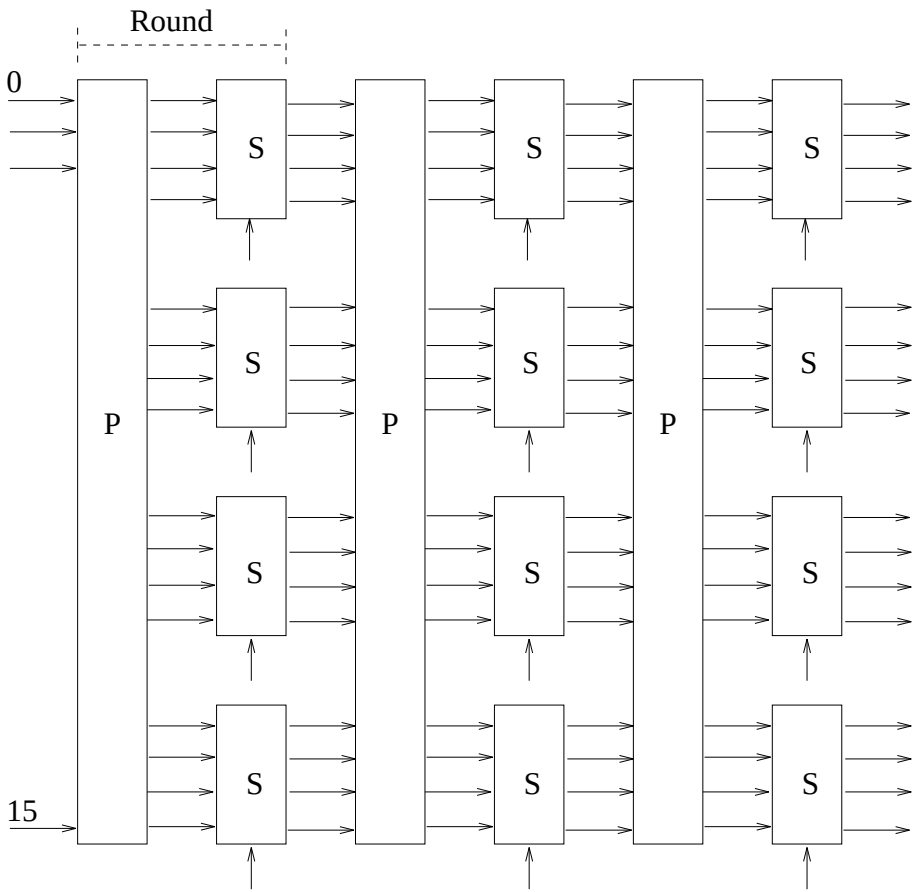


Figure 2: Incorporation of key material in a block

Confusion and Diffusion

Are achieved through *substitution* and *permutation*



Substitution

We achieve substitution by any of several methods, but most common is a lookup table called an *s-box*:

Input	Output
0000	0000
0001	0001
0010	1011
0011	1101
0100	1001
0101	1110
0110	0110
0111	0111
1000	1100
1001	0101
1010	1000
1011	0011
1100	1111
1101	0010
1110	0100
1111	1010

MACQUARIE University

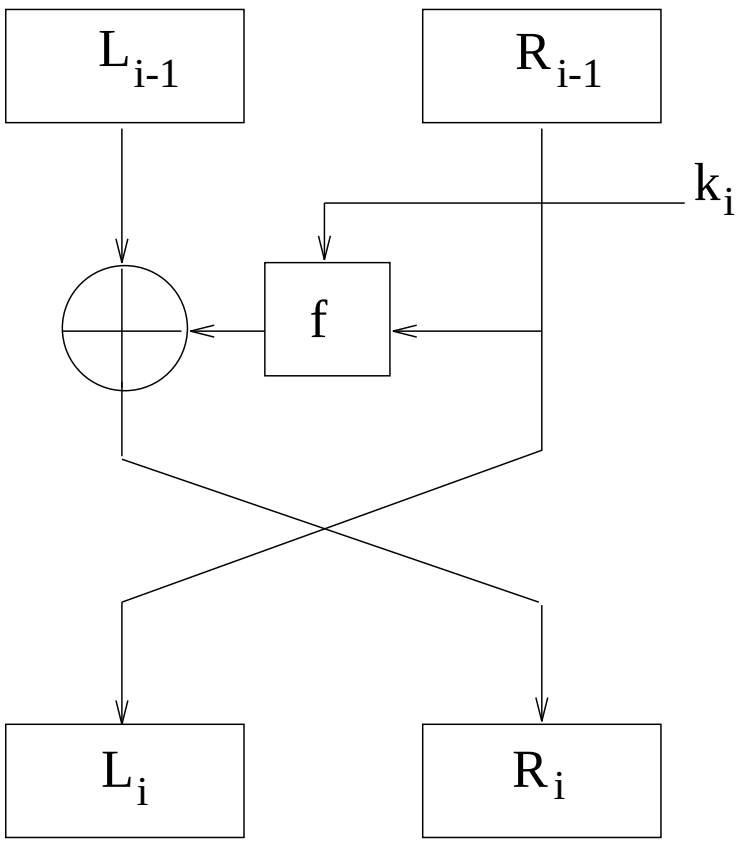
Permutation

There are various ways of achieving permutation and/or transposition of bits, including register swaps, bitwise left or right rotation of blocks or parts of blocks, or P-boxes.

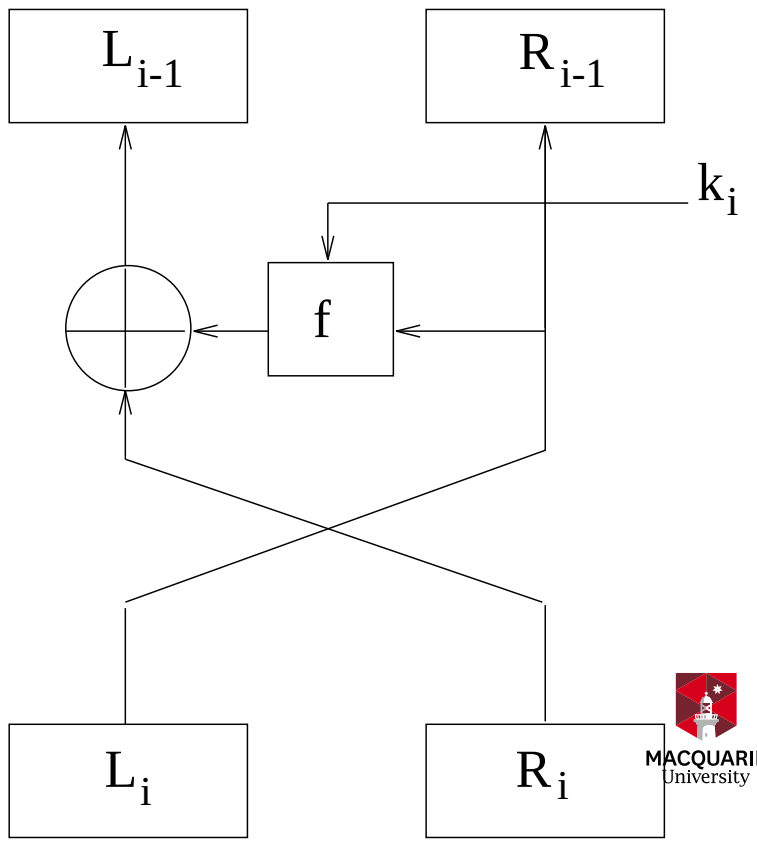
The permutation 'mixes' bits so that some of the output of one S-box enters a different S-box in the next round. When repeated across multiple rounds, this creates an *avalanche effect* - if one bit of an input block changes, then (ideally) every bit of the output block has a 50/50 chance of changing.

Feistel Permutation, or Feistel Network

a) Encryption



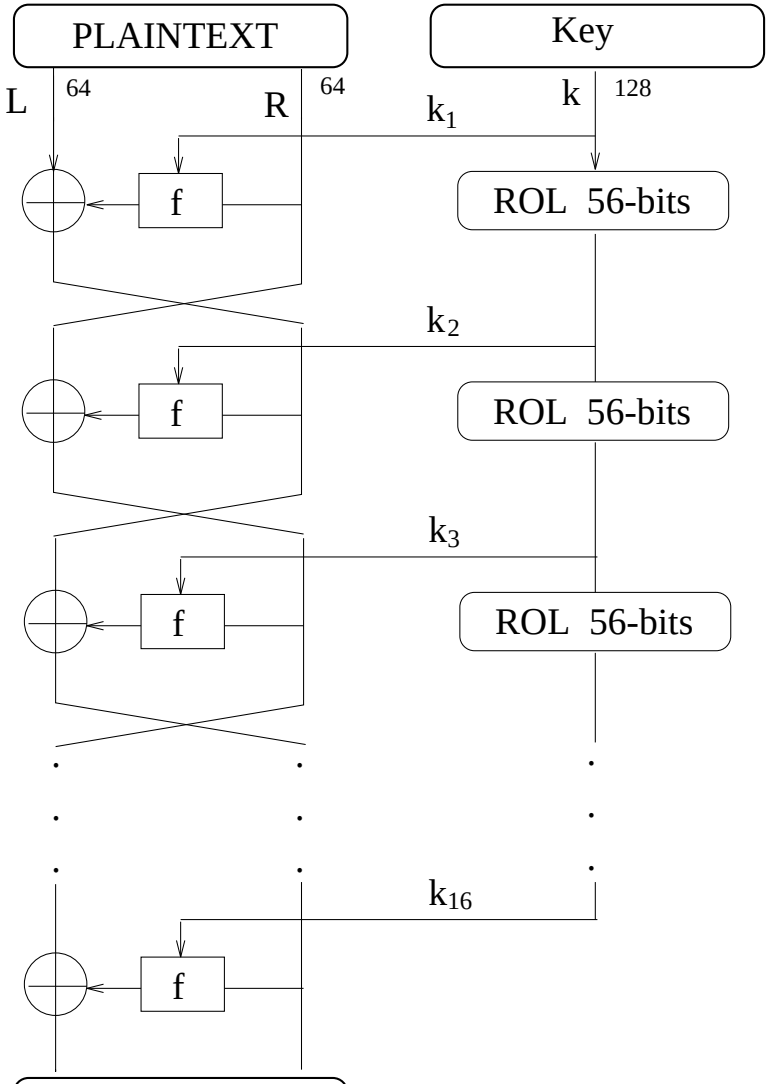
b) Decryption



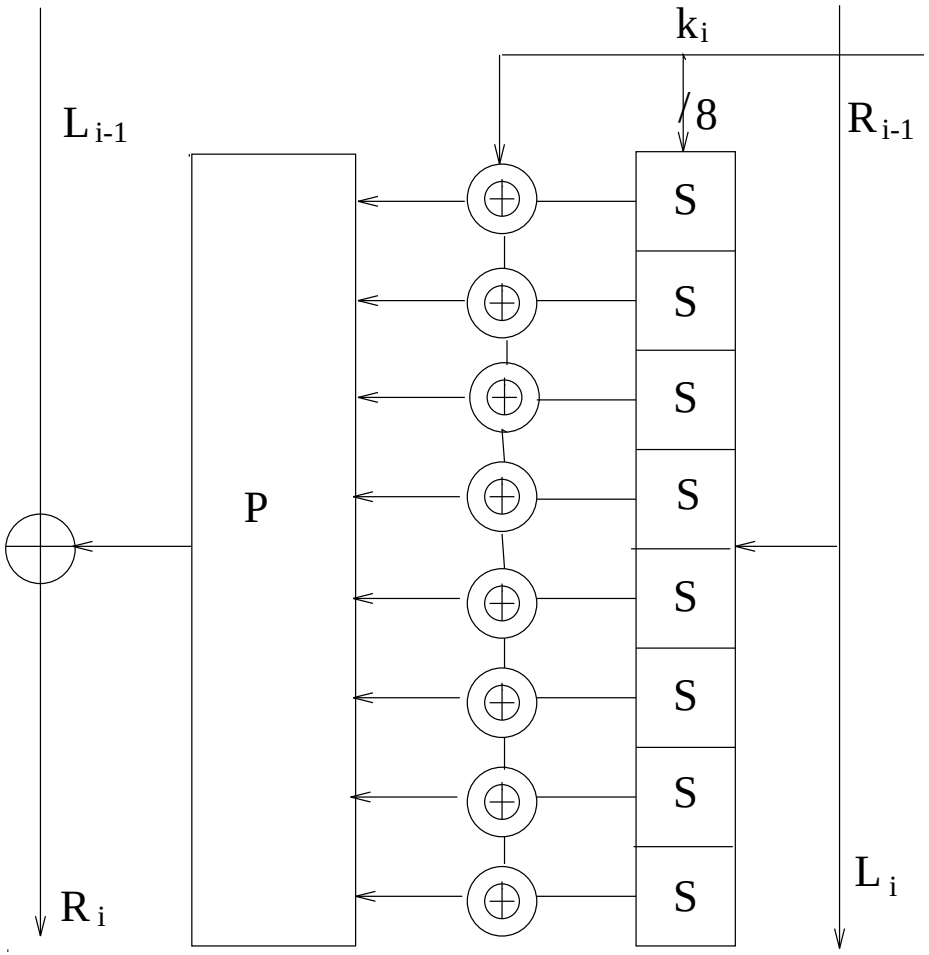
Data Encryption Standard

- Developed at IBM for government and banking applications
- Intended to be implemented in VLSI hardware
  - Remember, this is the era of the 2 MHz processor!
  - To minimize die size, use as much of the same circuitry for encryption and decryption as possible
- Original work by Horst Feistel on earlier versions
  - Dataseal
  - Demonstration Cipher
  - Demon
  - Lucifer (nice connotations of 'evil' and 'cifer') - used by Lloyds Bank Cashpoint in early-mid 1970's
  - DSD-1
- Had involvement from NBS (later NIST) (which really means NSA in the background) in two areas:
  - Key size - 56 bits
  - Selection of values in S-boxes

Lucifer Algorithm

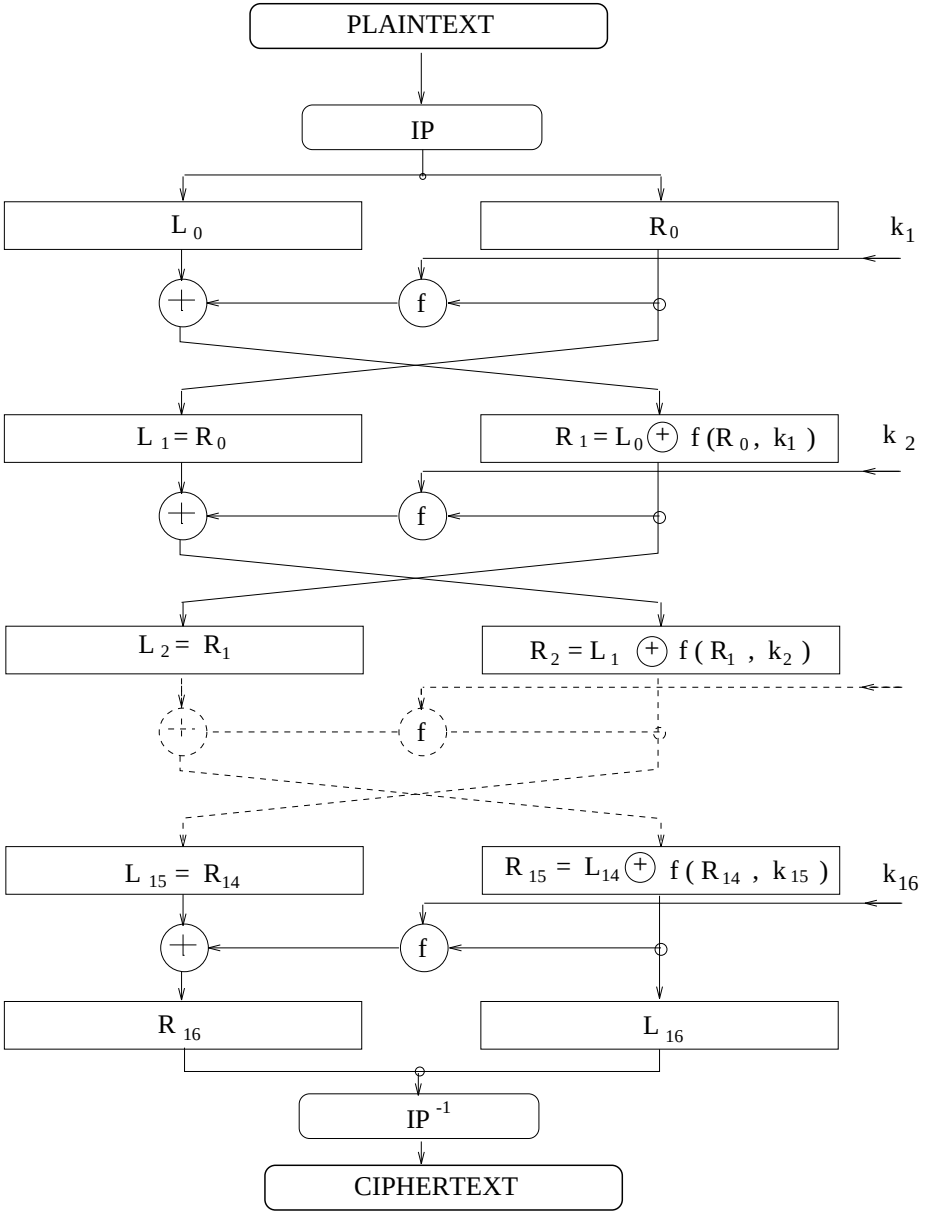


Lucifer Algorithm

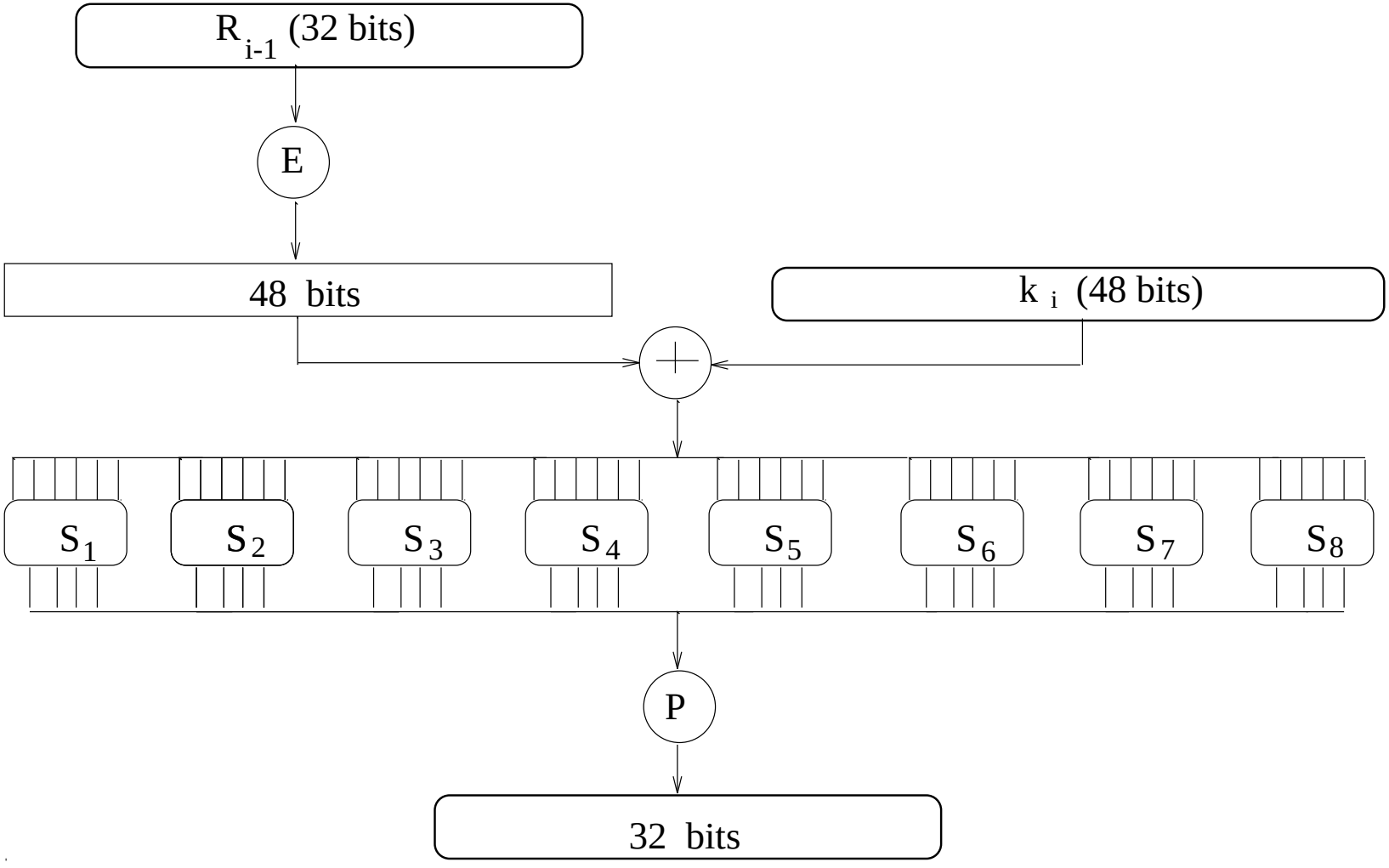


Lucifer function f

DES Algorithm



DES Algorithm



Function $f(k_i, R_{i-1})$

DES Algorithm S-Boxes (1)

	Column																
Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Box
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	

DES Algorithm S-Boxes (2)

	Column															
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

S-box Design Properties

Some S-box design properties are

- Each row function in an S-box is a permutation (S-boxes produce sequences with balanced number of 0s and 1s).
- No S-box is linear or affine function of the input (S-boxes are nonlinear).
- A single-bit change on the input of an S-box changes at least two output bits (S-boxes provide "avalanche" effect).
- For each S-box S, $S(x)$ and $S(x \oplus 001100)$ must differ in at least two bits.
- $S(x) \neq S(x \oplus 11ef00)$ for any choice of bits e and f.
- The S-boxes minimize the difference between the number of 1's and 0's in any S-box output when any single bit is constant.

MACQUARIE University

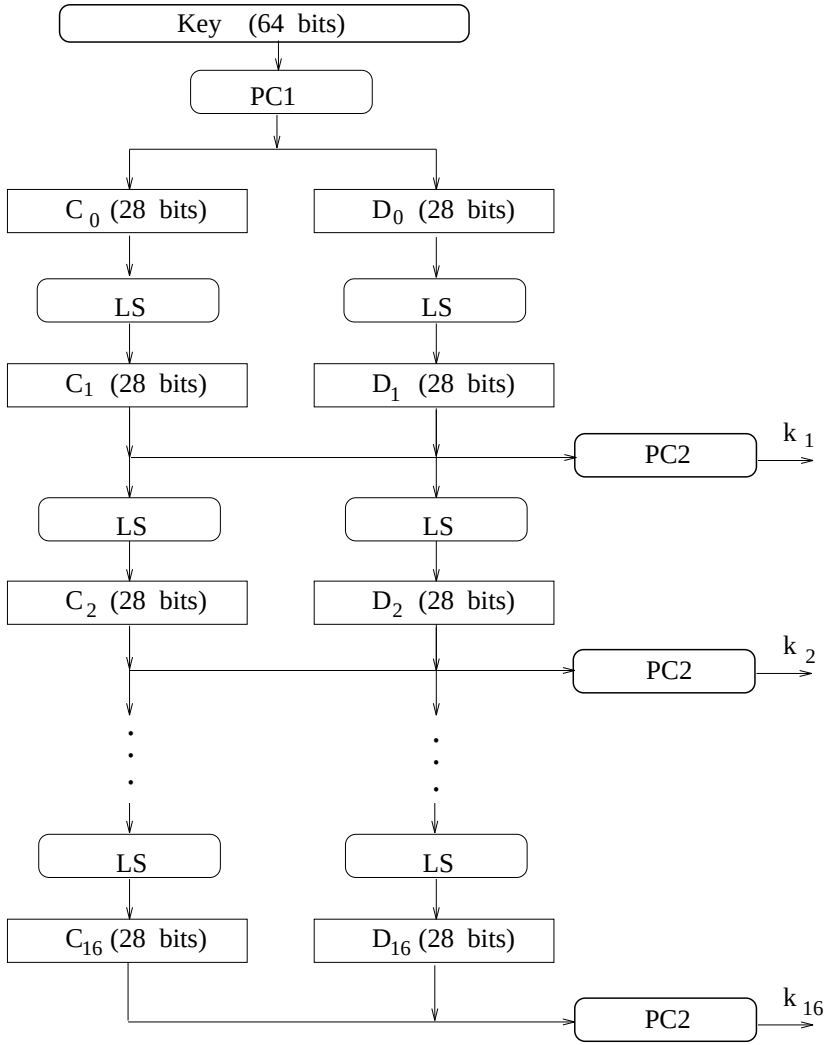
P-box and Function E

Concatenation of the P-box and the function E.

- Each S-box input bit comes from the output of a different S-box.
- No input bit to a given S-box comes from the output of the same S-box.
- An output from S_{i-1} goes to one of the *ef* input bits of S_i and further via E an output from S_{i-2} goes to one of the *ab* input bits.
- An output of S_{i+1} goes to one of the *cd* inputs bits of S_i.
- For each S-box output, two bits go to *ab* or *ef* input bits, the other two go to *cd* input bits.

MACQUARIE University

DES Key Schedule



The Trouble with DES

- There were initial concerns about DES:
  - Why was the key size so small? Lucifer used 128-bit keys.
  - Why did NBS (NIST) (really the NSA?) insist on certain values in the S-boxes?
    - At the second NBS workshop on DES, an IBM representative said the S-box values were chosen to strengthen the algorithm. When asked for proof, he said, "You must trust us, we are all good boy scouts".
- In short, was there a back door?
- There was also speculation that a DES-cracking machine could be built, or computers would become fast enough to crack it in software.

MACQUARIE University

56-bit Keys?

- The usual official argument advanced for the 56-bit keys was the desire to allow export
- It is certainly the case that IBM was permitted to export DES crypto hardware
- However, at least one other company was told by the Office of Munitions Control that there was no point in applying for a license to export, as it would be refused

Weak Keys for DES

- All the round keys produced by the *weak keys* are identical
  - All 1's
  - All 0's
  - Two other values
- There are also six other pairs of *semi-weak* keys keys, which each produce only two different round keys that are used eight times each
  - A key in each semi-weak pair can decrypt messages encrypted with the other

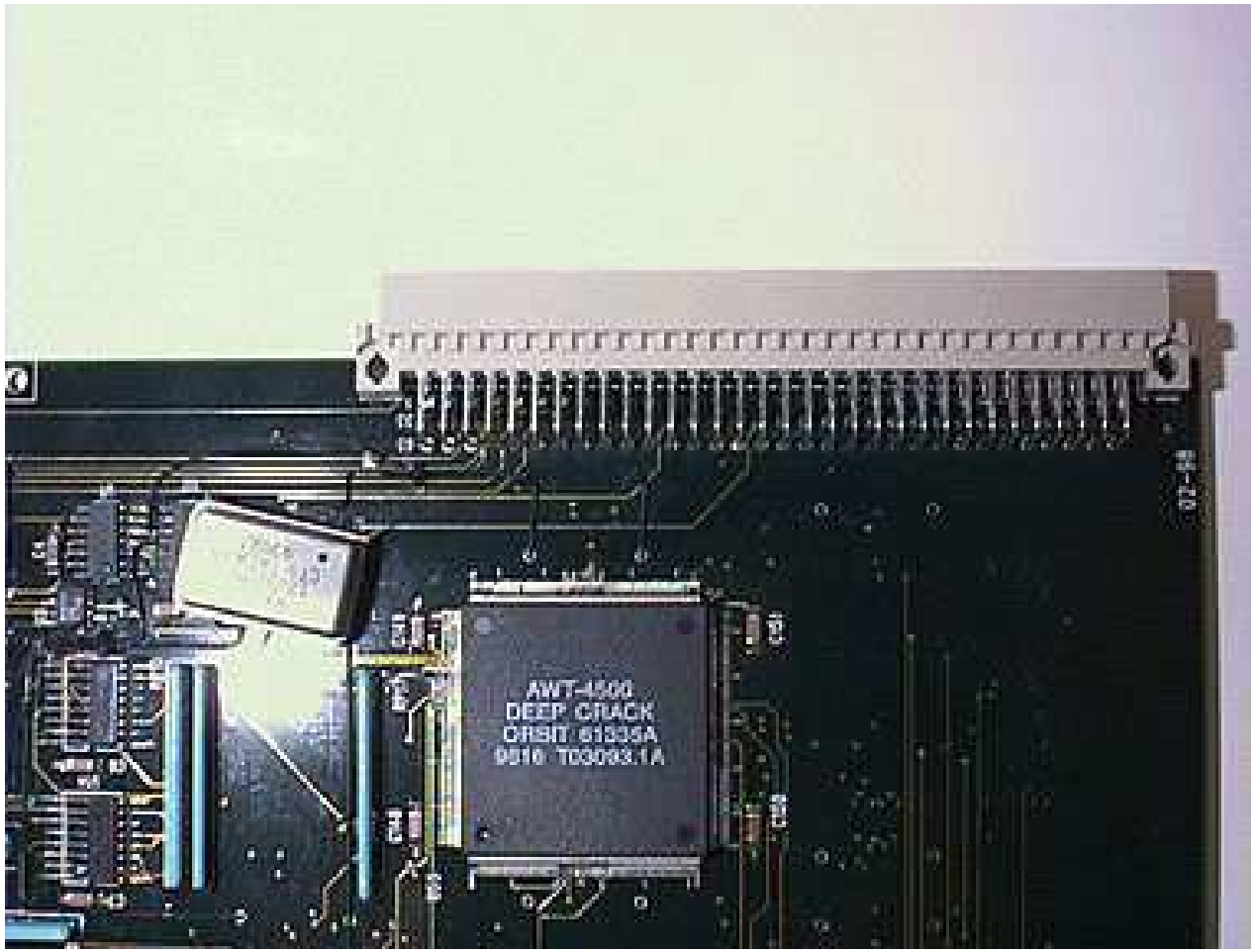
MACQUARIE University

DES Cracking

The Electronic Frontier Foundation's DES Cracker (1998) could test 90 billion keys per second and cracked RSA's DES Challenge II in less than 3 days.

MACQUARIE University

EFF Deep Crack Chip



Distributed Cracks

DES Challenge III was won by Distributed.Net, a world-wide network of nearly 100,000 PC's and workstations including the EFF DES Cracker. It broke DES in 22 hours and 15 minutes, and was running at a rate of 245 billion keys per second.

The EFF DES Cracker cost less than $US250,000. One *has* to assume that government agencies with vastly larger budgets have had even better hardware for a lot longer.

Strengthening DES

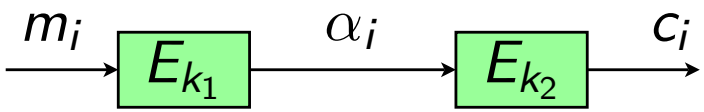
- If DES was weakening, what could be done?
- DES had achieved market success and IBM was now selling DES cryptographic coprocessors for IBM-compatible computers. These and similar DES-specific cryptographic hardware was too expensive to simply scrap.
- I know! If DES provides a security level of 2^{56}, just run it twice – $E_{k_1}(E_{k_2}(m))$ – and get a security level of 2^{112}!
- Uhhh . . . not so fast, there!

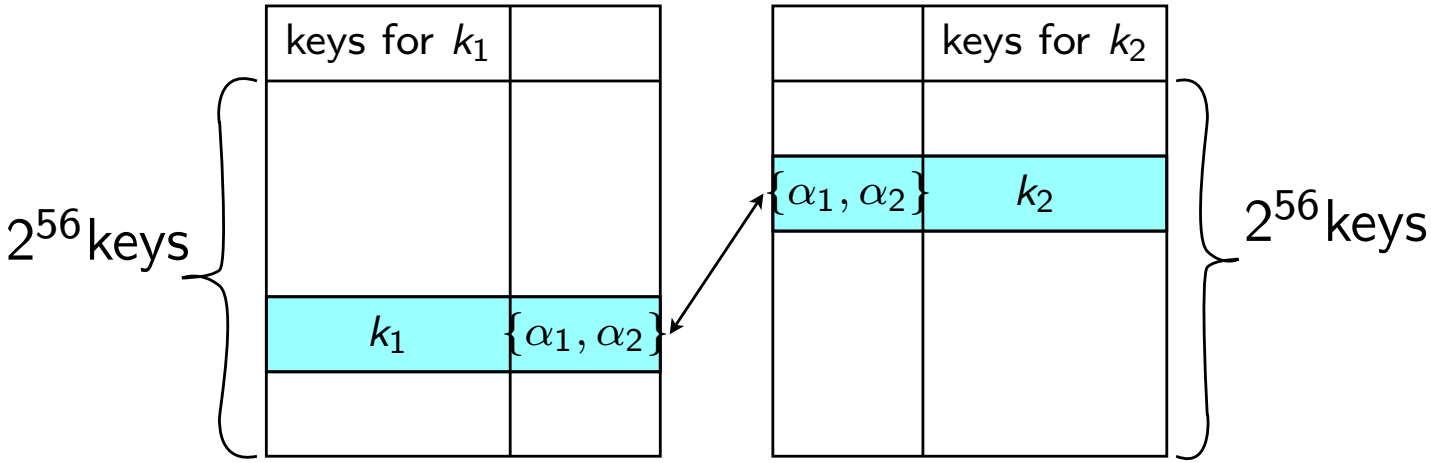
MACQUARIE University

Meet-in-the-Middle Attack

Attack on Double DES – Given two pair of observations (m_1, c_1) and (m_2, c_2) such that

$$c_i = E_{k_2}(E_{k_1}(m_i)) \text{ for } i = 1, 2$$





Encryptions for $\{m_1, m_2\}$ Decryptions for $\{c_1, c_2\}$

The complexity of the attack is $4 \times 2^{56} = 2^{58}$, which is much

Triple-DES (3DES)

The configurations for triple DES are as follows:

- $E_{k_1}(E_{k_2}(E_{k_3}(m))) - EEE(k_1, k_2, k_3)$ - security level 2^{168} (EEE3)
- $E_{k_1}(E_{k_2}(E_{k_1}(m))) - EEE(k_1, k_2, k_1)$ - security level 2^{112} (EEE2)
- $E_{k_1}(E_{k_2}^{-1}(E_{k_3}(m))) - EDE(k_1, k_2, k_3)$ - security level 2^{168} (EDE3)
- $E_{k_1}(E_{k_2}^{-1}(E_{k_1}(m))) - EDE(k_1, k_2, k_1)$ - security level 2^{112} (EDE2)

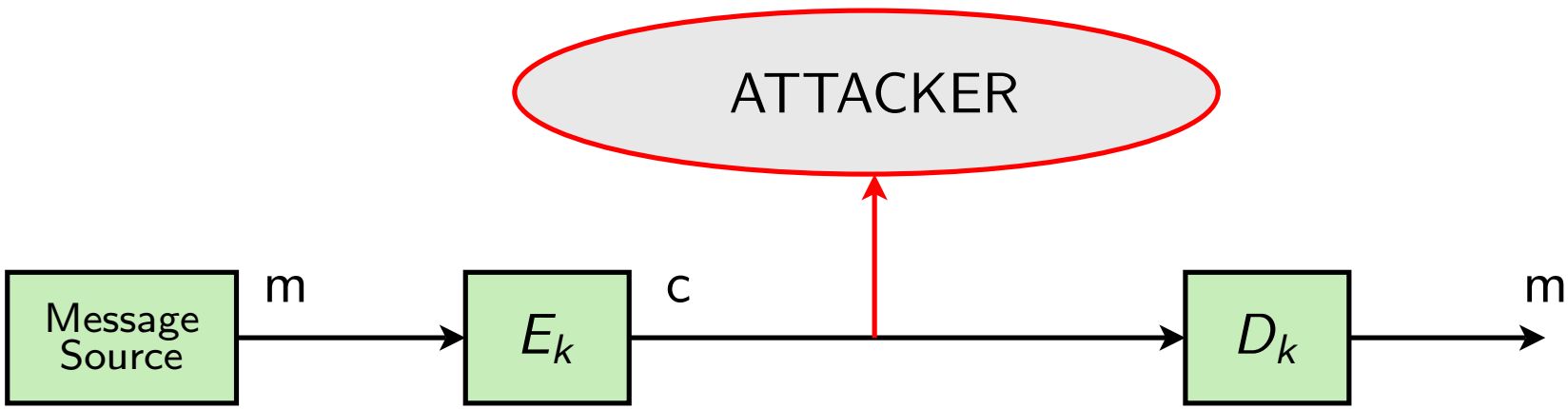
Notice that $E_{k_1}(E_{k_1}^{-1}(E_{k_1}(m))) - EDE(k_1, k_1, k_1)$ is the same thing as DES and has security level 2^{56}

General Attacks

General attacks:

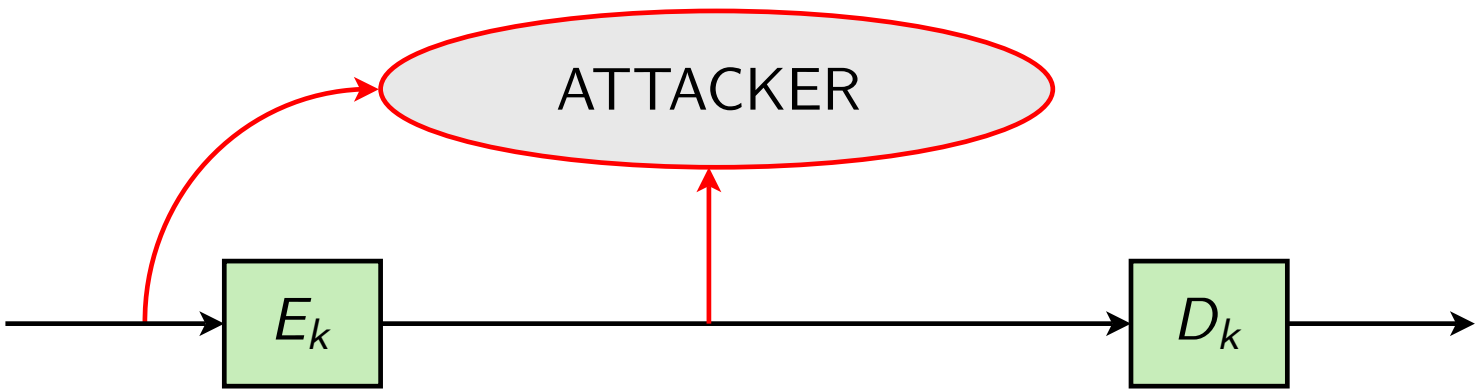
- Ciphertext-only attack
- Known-plaintext attack
- Chosen-plaintext attack
- Chosen-ciphertext attack

General Attacks – Ciphertext-Only Attack

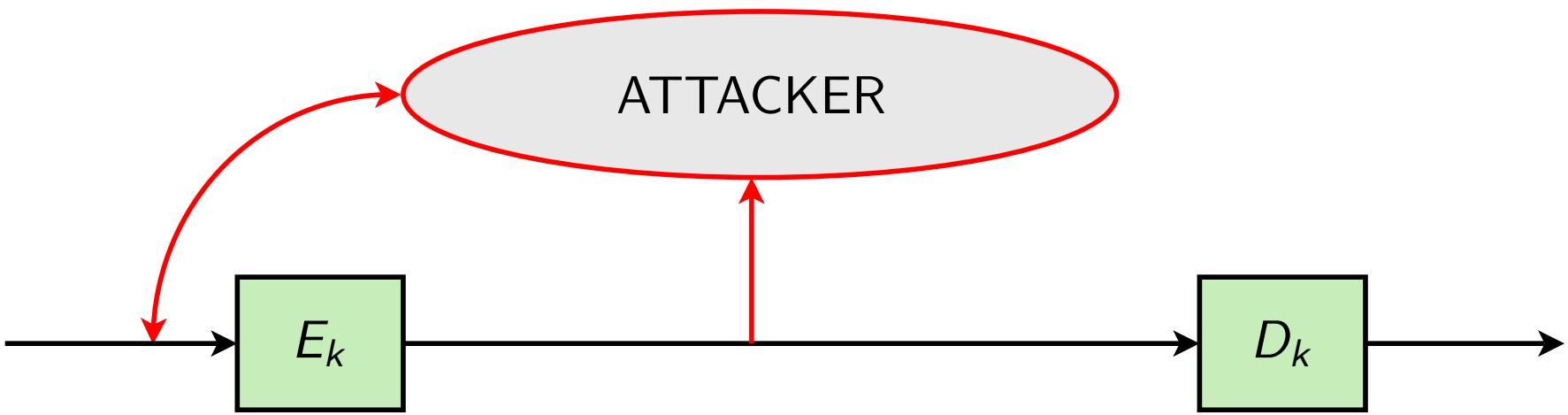


MACQUARIE University

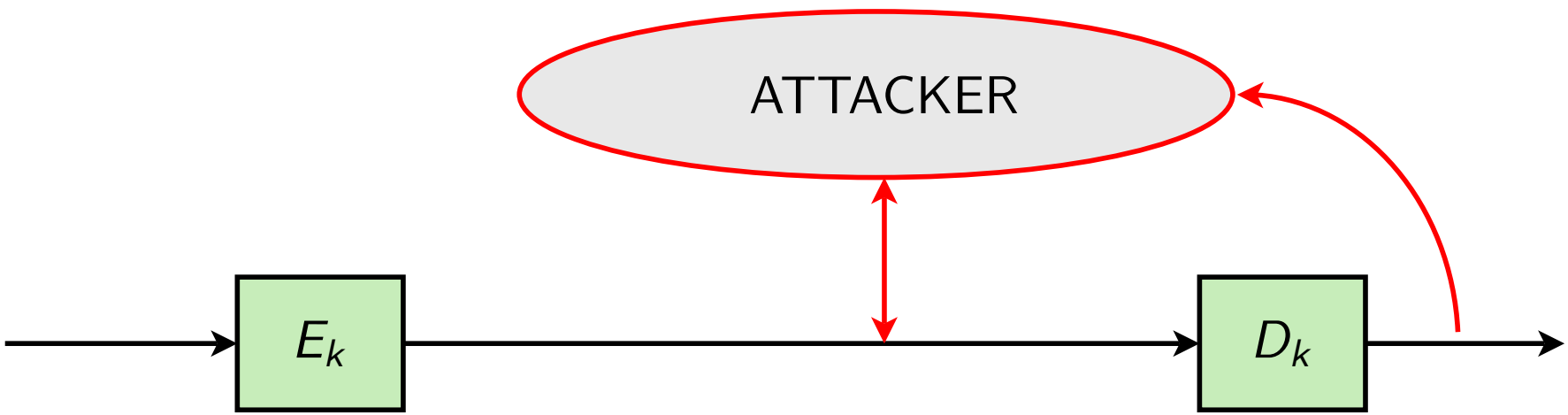
General Attacks – Known-Plaintext Attack



General Attacks – Chosen-Plaintext Attack



General Attacks – Chosen-ciphertext Attack



Specialized Attacks

- Differential Cryptanalysis
  - A known-plaintext attack which depends on variations (called *characteristics* in the probability with which certain bits are changed by the S-boxes.
- Linear Cryptanalysis
- Related-key Attacks
- Side-channel Attacks
  - Measuring current drain
  - Measuring radio frequency emissions

Differential Cryptanalysis

- "Discovered" by Eli Biham and Adi Shamir in 1990
- But is only better than brute force for 15 rounds or less
  - But for some reason, DES has 16 . . .
- It turns out the designers of DES had anticipated differential cryptanalysis (they called it "T attack"). So they generated and tested S-boxes to find values that were resistant to both T attack and linear cryptanalysis
- This answers the question of why NIST (NSA) changed IBM's values for the S-boxes.
- Uses *ciphertext pairs* – pairs of ciphertexts whose plaintexts have specific differences. It analyses the evolution of these differences through the rounds of DES when they are encrypted with the same key.

MACQUARIE University

Linear Cryptanalysis - Overview

Linear cryptanalysis was first described by Mitsuru Matsui in a 1993 paper [Mat94].

The idea is to use *linear approximation* to describe the action of a block cipher.

$$\left.\begin{array}{l}\text{plaintext bits} \oplus \\ \text{ciphertext bits} \oplus\end{array}\right\} \oplus \rightarrow b$$

b is a single bit that is the XOR of some of the key bits, with some probability, p. $p = 0.5$ for an ideal cipher.
So, we collect plaintexts and associated ciphertexts, to guess the values of the key bits. The more data, the better. And the more $p \neq 0.5$ – the more *bias* – the better.

Other Block Ciphers

No need to memorize these - just look at the similarities.

What you will notice is that they are almost all variants on a Feistel network, and some use more complex substitution techniques such as addition and multiplication.

While some of these ciphers are mostly of historical significance in the development of cryptography, many are still in use - for example, in PGP and Gnu Privacy Guard.

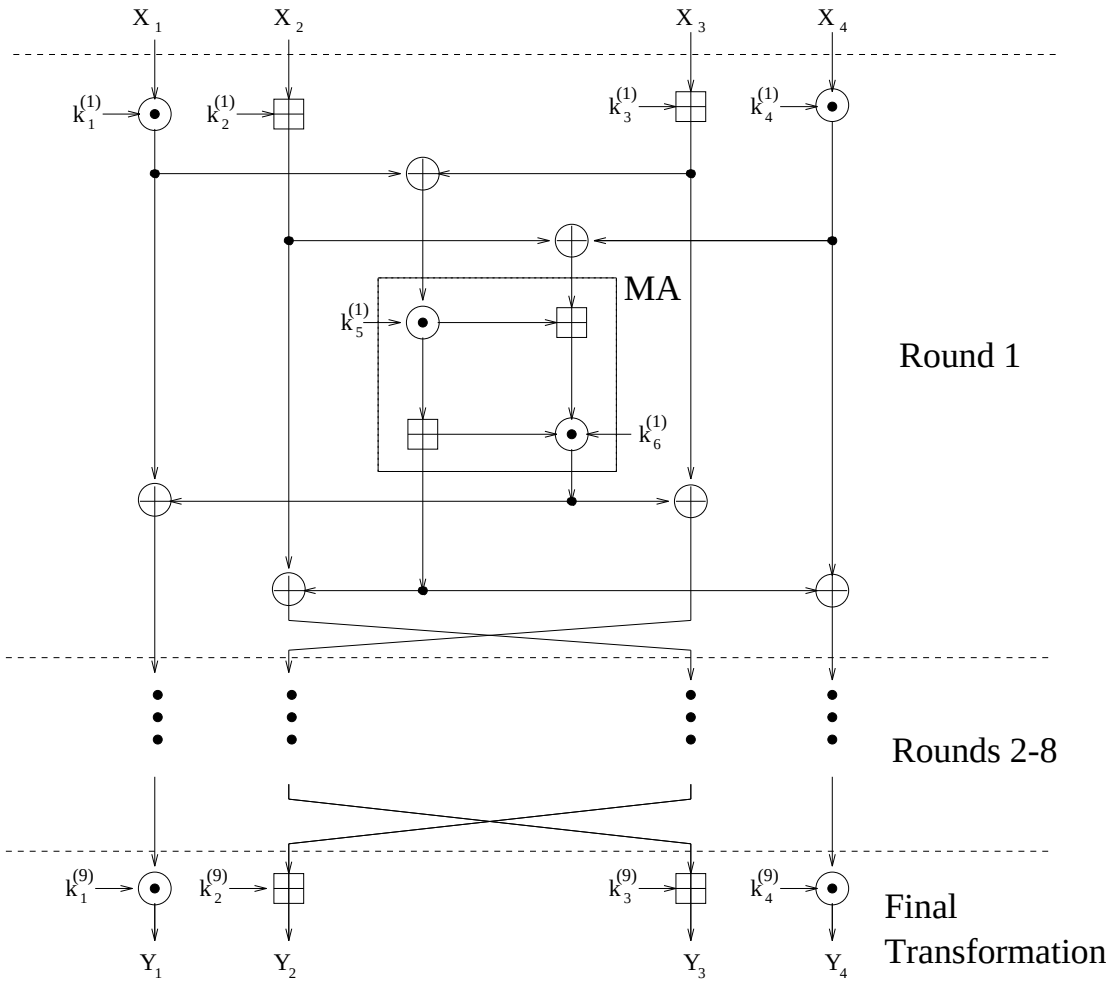
MACQUARIE University

International Data Encryption Algorithm

- Developed by Xuejia Lai and James Massey
- 64-bit block cipher with 128-bit keys
  - Divides the input block into four 16-bit sub-blocks (X1 - X4)
  - Eight rounds
    - Each round XOR's, adds and multiplies the blocks with each other and with six 16-bit sub-blocks of key material
    - Between rounds, swaps X2 and X3
  - Plus an output "half-round"
- Used in PGP/GPG
- Performance similar to DES (slightly faster in hardware)
- Patents expired in 2011
- Led to State Department permitting export of strong crypto

MACQUARIE University

IDEA Algorithm



MACQUARIE University

CAST-5 (CAST-128)

- Developed by Carlisle Adams and Stafford Tavares
- 64-bit blocks, 40 - 128-bit keys, 12 or 16 Feistel network rounds
- Patented but royalty-free for commercial and non-commercial use
- Standard symmetric cipher in PGP and GnuPG
- A derivative, CAST-256, was an early AES candidate

MACQUARIE University

RC5

- 32-bit, 64-bit and 128-bit block cipher
- Parameterized algorithm
  - Key size anywhere from 0 to 2040 bits
  - 0 to 255 rounds
- Performance about twice DES
  - Approximately - due to key setup times and key expansion routine
  - Each round performs integer addition, bitwise XOR and variable rotation

Blowfish

- 64-bit blocks
- Variable-length keys: 32 bits to 448 bits
- Much faster than DES and IDEA
- Unpatented and royalty-free

The AES Competition

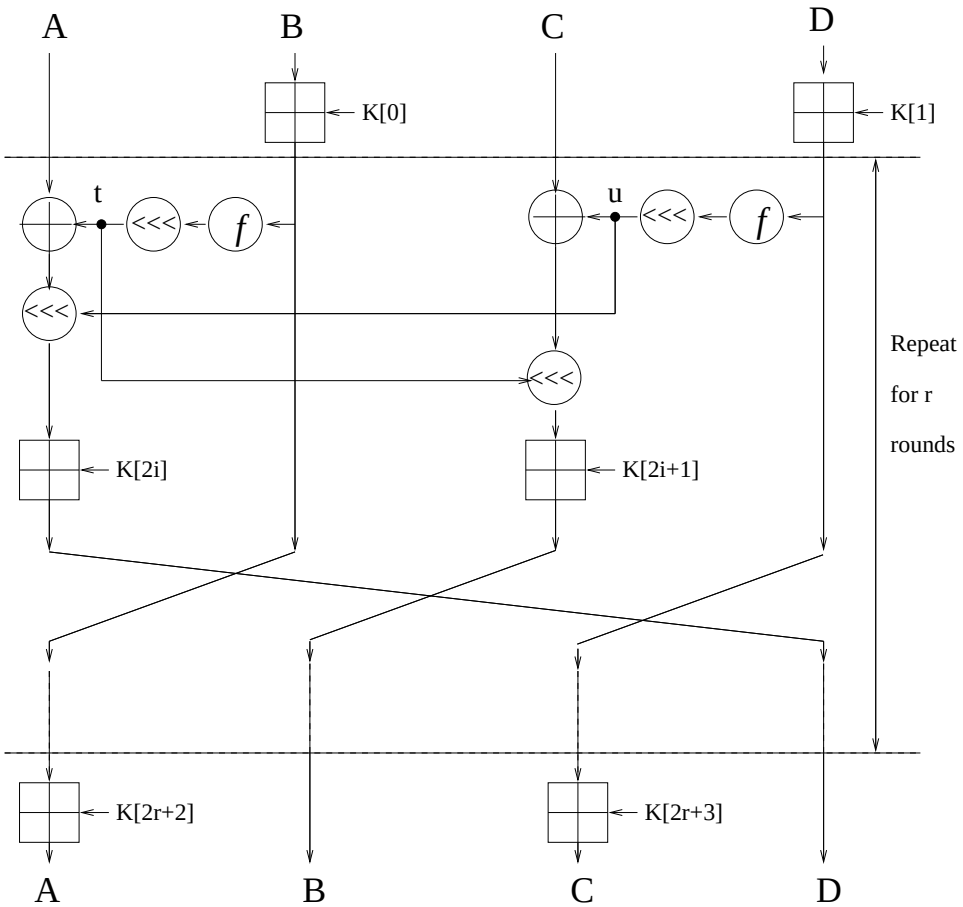
The five AES finalists were:

1. RC6 (Rivest Cipher 6) from RSA, Inc.
2. MARS from IBM
   - Which gave rise to the magnificently-titled paper, "Mars Attacks! Preliminary Cryptanalysis of Reduced-Round Mars Variants" by Schneier and Kelsey
3. Twofish from Counterpane Inc (Schneier et. al.)
4. Serpent from Ross Anderson, Eli Biham and Lars Knudsen
5. Rijndael from Vincent Rijmen and Joan Daemen

Two were eliminated due to cost and one due to slow runtimes.

RC6 Algorithm

Based on RC5, but adds integer multiplication and more, larger registers (hence fewer rounds required).



MARS

- 128-bit block size, keys from 128 to 448 bits (in 32-bit increments)
- 32 rounds of Feistel network
  - Initial XOR of input block with key material (key whitening)
  - 8 unkeyed DES-like transformation rounds
  - 16 rounds of the cryptographic core
    - one 32-bit block is mixed with the other three by multiplication with key material, S-box lookup and variable rotation
  - 8 more unkeyed transformation rounds
  - Final key whitening before output
- Uses well-understood structures (conservative design)

See `http://www.quadibloc.com/crypto/co040406.htm`

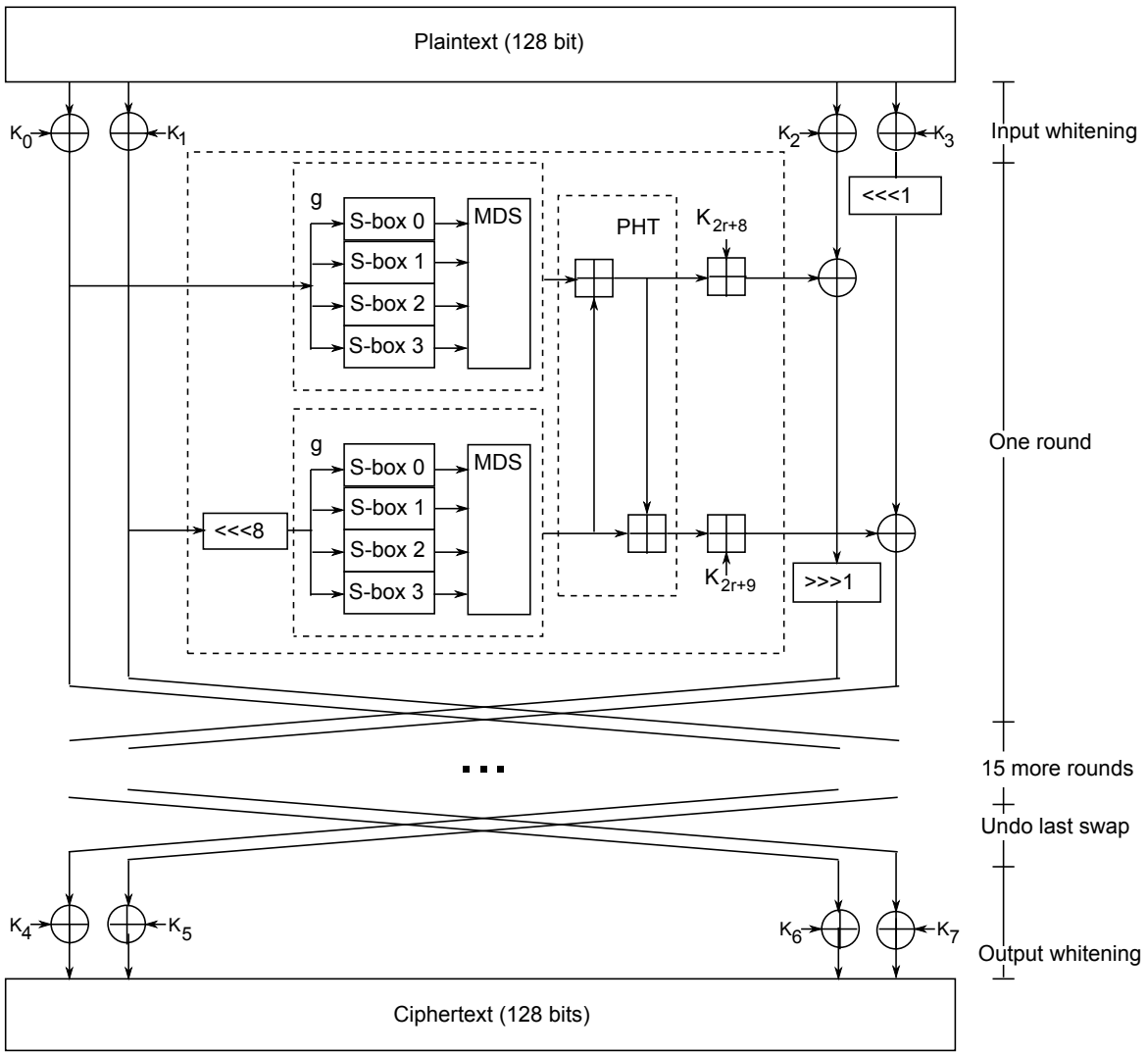
MACQUARIE University

Twofish

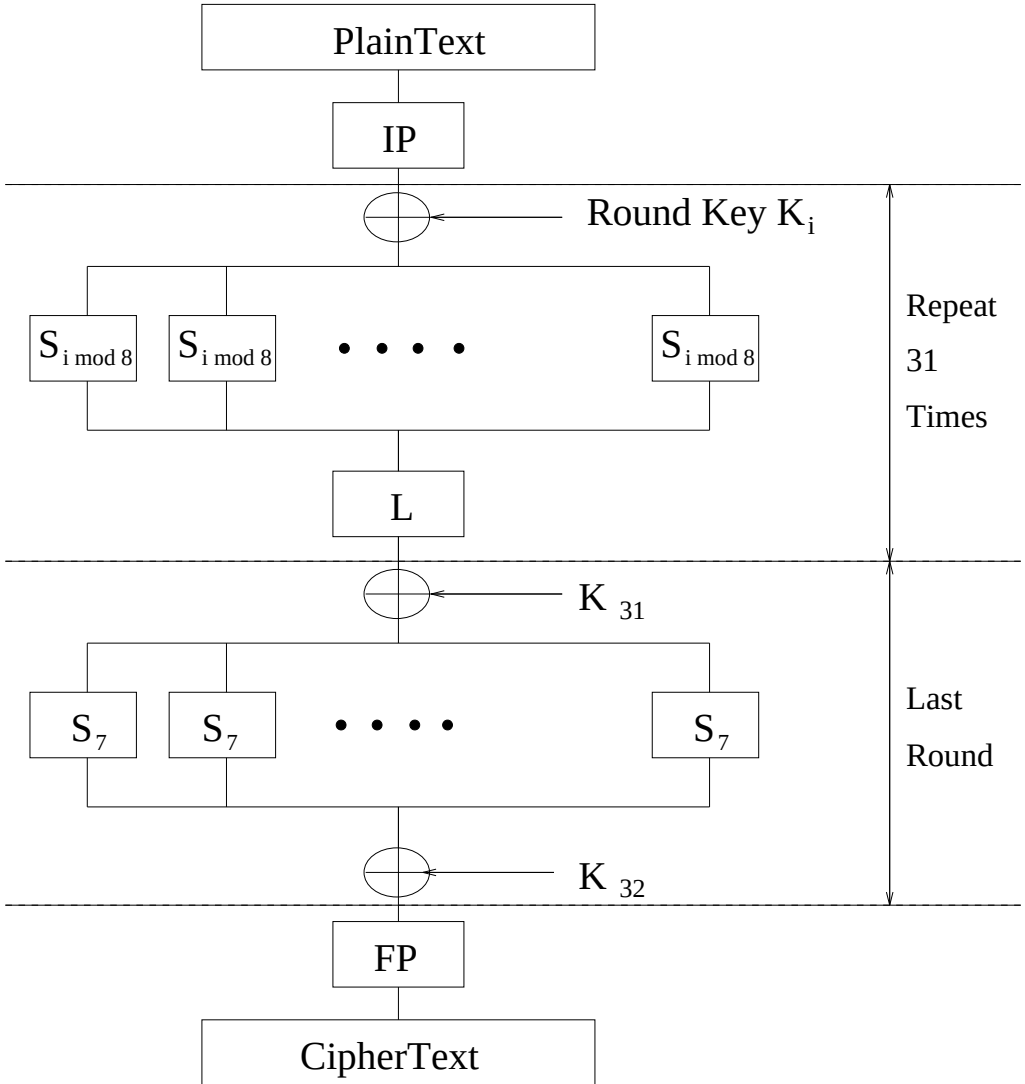
- 128-bit block cipher, accepts variable-length key up to 256 bits
- "The cipher is a 16-round Feistel network with a bijective F function made up of four key-dependent 8-by-8-bit S-boxes, a fixed 4-by-4 maximum distance separable matrix over $GF(2^8)$, a pseudo-Hadamard transform, bitwise rotations, and a carefully designed key schedule."

MACQUARIE University

Twofish


Plaintext (128 bit)
K0
K1
K2
K3
Input whitening
<<<1
g
S-box 0
S-box 1
S-box 2
S-box 3
MDS
PHT
K2r+8
g
S-box 0
S-box 1
S-box 2
S-box 3
MDS
<<<8
K2r+9
>>>1
One round
15 more rounds
Undo last swap
K4
K5
K6
K7
Output whitening
Ciphertext (128 bits)

Serpent Algorithm



Rijndael

The winner is . . .

Rijmen and Daemen put an audio file on their website for those who wanted to learn how to say it correctly. When played, one heard, "The correct pronunciation is ... AES".

The AES standard is FIPS-197, which specifies a subset of the Rijndael functionality (e.g. block size is always 128 bits) but keys can be 128 bits, 192 bits or 256 bits.
The 128-bit keyspace provides $3.4x10^{38}$ keys, while 256-bit keys give $1.15x10^{77}$ keys.
A device that could crack DES in one second would take 149 trillion years to crack a 128-bit DES key.

AES – Rijndael

Parameters:

- N_b – number of 32-bit words in message/cryptogram bocks
- N_k – number of 32-bit words in the primary key
- N_r – number of rounds

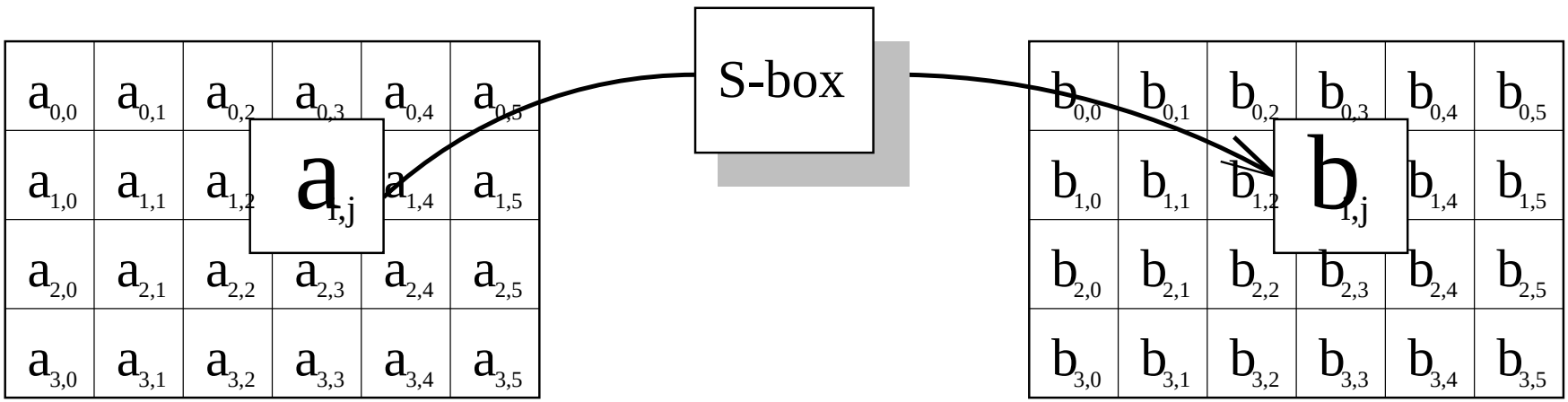
For 128-bit Rijndael $N_b = N_k = 4$ and $N_r = 10$.

For AES with a 256-bit key, $N_b = 4, N_k = 8$ and $N_r = 14$.

- `ByteSub` – the input block with $4N_b$ bytes is subject to byte-by-byte transformation using the S-box,
- `ShiftRow` – the bytes of the input are arranged into four rows and every row is rotated the fixed number of positions,
- `MixColumn` – the bytes of the input are arranged into four rows and every column is transformed using polynomial multiplication over $GF(2^8)$,
- `AddRoundKey` – the input block is XOR-ed with the round key.

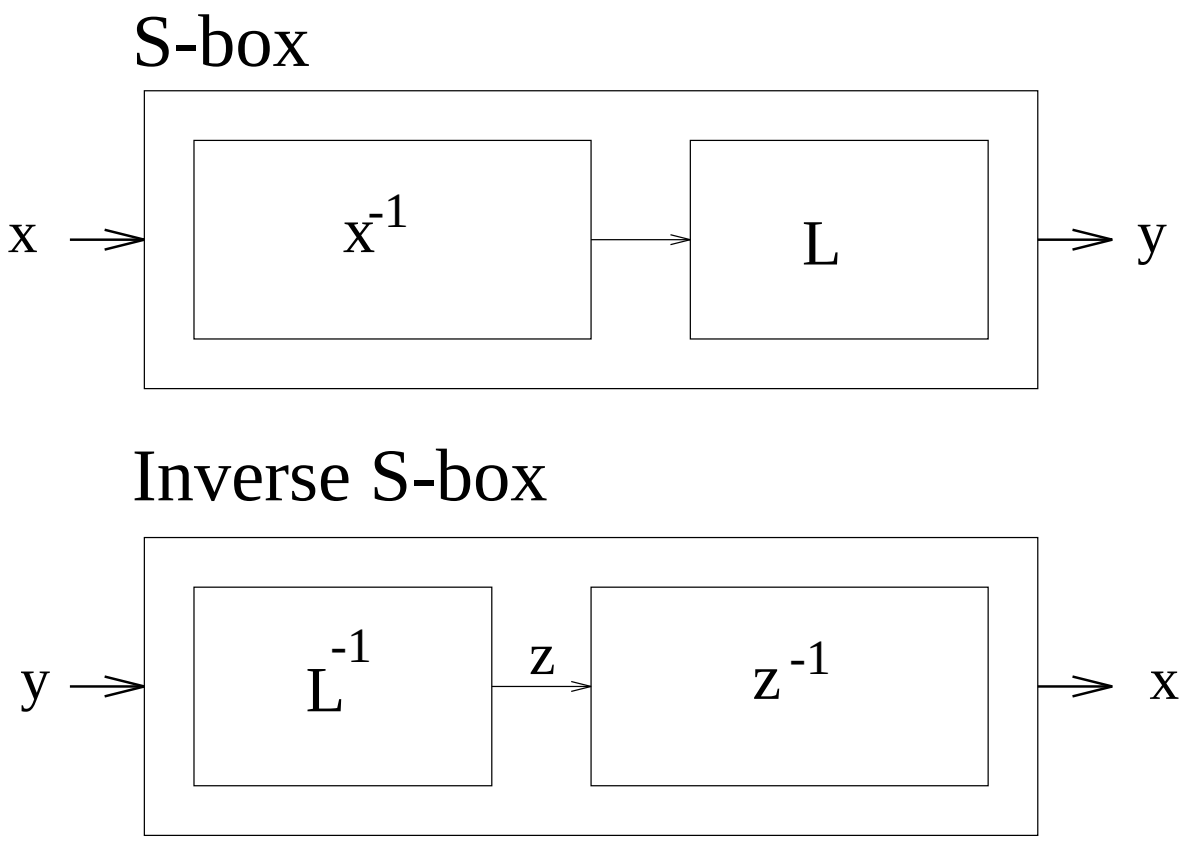
MACQUARIE University

Rijndael – Bytesub Transformation



MACQUARIE University

Rijndael – S-box

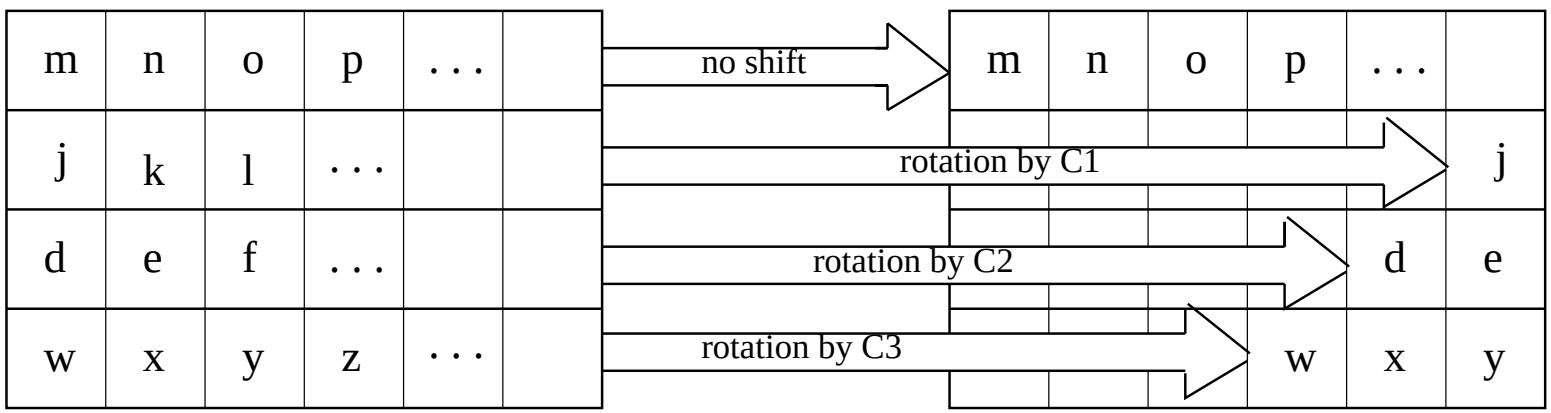


Rijndael – S-box

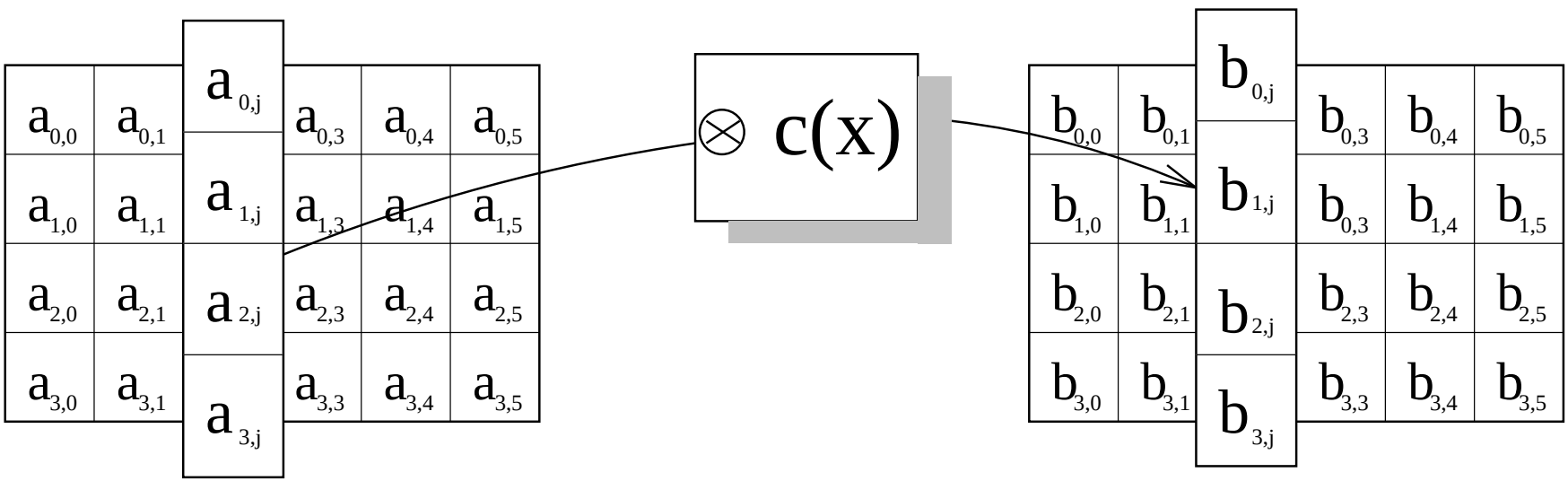
$$y = \begin{bmatrix} 10001111 \\ 11000111 \\ 11100011 \\ 11110001 \\ 11111000 \\ 01111100 \\ 00111110 \\ 00011111 \end{bmatrix} x^{-1} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix},$$

MACQUARIE University

Rijndael – ShiftRow Transformation



Rijndael – MixColumn Transformation



MACQUARIE University

KeyExpansion Procedure

The key schedule procedure `KeyExpansion` produces key material $W = (W_0, \ldots, W_{N_b(N_r+1)-1})$ from the primary key $K = (K_0, \ldots, K_{N_k-1})$ where W_i, K_i are 32-bit words. It applies two functions:

- `SubByte`(a, b, c, d) which accepts four bytes and returns $(S(a), S(b), S(c), S(d))$,
- `RotByte`(a, b, c, d)=(b, c, d, a) - rotates bytes.

MACQUARIE University

KeyExpansion – First Version

`KeyExpansion` has two versions: one for $N_k \leq 6$ and the other for $N_k > 6$. The first version (for $N_k \leq 6$) takes two phases:

- initialisation where $W_i = K_i$ for $i = 0, \ldots, N_K - 1$,
- expansion phase which takes the last computed word and extends it for the next one. The steps are as follows:

$$
\begin{aligned}
&\text{tmp} = W_{i-1},\\
&\text{if } i \pmod{N_k} = 0,\\
&\text{ then tmp=}\texttt{SubByte}(\texttt{RotByte}(\text{tmp})) \oplus \texttt{Rcon}_{\lfloor i/N_k \rfloor},\\
&W_i = W_{i-N_k} \oplus \text{tmp},
\end{aligned}
$$

where the constants $\texttt{Rcon}_i = (RC_i, 0, 0, 0)$ and $RC_i = xRC_{i-1} = x^{i-1}$ where x is an element of $GF(2^8)$.

KeyExpansion – Second Version

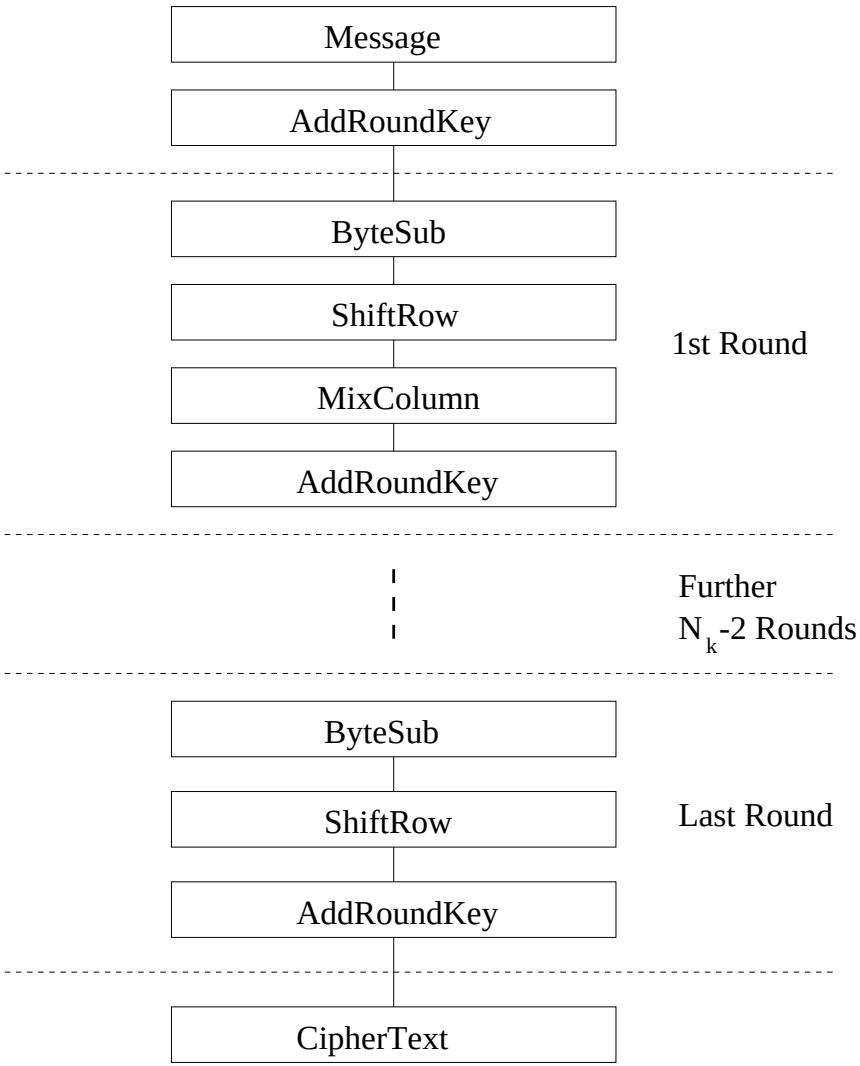
The second version (for $N_k > 6$) takes two phases:

- initialisation where $W_i = K_i$ for $i = 0, \ldots, N_K - 1$,
- expansion phase which takes the last computed word and extends it for the next one. The steps are as follows:

$$
\begin{aligned}
&\text{tmp} = W_{i-1},\\
&\text{if } i \pmod{N_k} = 0,\\
&\text{ then tmp=}\texttt{SubByte}(\texttt{ RotByte}(\text{tmp})) \oplus \texttt{Rcon}_{\lfloor i/N_k \rfloor}\\
&\text{else if } i \pmod{N_k} = 4 \text{ then tmp=}\texttt{SubByte}(\text{tmp}),\\
&W_i = W_{i-N_k} \oplus \text{tmp}.
\end{aligned}
$$

MACQUARIE University

Rijndael Encryption



Attacks on Rijndael

> *What concerns us most about AES is its simple algebraic structure. It is possible to write AES encryption as a relatively simple closed algebraic formula over the finite field with 256 elements. This is not an attack, just a representation, but if anyone can ever solve those formulas, then AES will be broken. This opens up an entirely new avenue of attack. No other block cipher we know of has such a simple algebraic representation. We have no idea whether this leads to an attack or not, but not knowing is reason enough to be skeptical about the use of AES.*

— Bruce Schneier

MACQUARIE University

The XSL Attack

The eXtended Spare Linearization attack (Courtois & Pieprzyk, 2002) is an example of an algebraic attack on Rijndael.

While it is faster than a brute-force exhaustive search, and requires very few plaintexts to perform, it still has a high work factor.

It works by deriving a system of quadratic simultaneous equations which are solved for the key – for AES-128, this is 8,000 equations with 1600 variables.

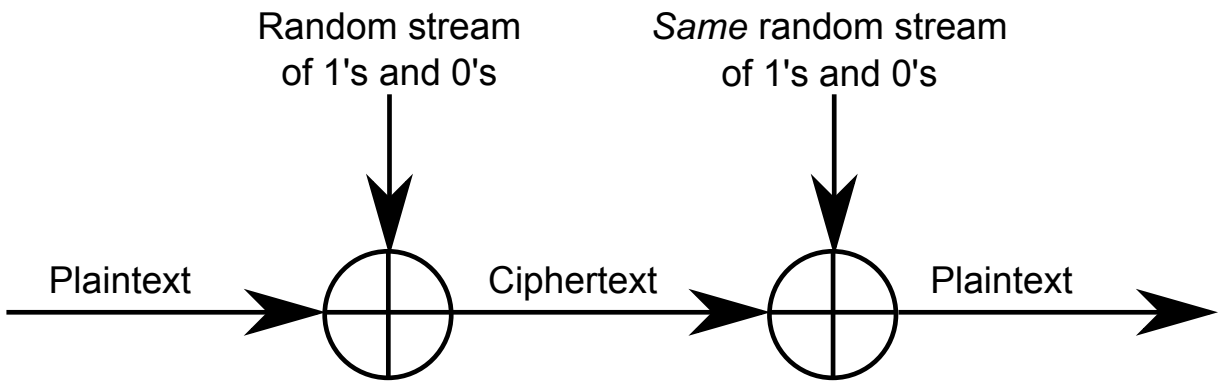
Key Generation and Stream Ciphers

The generation of random numbers is too important to be left to chance.

Robert E. Coveyou, Oak Ridge National Laboratory

Arranging a Stream Cipher

Remember this?



It's the one-time pad, a.k.a. Vernam encryption.

The $64,000 question: how can we arrange for the *same* random stream of 1's and 0's at both ends of our communications link? Another related, but even *more* important question: how can we choose truly random keys?

MACQUARIE University

Sources of Randomness

There are many sources of *randomness* or *entropy* in the natural world:

- The emission of particles in radioactive decay
- Thermal noise in integrated circuits
- Sound samples in a noisy environment
- Digitized images of a lava lamp

The problem is, these are mostly not accessible without special hardware (although Intel has built a thermal-noise based RNG into their latest processors)

MACQUARIE University

Hardware-Independent Techniques

There are some techniques which have been used in operating systems (close to the hardware):

- Precise measurement of the timing effects of air turbulence on the movement of hard drive heads
- Timing of keystrokes as user enters password
- Sampling of rapid mouse movement
- Timing of memory accesses under artificially-induced virtual memory thrashing
- Timing of disk I/O
- Measurements of skew between two system timers – one hardware, one software

Advice from RFC 1950

- Avoid relying on the system clock - it's often surprisingly non-random and not very accurate
- Do not use Ethernet addresses or hardware serial numbers - these numbers are often structured and guessable
- Beware of using information such as arrival times of network packets. If the adversary is sending those packets, he can control their timing. . .
- Do not use random selections from a large database (like a CD-ROM or DVD). The attacker could have access to the same database, and it may also have a guessable structure
- Consider using analog inputs to the system (e.g. `/dev/audio`) `along with compression (to remove systematic skew)`
  - `e.g. cat /dev/audio | compress - > random-bit-stream`

Bad Examples - Netscape

The old Netscape browser RNG used the current time and process ID (both very predictable):

```
a = mixbits( time.tv_usec );
b = mixbits( getpid() + time.tv_sec + (getppid() << 12 ));
seed = MD5(a, b);

nonce = MD5( seed++ );
key = MD5( seed++ );
```

But PID's start at 1, the time is guessable, and both fall within a limited range of values.
Result: 128-bit session keys had - at best - 47 bits of entropy.

MACQUARIE University

Bad Examples - Kerberos V4

The Kerberos V4 code was even worse – for a start, it called `random()`, rather than using `MD5()`.

```
MIT_MAGIC_COOKIE:
    key = rand % 256;
```

So it can have only 256 possible values!

Another bad example is Firewall-1; it calls `time()` to generate a key and then refreshes it after 99 uses.

Bad Examples - Programming Language Libraries

Programming language libraries often provide a `rand()` function — but they usually use a linear congruential RNG:

$$x_{n+1} = (ax_n + c) \mod m$$

For example, Linux's glibc uses
$a = 1103515245, c = 12345, m = 2^{32}$.

The same technique is used in (e.g.) pocket calculators to generate floating-point values on the interval $(0, 1)$.

This gives a uniform distribution but . . . Oh! So *predictable!*

MACQUARIE University

Cryptographic Randomness vs Statistical Randomness

This illustrates the difference between statistical notions of randomness, which almost *require* a notion of predictability and the cryptographic idea of randomness, which is entirely concerned with *unpredictability*.

Randomness and entropy are not physical quantities, but rather walk a very slippery line between being physical and philosophical entities.

MACQUARIE University

Linear Congruential Generators

- Fast - few operations per bit, and those are fast on modern processors
- With good constants:
  - Provide maximal periodicity
  - Pass spectral test for randomness
- But are predictable!
- Use only for non-cryptographic applications, e.g. simulations, Monte-Carlo Method, etc.

Other Epic FAILS with LCRNG's

Using a LCRNG with DSA (Digital Signature Algorithm) makes it possible to recover the signer's secret key after seeing only three signatures.

The DSA code in Sun's JDK 1.1 for Java was even worse, using a *constant value* for one of the DSA parameters, so that the key could be recovered if DSA was used more than once!

Kevin Mitnick *notoriously* exploited a vulnerability in the BSD Unix TCP/IP stack - which simply added a *constant value* (128K) to the initial sequence number for each connection - to perform source IP address spoofing.

MACQUARIE University

Linear Feedback Shift Registers

Reference: Menezes, et. al. [MVOV97], Chapter 9

Linear Feedback Shift Registers

- Easy to implement in hardware
- Maximal periodicity
- Produce sequences of good statistical properties
- Readily analyzable using algebraic techniques

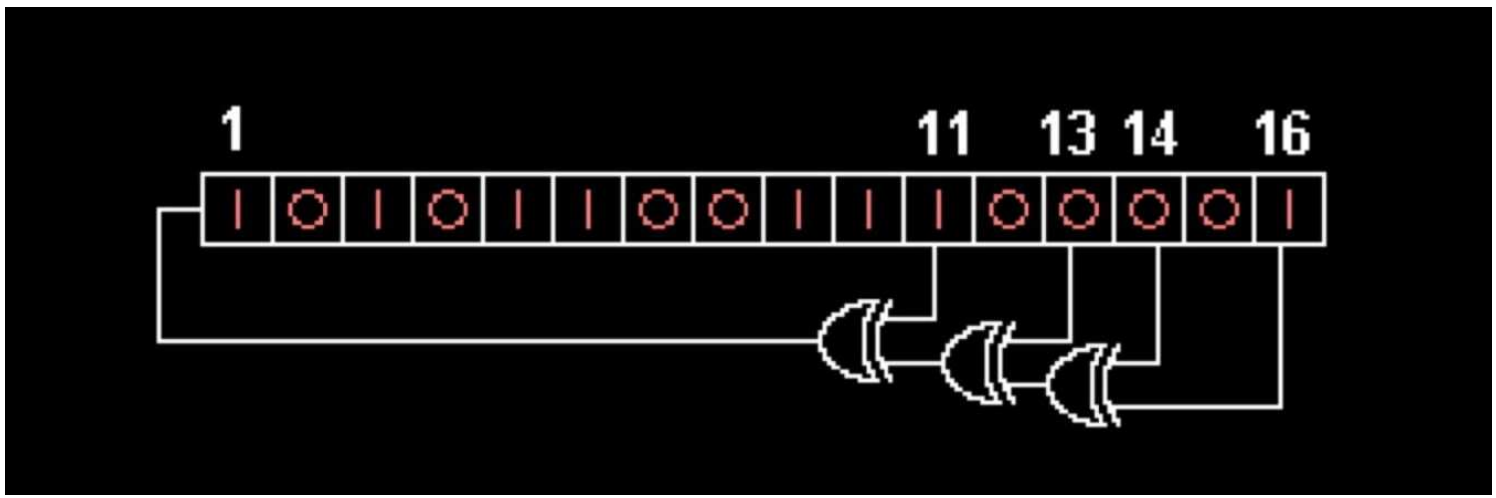
Stream Ciphers

Constructions based on

- linear feedback shift registers (LFSR)
- nonlinear feedback shift registers (NLFSR)
- arrays

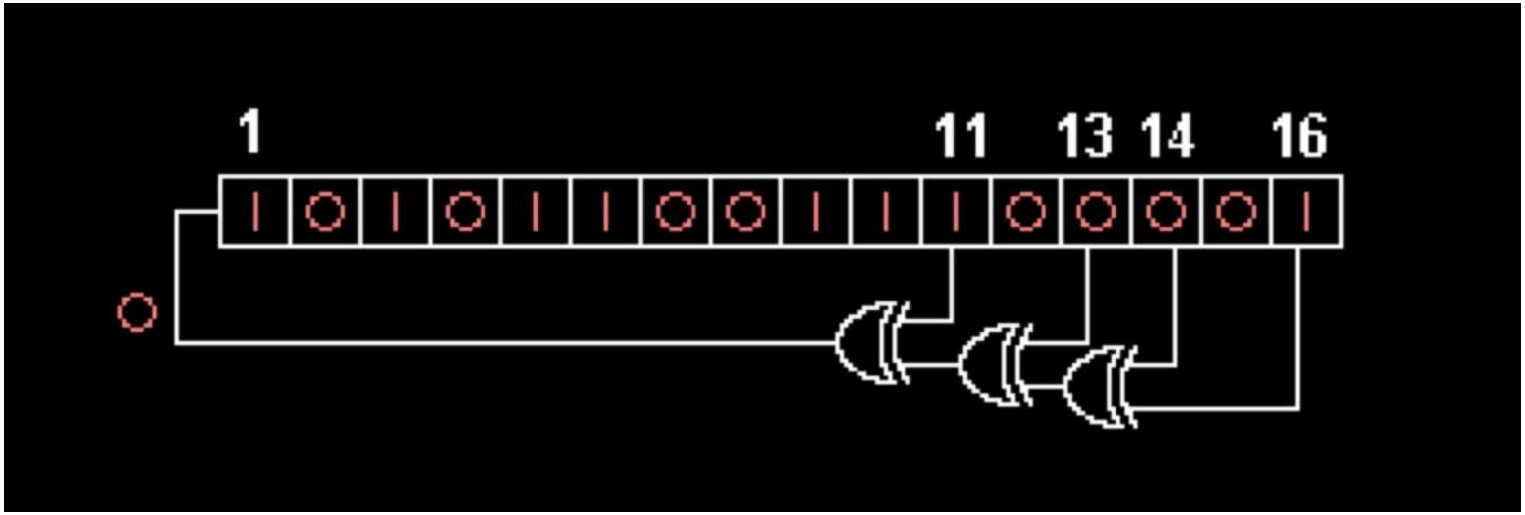
Stream Ciphers

LFSR - how it works



MACQUARIE University

Stream Ciphers

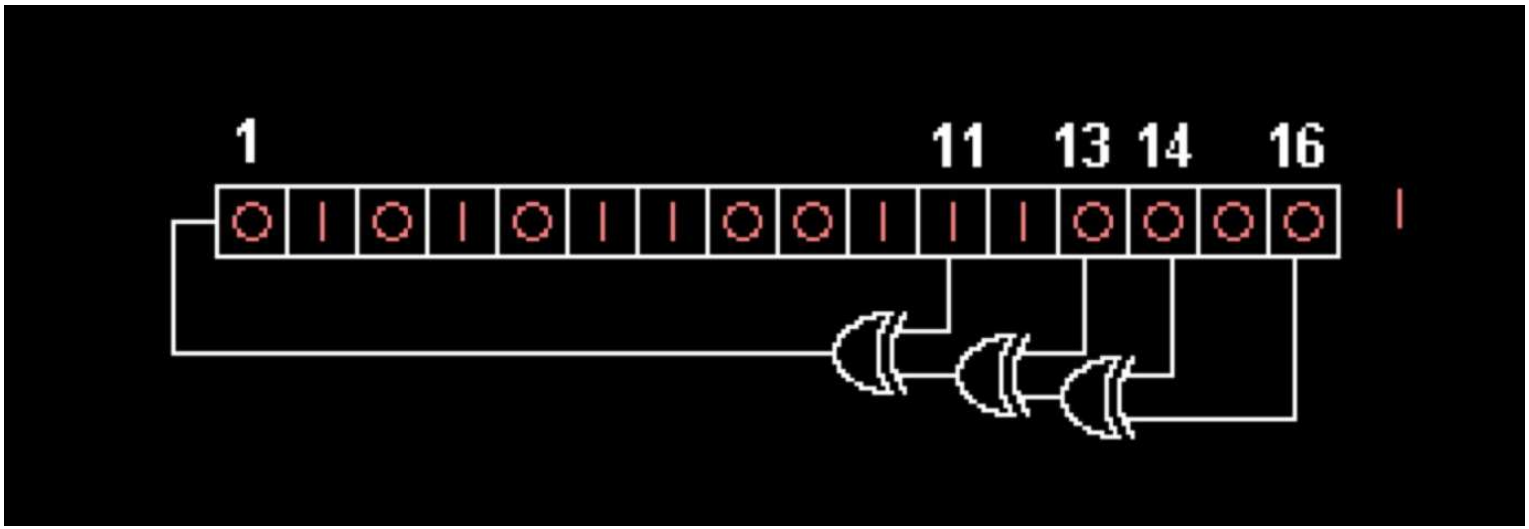


If we XOR bits on positions:

- 11
- 13
- 14
- 16

then we get 0

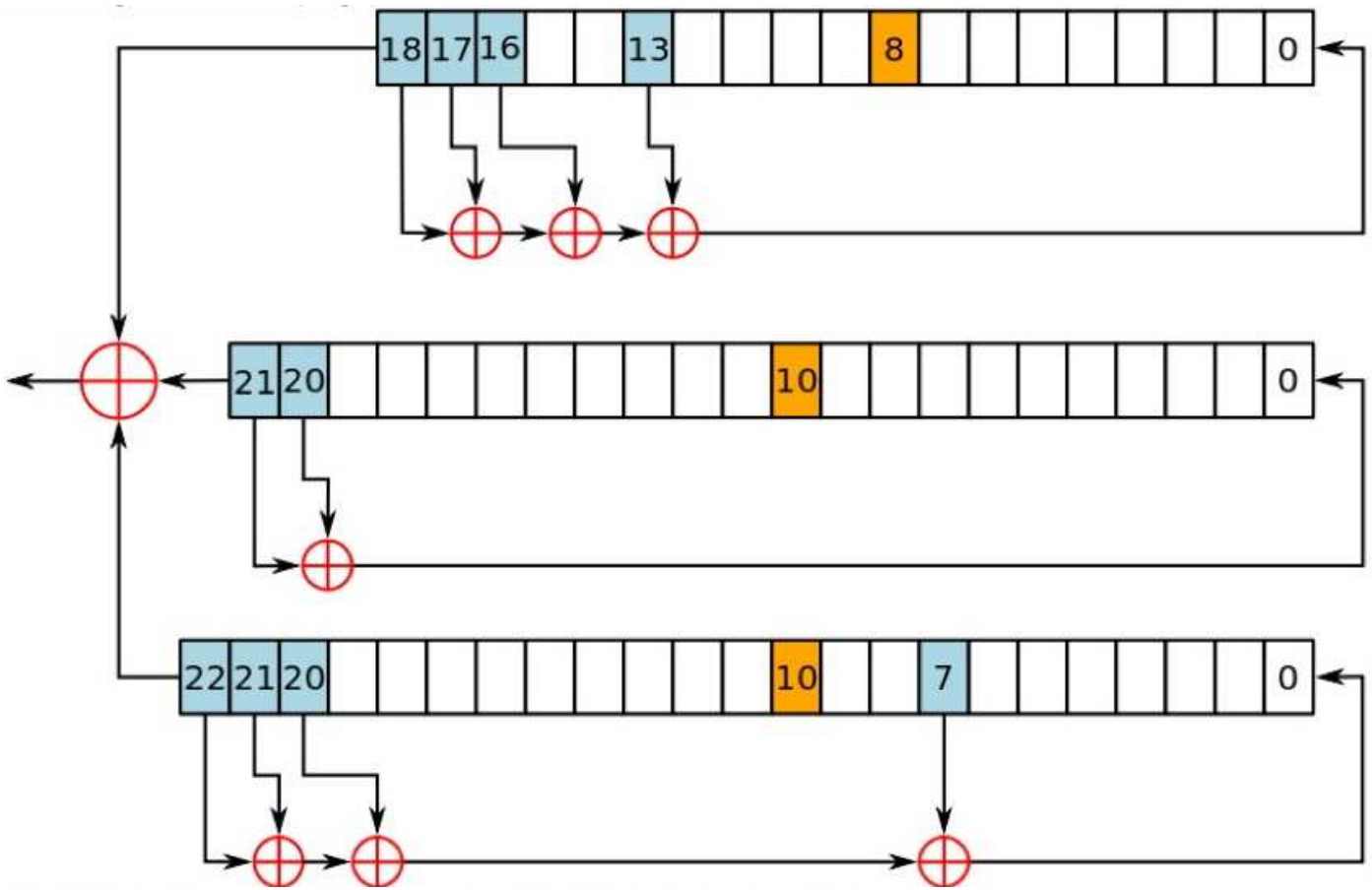
Stream Ciphers



In this case the output is 1

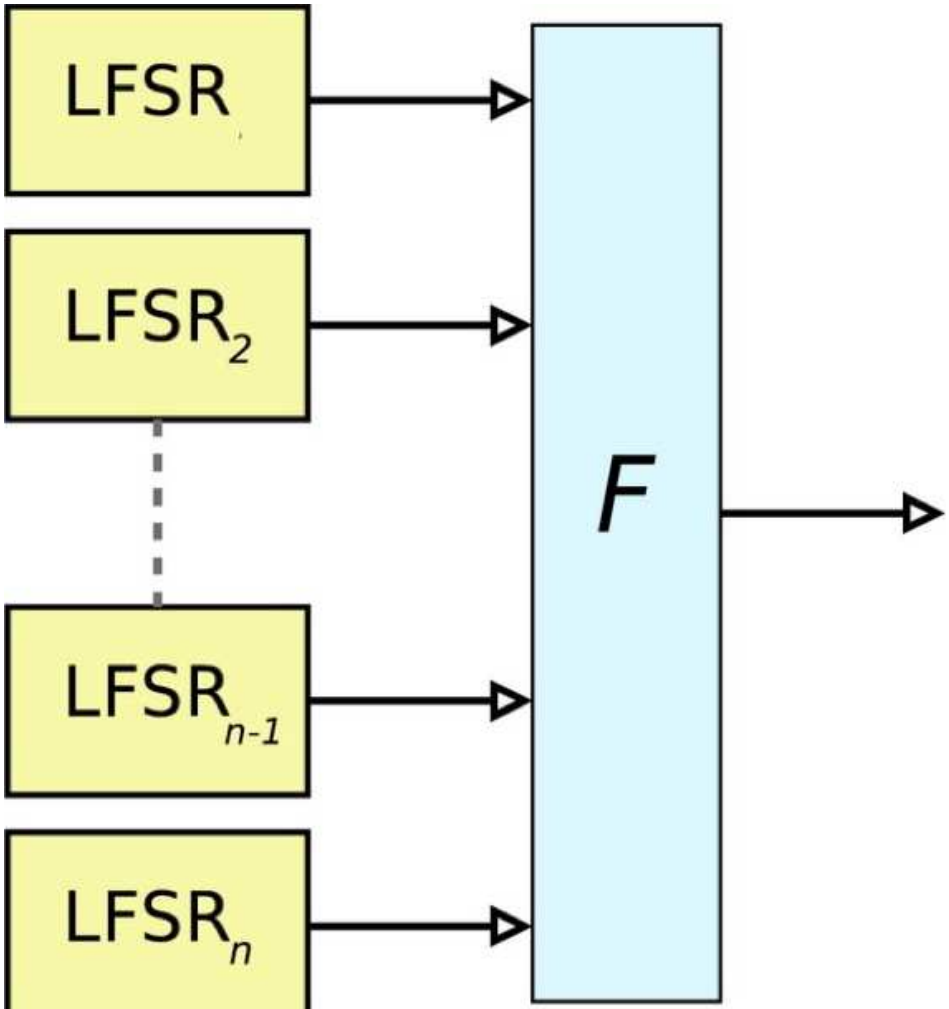
Stream Ciphers

Linear combination of LFSRs – A5/1 Cipher



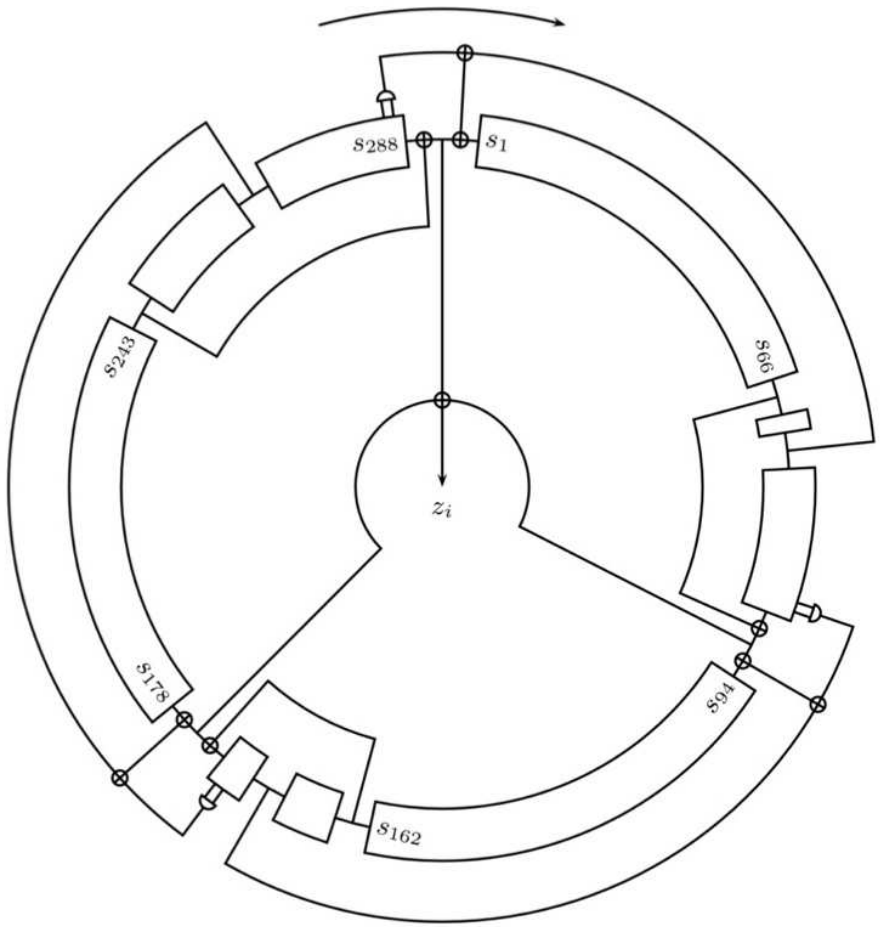
Stream Ciphers

Combination of LFSR with non-linear filter F



Stream Ciphers

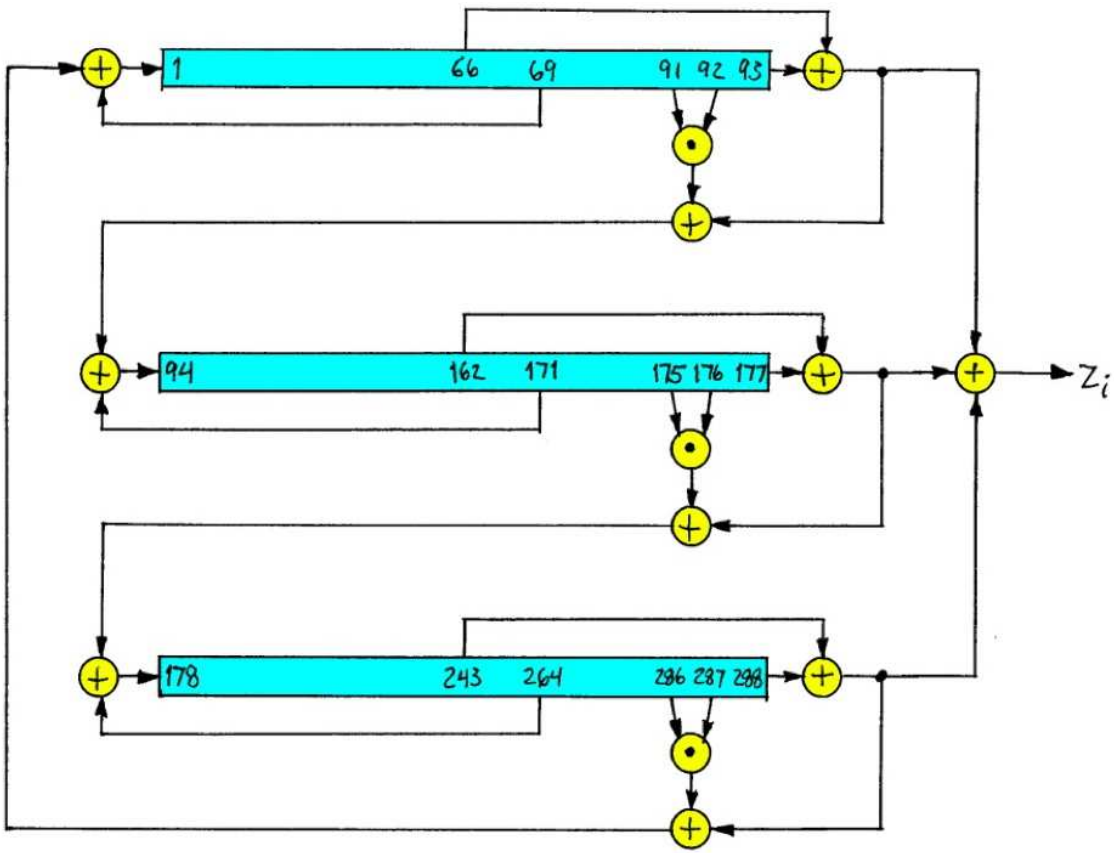
Trivium - three NLFSRs



MACQUARIE University

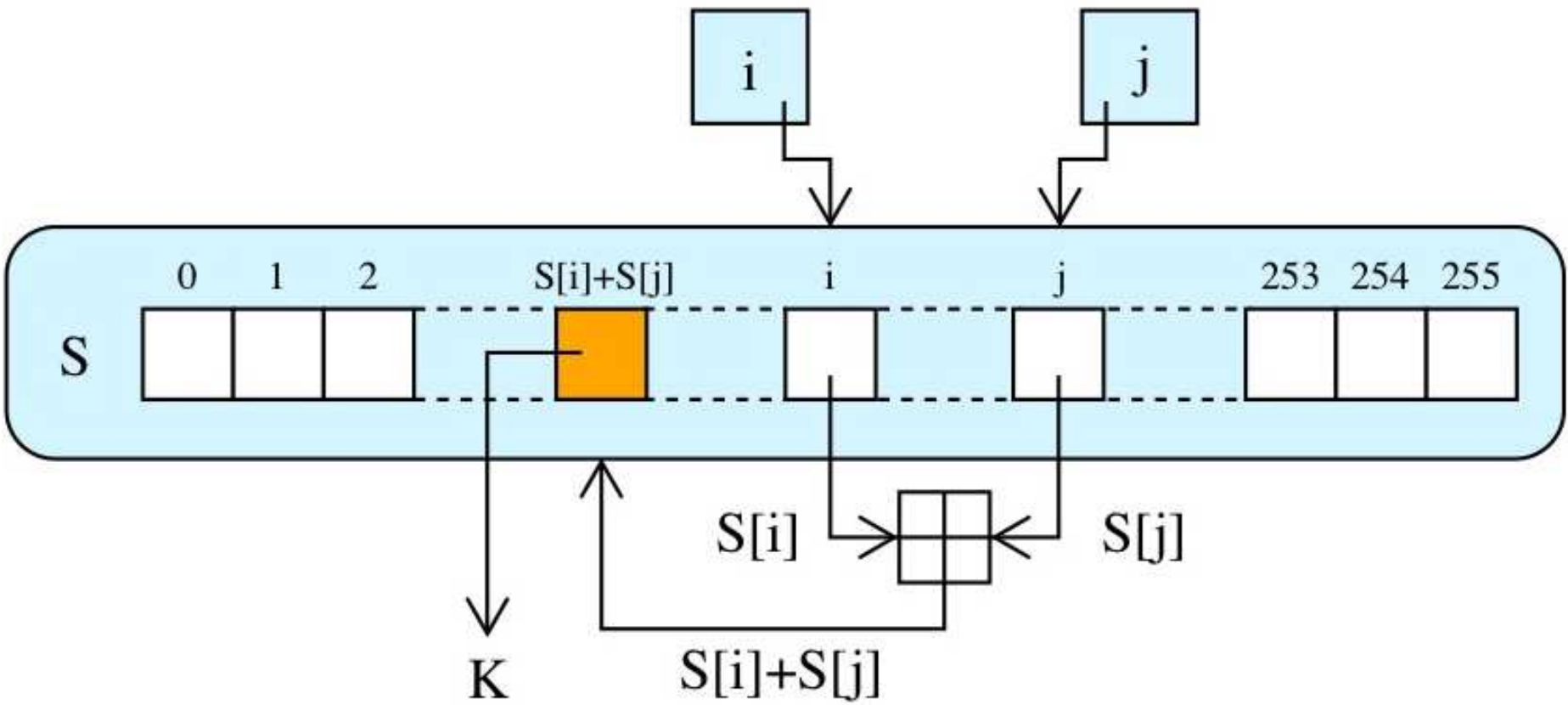
Stream Ciphers

Trivium - three NLFSRs



Stream Ciphers

RC-4 – array based cipher



Other common stream ciphers

- A5/1 - Stream cipher used in GSM cellphones; now regarded as too weak for serious use
- GMR-2 - Stream cipher used in satphones; also now cracked
- Salsa20 - Stream cipher developed by DJB
- ChaCha - Derivation of Salsa20, used as a replacement for RC4 by Google and also used for filesystem encryption in low-end Android phones

These are predominantly utilised in low-power devices where battery life is important. But cellphone processors and batteries have improved to the point where mostly, block ciphers are used.

DES - Modes of Operation

There are many modes of operation:

- electronic codebook mode (ECB)
- cipher block chaining mode (CBC)
- cipher feedback mode (CFB)
- output feedback mode (OFB)
- counter mode (CTR)
- Galois counter mode (GCM)

And even more, like XEX and XTS, which we will discuss later.

MACQUARIE University

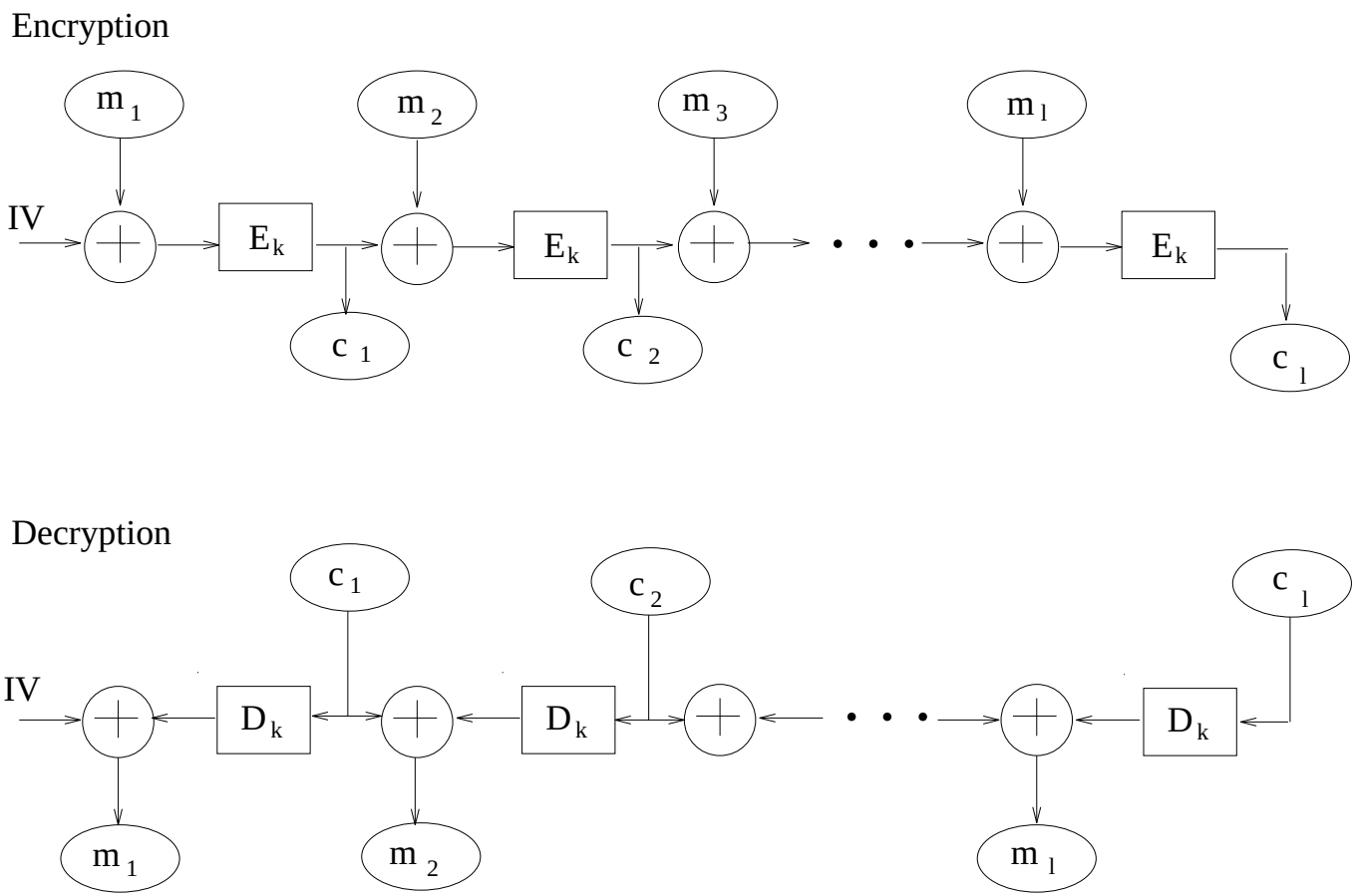
Electronic Codebook Mode

In the ECB mode, a data block m of arbitrary length is divided into 64-bit blocks $m_1, m_2, \ldots, m_\ell$. The last block, if it is shorter than 64 bits, needs to be padded to the full length of 64 bits. Later the DES algorithm is applied independently for each block using the same cryptographic key k so the ciphertext

$$c = (c_1, \ldots, c_\ell) = (E_k(m_1), \ldots, E_k(m_\ell))$$

consists of ℓ independent cryptograms each related to a single message m_i.

Cipher Block Chaining (CBC) Mode



Cipher Block Chaining (CBC) Mode

For encryption, cryptograms are created for the current message block and the previous cryptogram according to the following equation

$$c_i = E_k(m_i \oplus c_{i-1})$$

where $c_1 = E_k(m_1 \oplus IV)$ and $i = 2, \ldots, \ell$. The decryption process unravels the ciphertext

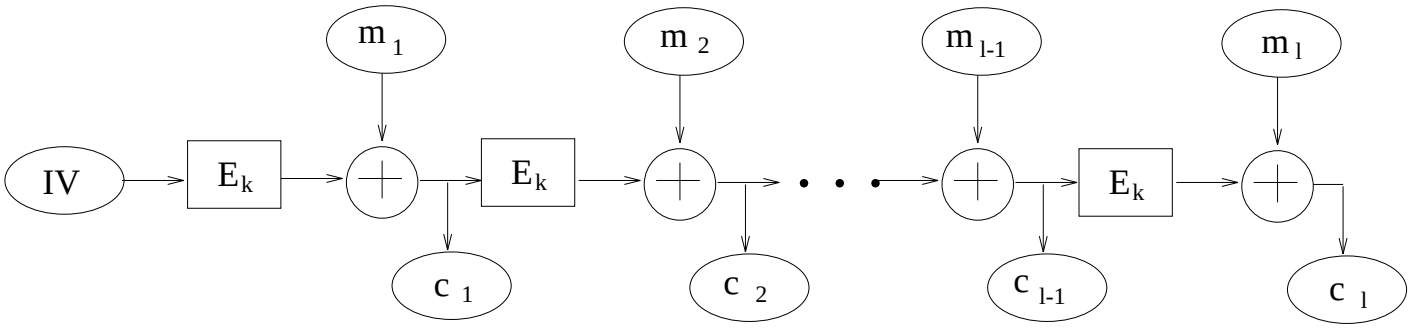
$$m_i = D_k(c_i) \oplus c_{i-1}$$

for $i = 2, \ldots, \ell$ and $m_1 = D_k(c_1) \oplus IV$.

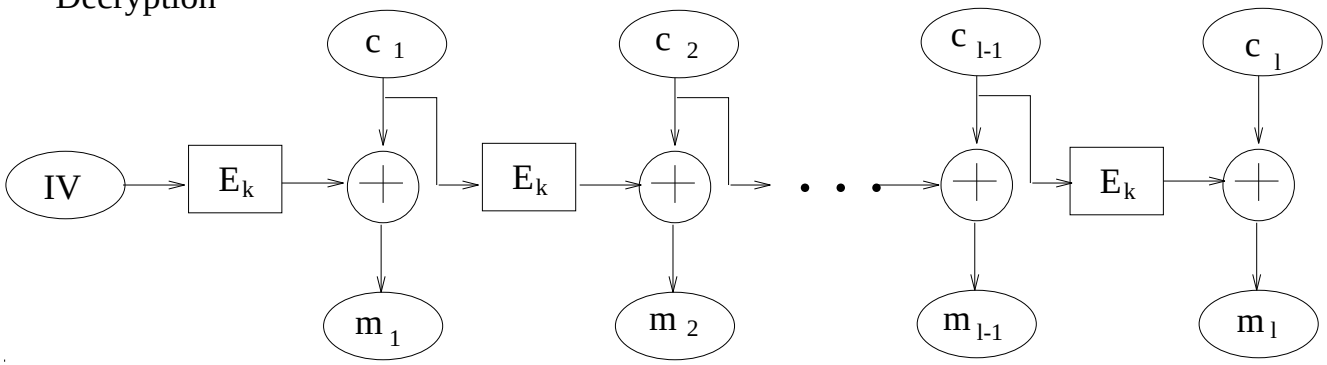
MACQUARIE University

Cipher Feedback (CFB) Mode

Encryption



Decryption



MACQUARIE University

Cipher Feedback (CFB) Mode

In the CFB mode, cryptograms are equal to

$$c_i = m_i \oplus E_k(c_{i-1})$$

where $c_1 = m_1 \oplus E_k(IV)$ and $i = 2, \ldots, \ell$. The decryption uses E_k function as well therefore

$$m_i = c_i \oplus E_k(c_{i-1}).$$

and the decryption D_k is never used.

MACQUARIE University

Output Feedback (OFB) Mode

Note that the sequence $E_k(c_i)$ mimics a random key in the one-time pad system. If the pseudorandom string $E_k(c_i)$ $(i = 1, \ldots, \ell)$ is simplified to the string $E_k^i(IV)$, then this mode of operation becomes OFB where $E_k^i = \underbrace{E_k \circ E_k \circ \ldots \circ E_k}_{i}$.

Counter Mode

In *counter mode*, the feedback register is filled from a counter - after each block encryption, the counter increments, often by one.

References


Mitsuru Matsui.
Linear Cryptanalysis Method for DES Cipher.
In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT '93*, number 765 in Lecture Notes in Computer Science, pages 386–397. Springer Berlin Heidelberg, January 1994.


A. J Menezes, Paul C Van Oorschot, and Scott A Vanstone.
Handbook of applied cryptography.
CRC Press, Boca Raton, 1997.


Claude E Shannon.
Communication Theory of Secrecy Systems.
Bell System Technical Journal, 28(4):656–715, 1949.

MACQUARIE University