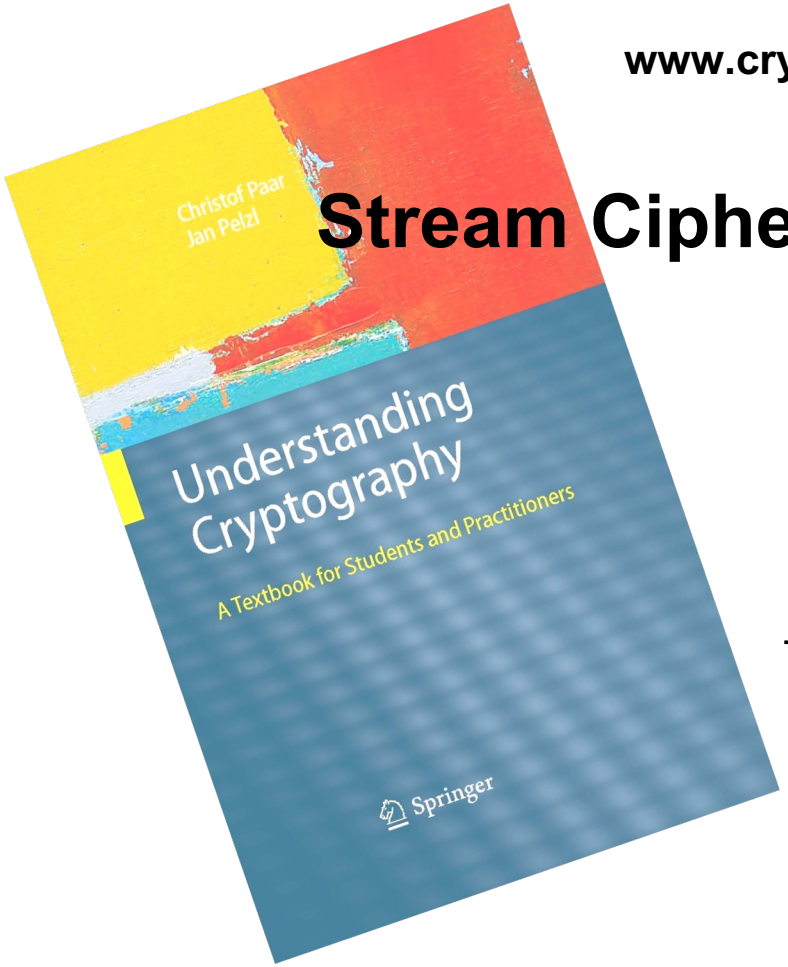
Understanding Cryptography – A Textbook for Students and Practitioners

by Christof Paar and Jan Pelzl

www.crypto-textbook.com



Stream Ciphers & Block Ciphers

These slides are a modified version of the slides by Thomas Eisenbarth, Christof Paar and Jan Pelzl.

Some legal stuff (sorry): Terms of Use

- The slides can used free of charge. All copyrights for the slides remain with the authors.
- The title of the accompanying book “Understanding Cryptography” by Springer and the author’s names must remain on each slide.
- If the slides are modified, appropriate credits to the book authors and the book title must remain within the slides.
- It is not permitted to reproduce parts or all of the slides in printed form whatsoever without written consent by the authors.

Outline

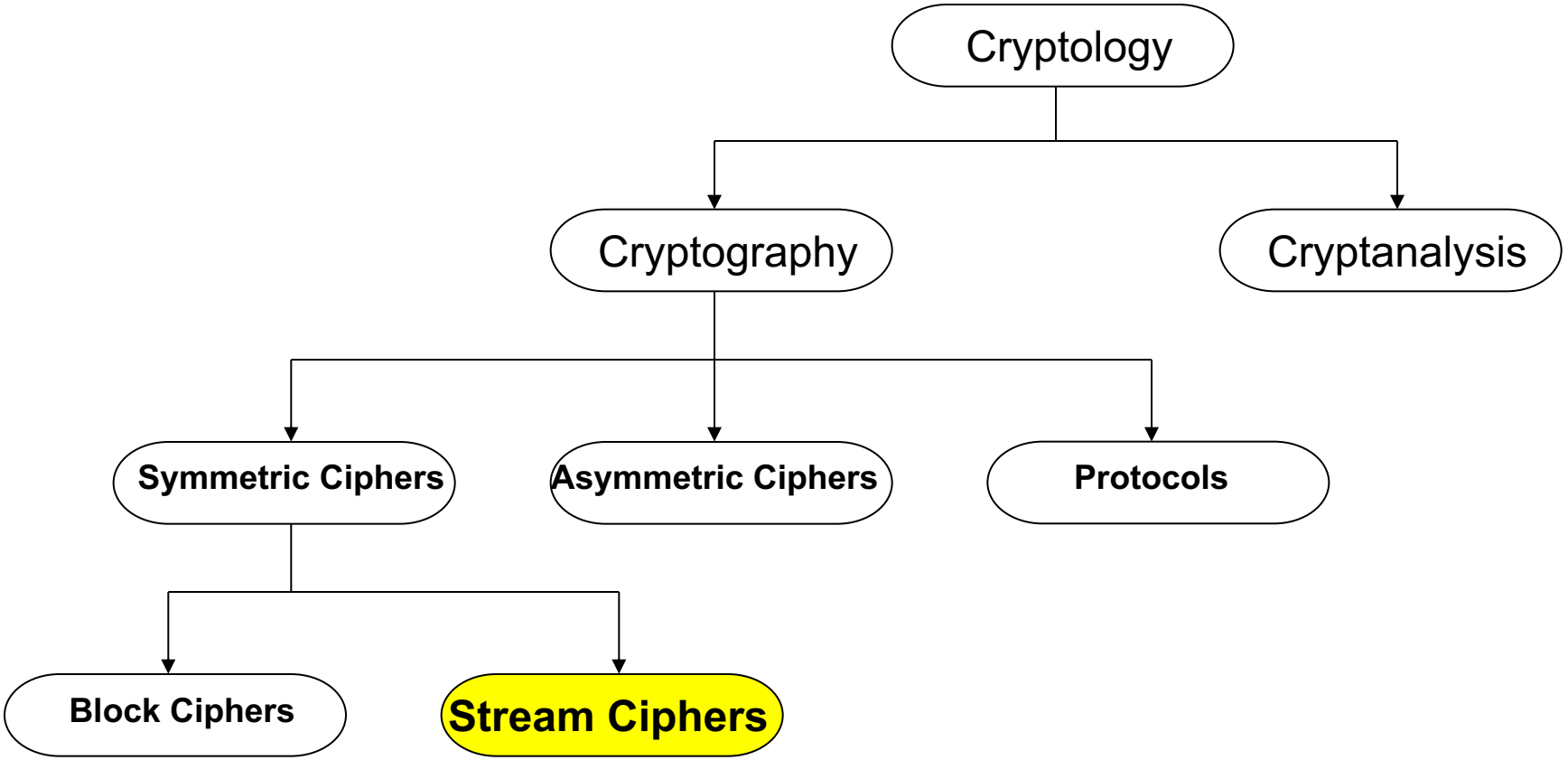
- Stream ciphers
- Block ciphers
- Encryption mode

Stream Ciphers

Content of this Chapter

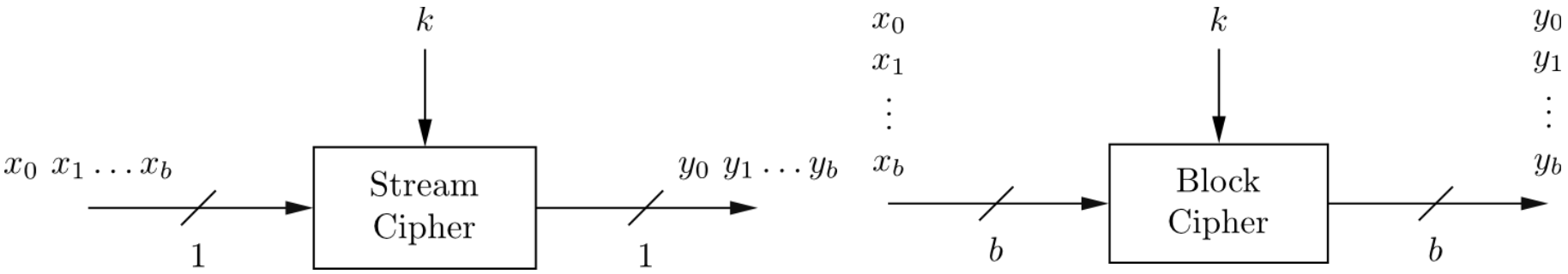
- **Intro to stream ciphers**
- Random number generators (RNGs)
- One-Time Pad (OTP)

Stream Ciphers in the Field of Cryptology



Stream Ciphers were invented in 1917 by Gilbert Vernam

Stream Cipher vs. Block Cipher

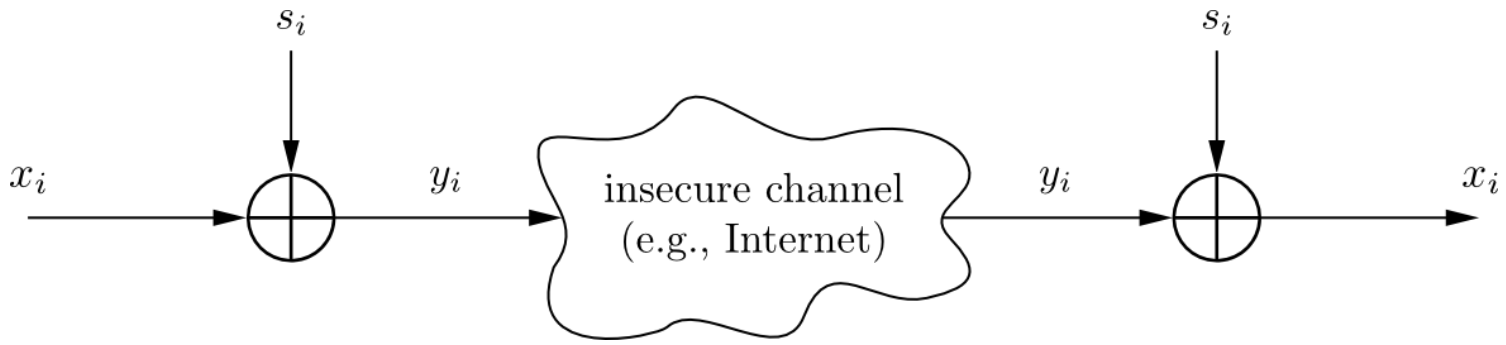


- **Stream Ciphers**
  - Encrypt bits individually
  - Usually small and fast → common in embedded devices (e.g., A5/1 for GSM phones)

- **Block Ciphers:**
  - Always encrypt a full block (several bits)
  - Are common for Internet applications

Encryption and Decryption with Stream Ciphers

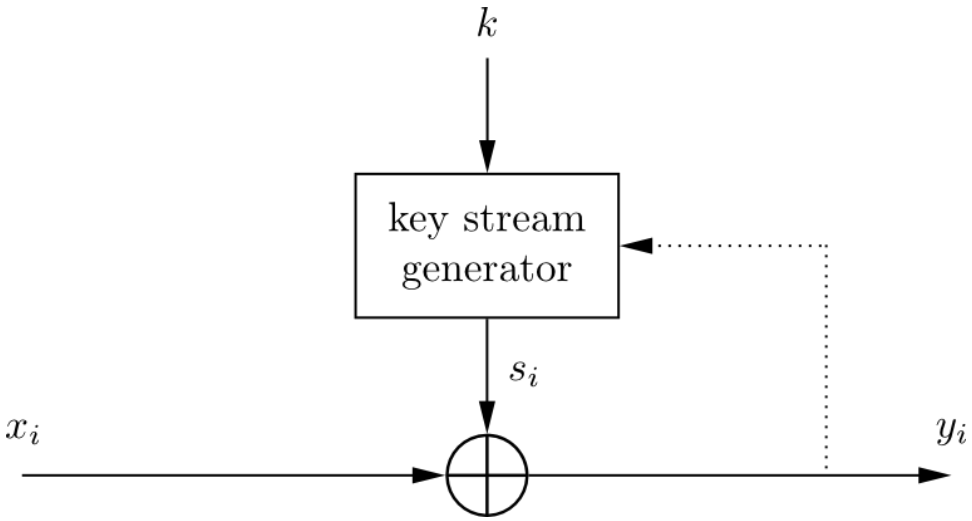
Plaintext x_i, ciphertext y_i and key stream s_i consist of individual bits



- Encryption and decryption are simple additions modulo 2 (aka XOR)
- Encryption and decryption are the same functions

- **Encryption:** $y_i = e_{s_i}(x_i) = x_i + s_i \; mod \; 2 \qquad x_i, yi, s_i \in \{0,1\}$
- **Decryption:** $x_i = e_{s_i}(y_i) = y_i + s_i \; mod \; 2$

Synchronous vs. Asynchronous Stream Cipher



- Security of stream cipher depends entirely on the key stream s_i :
  - Should be **random** , i.e., $Pr(s_i = 0) = Pr(s_i = 1) = 0.5$
  - Must be **reproducible** by sender and receiver

- **Synchronous Stream Cipher**
  - Key stream depend only on the key (and possibly an initialization vector IV)

- **Asynchronous Stream Ciphers**
  - Key stream depends also on the ciphertext (dotted feedback enabled)

Why is Modulo 2 Addition a Good Encryption Function?

- Modulo 2 addition is equivalent to XOR operation
- For perfectly random key stream s_i, each ciphertext output bit has a 50% chance to be 0 or 1

  → Good statistic property for ciphertext
- Inverting XOR is simple, since it is the same XOR operation

x_i	s_i	y_i
0	0	0
0	1	1
1	0	1
1	1	0

Stream Cipher: Throughput

Performance comparison of symmetric ciphers (Pentium4):

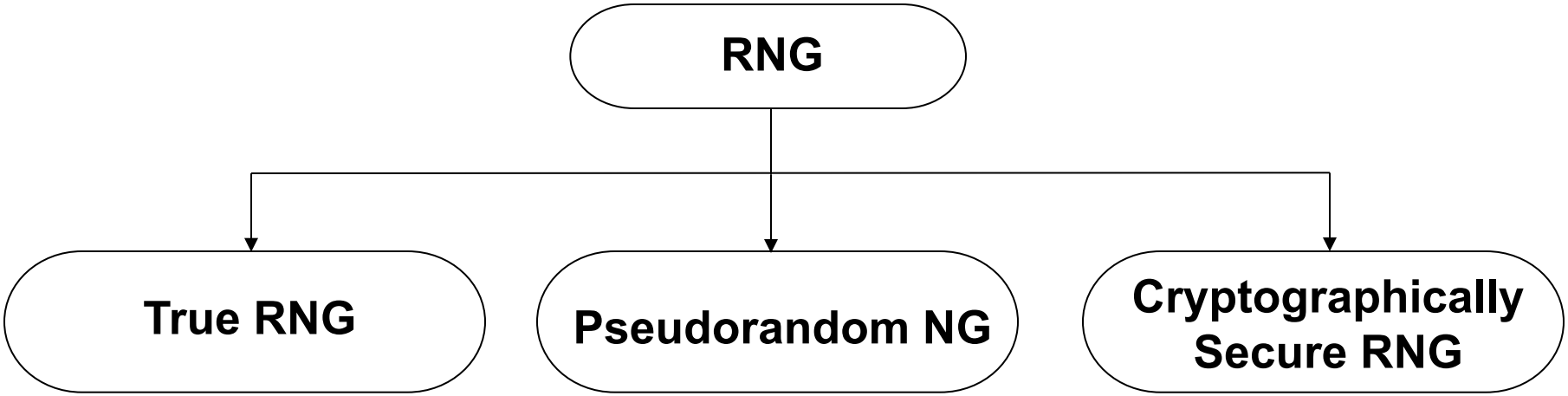
Cipher	Key length	Mbit/s
DES	56	36.95
3DES	112	13.32
AES	128	51.19
RC4 (stream cipher)	(choosable)	211.34

Source: Zhao et al., Anatomy and Performance of SSL Processing, ISPASS 2005

Content of this Chapter

- Intro to stream ciphers
- **Random number generators (RNGs)**
- One-Time Pad (OTP)

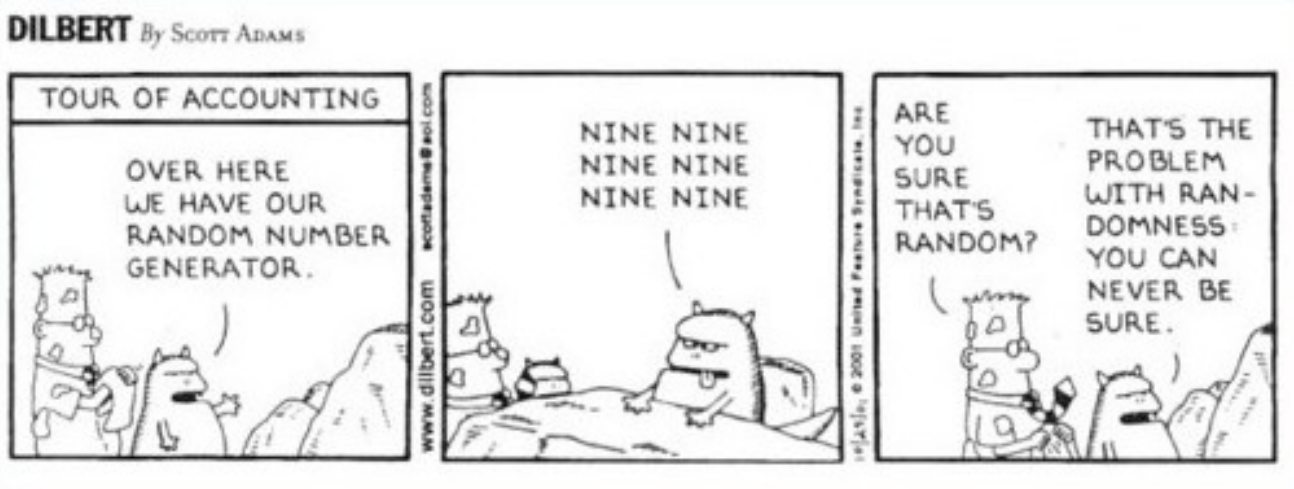
Random number generators (RNGs)



True Random Number Generators (TRNGs)

- Based on physical random processes: coin flipping, dice rolling, semiconductor noise, radioactive decay, mouse movement, clock jitter of digital circuits
- Output stream s_i should have good statistical properties:
  $\Pr(s_i = 0) = \Pr(s_i = 1) = 50\%$ (often achieved by post-processing)
- Output can neither be predicted nor be reproduced

Typically used for generation of keys, nonces (used only-once values) and for many other purposes



Pseudorandom Number Generator (PRNG)

- Generate sequences from initial seed value
- Typically, output stream has good statistical properties
- Output can be reproduced and can be predicted

Often computed in a recursive way:

$$s_0 = seed$$

$$s_{i+1} = f(s_i, s_{i-1}, \ldots, s_{i-t})$$

Example: *rand() function* in ANSI C:

$$s_0 = 12345$$

$$s_{i+1} = 1103515245 s_i + 12345 \bmod 2^{31}$$

Most PRNGs have bad cryptographic properties!

Cryptanalyzing a Simple PRNG

Simple PRNG: **Linear Congruential Generator**

$$S_0 = seed$$

$$S_{i+1} = AS_i + B \bmod m$$

Assume

- unknown *A*, *B and* S_0 as key
- Size of *A*, *B* and S_i to be 100 bit
- 300 bit of output are known, i.e. S_1, S_2 and S_3

Solving

$$S_2 = AS_1 + B \bmod m$$

$$S_3 = AS_2 + B \bmod m$$

…directly reveals A and B. All S_i can be computed easily! (See page Example 2.2 in *Understanding Cryptography* for more details).

Bad cryptographic properties due to the linearity of most PRNGs

Cryptographically Secure Pseudorandom Number Generator (CSPRNG)

- Special PRNG with additional property:
  - Output must be **unpredictable**

More precisely: Given n consecutive bits of output s_i , the following output bits s_{n+1} cannot be predicted (in polynomial time).

- Needed in cryptography, in particular for stream ciphers

- Remark: There are almost no other applications that need unpredictability, whereas many, many (technical) systems need PRNGs.

Content of this Chapter

- Intro to stream ciphers
- Random number generators (RNGs)
- **One-Time Pad (OTP)**

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

- A cryptosystem is unconditionally secure if it cannot be broken even with *infinite* computational resources

One-Time Pad

- A cryptosystem developed by Mauborgne that is based on Vernam's stream cipher:
- Properties:

  Let the plaintext, ciphertext and key consist of individual bits $x_i, y_i, k_i \in \{0,1\}$.

Encryption:	$e_{k_i}(x_i) = x_i \oplus k_i$
Decryption:	$d_{k_i}(y_i) = y_i \oplus k_i$

OTP is unconditionally secure if and only if the key k_i is used only once!

One-Time Pad (OTP)

Unconditionally secure cryptosystem:

$$y_0 = x_0 \oplus k_0$$
$$y_1 = x_1 \oplus k_1$$
$$\vdots$$

Every equation is a linear equation with two unknowns

$\Rightarrow$ for every y_i, $x_i = 0$ and $x_i = 1$ are equiprobable!

$\Rightarrow$ This is true if and only if $k_0, k_1, \ldots$ are independent, i.e., all k_i have to be generated truly random

$\Rightarrow$ It can be shown that this systems can *provably* not be solved.

Disadvantage: For almost all applications the OTP is **impractical** since the key must be as long as the message! (Imagine you have to encrypt a 1GByte email attachment.)

Lessons Learned

- Stream ciphers are less popular than block ciphers in most domains such as Internet security. There are exceptions, for instance, the popular stream cipher RC4.

- Stream ciphers sometimes require fewer resources, e.g., code size or chip area, for implementation than block ciphers, and they are attractive for use in constrained environments such as cell phones.

- The requirements for a *cryptographically secure pseudorandom number generator* are far more demanding than the requirements for pseudorandom number generators used in other applications such as testing or simulation

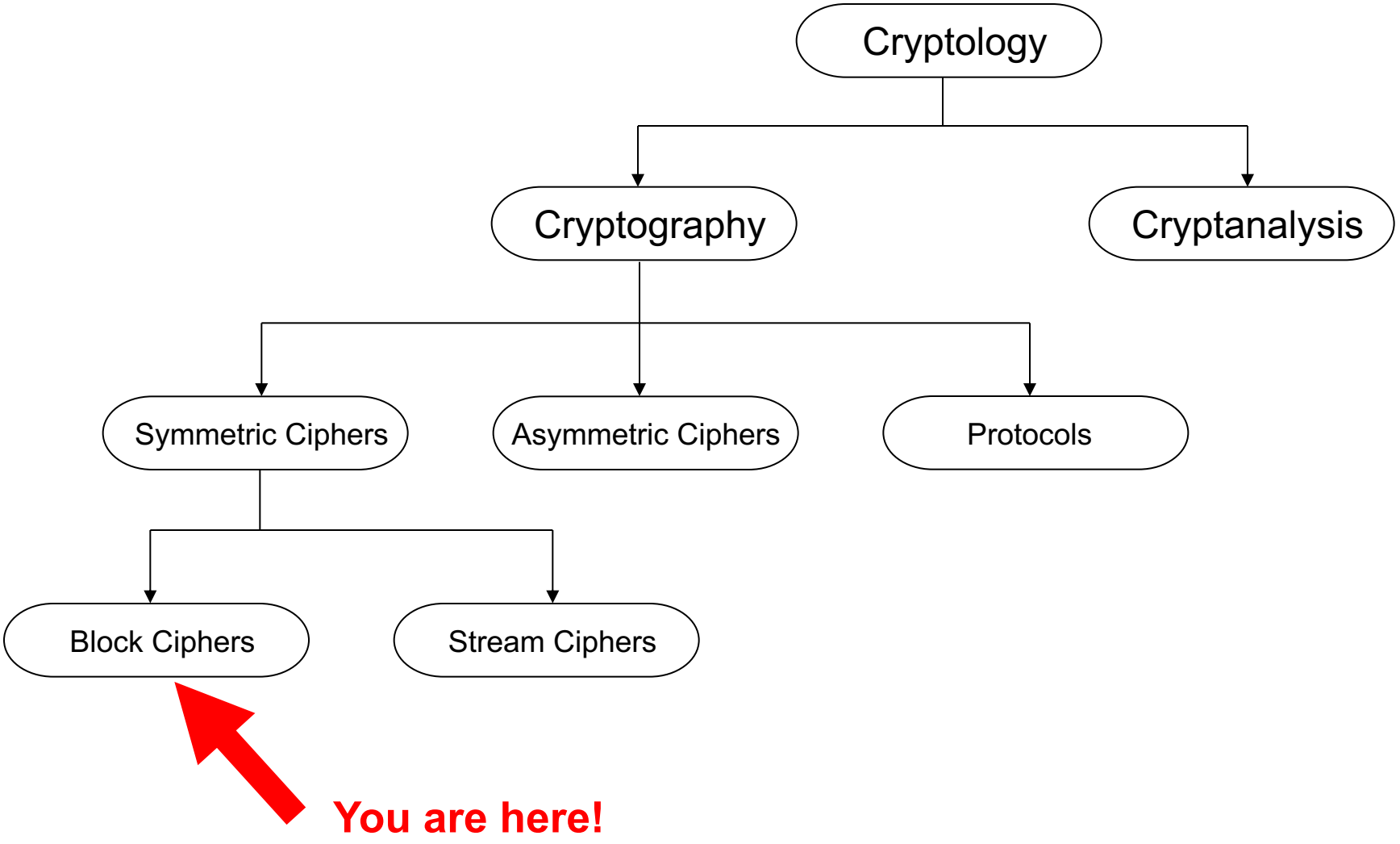
- The One-Time Pad is a provable secure symmetric cipher. However, it is highly impractical for most applications because the key length has to equal the message length.

BLOCK CIPHERS

Content of this Chapter

- **Overview of block ciphers**
- Overview of the AES algorithm
- Galois Field
- Internal structure of AES
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- Decryption
- Practical issues

Block Ciphers in the Field of Cryptology



Block Ciphers

- A block cipher is much more than just an encryption algorithm, it can be used ...
  - to build different types of block-based encryption schemes
  - to realize stream ciphers
  - to construct hash functions
  - to make message authentication codes
  - to build key establishment protocols
  - to make a pseudo-random number generator
  - ...

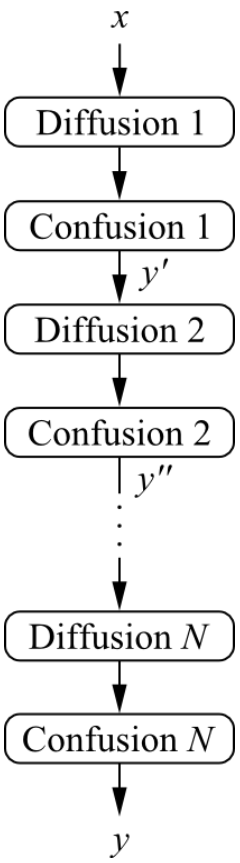
Block Cipher Primitives: Confusion and Diffusion

- Claude Shannon: There are two primitive operations with which strong encryption algorithms can be built:
  1. **Confusion:** An encryption operation where the **relationship between key and ciphertext is obscured**.
     Today, a common element for achieving confusion is **substitution**, which is found in both AES and DES.

  2. **Diffusion:** An encryption operation where the **influence of one plaintext symbol is spread over many ciphertext symbols** with the goal of hiding statistical properties of the plaintext.
     A simple diffusion element is the **bit permutation**, which is frequently used within DES.

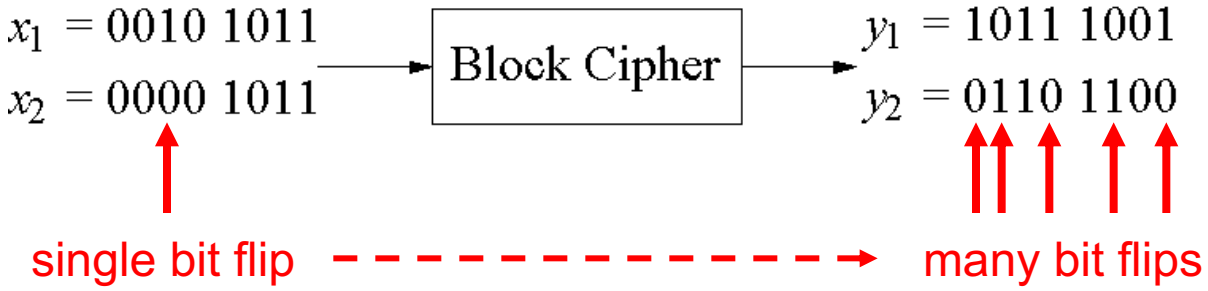
- Both operations by themselves cannot provide security. The idea is to concatenate confusion and diffusion elements to build so called *product ciphers.*

Product Ciphers



- Most of today's block ciphers are *product ciphers* as they consist of rounds which are applied repeatedly to the data.
- Can reach excellent diffusion: **changing of one bit of plaintext results** *on average* in the **change of half the output bits**.

Example:



Content of this Chapter

- Overview of block ciphers
- **Overview of the AES algorithm**
- Galois Field
- Internal structure of AES
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- Decryption
- Practical issues

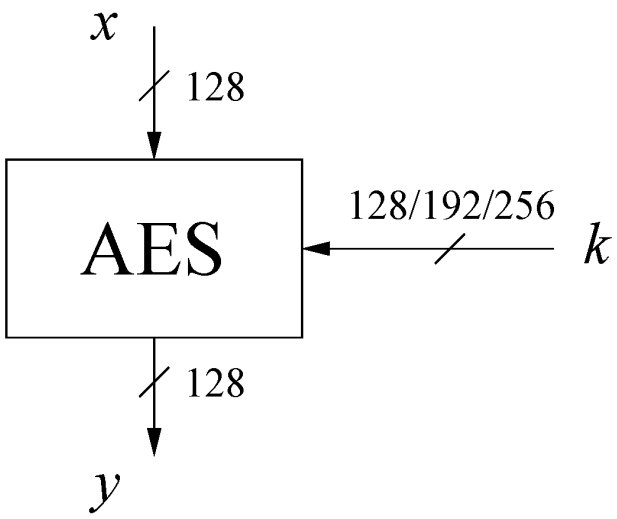
Some Basic Facts

- AES is the most widely used symmetric cipher today
- The algorithm for AES was chosen by the US *National Institute of Standards and Technology* (NIST) in a multi-year selection process
- The requirements for all AES candidate submissions were:
  - Block cipher with **128-bit block size**
  - **Three supported key lengths**: 128, 192 and 256 bit
  - Security relative to other submitted algorithms
  - **Efficiency** in software and hardware

Chronology of the AES Selection

- The need for a new block cipher announced by NIST in January, 1997
- 15 candidates algorithms accepted in August, 1998
- 5 finalists announced in August, 1999:
  - *Mars* – IBM Corporation
  - *RC6* – *RSA* Laboratories
  - *Rijndael* – J. Daemen & V. Rijmen
  - *Serpent* – Eli Biham et al.
  - *Twofish* – B. Schneier et al.
- In October 2000, *Rijndael* was chosen as the AES
- AES was formally approved as a US federal standard in November 2001

AES: Overview

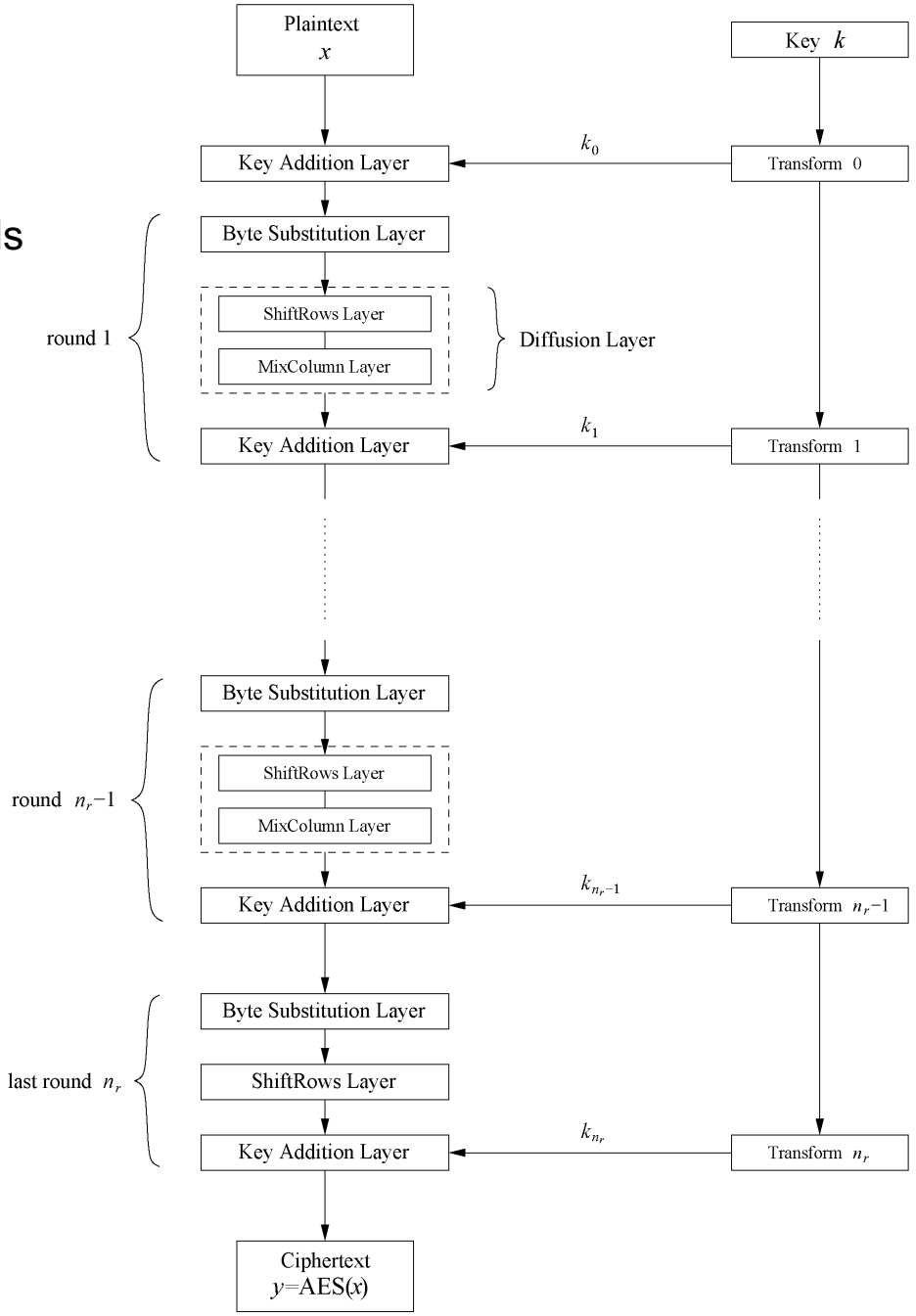


The number of rounds depends on the chosen key length:

Key length (bits)	Number of rounds
128	10
192	12
256	14

AES: Overview

- Iterated cipher with 10/12/14 rounds
- Each round consists of “Layers”



Content of this Chapter

- Overview of block ciphers
- Overview of the AES algorithm
- **Galois Field**
- Internal structure of AES
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- Decryption
- Practical issues

Galois Field

- *Galois field*, or *finite field*, is used in most layers of AES, especially Byte Substitution Layer (S-Box) and MixColumn layer.
- Roughly a finite field is a finite set of elements in which we can add, subtract, multiply and invert.
- To define a field, we first define an algebraic structure called a *group*.
- A field can be seen as an aggregate of two groups with a certain distributivity property.

Évariste Galois (1811 – 1832)
Source: Wikipedia

Group

A group *is a set of elements G together with an operation $\circ$ which combines two elements of G. A group has the following properties:*

1. *The group operation $\circ$ is* closed. *That is, for all $a, b, \in G$, it holds that $a \circ b = c \in G$.*
2. *The group operation is* associative. *That is, $a \circ (b \circ c) = (a \circ b) \circ c$ for all $a, b, c \in G$.*
3. *There is an element $1 \in G$, called the* neutral element *(or* identity element*), such that $a \circ 1 = 1 \circ a = a$ for all $a \in G$.*
4. *For each $a \in G$ there exists an element $a^{-1} \in G$, called the* inverse *of a, such that $a \circ a^{-1} = a^{-1} \circ a = 1$.*
5. *A group G is* abelian (or commutative) *if, furthermore, $a \circ b = b \circ a$ for all $a, b \in G$.*

Example of a group

- The set of integers Z_m = {0, 1, . . . , m − 1} and the operation addition modulo m form a group with the neutral element 0.
- Every element a has an inverse $-a$ such that $a + (-a) = 0 \bmod m$.
- Note that this set does not form a group with the operation multiplication because, for most value of m, most elements a do not have an inverse such that $a \cdot a^{-1} = 1 \, mod \, m$.
  - One exception to this is when m is a prime number, in which case all non-zero elements of Z_m has a multiplicative inverse.

Field (Definition)

A field F *is a set of elements with the following properties:*

- *All elements of* F *form an additive group with the group operation "+" and the neutral element 0.*
- *All elements of* F *except 0 form a multiplicative group with the group operation "*$\times$*" and the neutral element 1.*
- *When the two group operations are mixed, the distributivity law holds, i.e., for all* $a,b,c \in F$*:* $a(b+c) = (ab)+(ac)$.

Finite field

- A finite field (Galois field) is a field with a finite number of elements.
- The number of elements in a field is called the order or cardinality of the field

Theorem:
A field with order m exists if and only if m is a prime power, i.e., $m = p^n$, for some positive integer n and prime integer p. The prime p is called the characteristic of the finite field.

Prime field

- A simple example of a Galois field is the so-called prime fields.

- Given a prime p, the integer ring Z_p is a prime field:
  - Elements of the field are $0, \ldots, p-1$.
  - Addition and multiplication are always performed modulo p.
  - Multiplicative inverse always exists for non-zero elements of Z_p.

- A prime field of order p is often denoted by $GF(p)$.

An example of a prime field

- Consider the prime field GF(5):

addition

+	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

additive inverse

$$-0 = 0$$
$$-1 = 4$$
$$-2 = 3$$
$$-3 = 2$$
$$-4 = 1$$

multiplication

×	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

multiplicative inverse

0^{-1} does not exist
$$1^{-1} = 1$$
$$2^{-1} = 3$$
$$3^{-1} = 2$$
$$4^{-1} = 4$$

GF(2)

- This is a prime field of order 2: its elements are {0,1}.
- This is an important prime field used in AES.
- Addition is XOR, multiplication is AND.

addition

+	0	1
0	0	1
1	1	0

multiplication

×	0	1
0	0	0
1	0	1

Extension field

- In AES, the finite field contains 256 elements and denoted by $GF(2^8)$.
- Such a field is called an *extension field* (of $GF(2)$).
- Each element of $GF(2^8)$ can be represented by one byte.
- In $GF(2^8)$ elements of the fields are not numbers, but are polynomials, whose coefficients are in GF(2) (i.e., 0 or 1):

$$A(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

where each $a_i \in GF(2)$.

- Every such polynomial can be stored in a byte (8-bit vector):

$$A = (a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0)$$

Addition and subtraction in $GF(2^m)$

Let $A(x), B(x) \in GF(2^m)$. The sum of the two elements is then computed according to:

$$C(x) = A(x) + B(x) = \sum_{i=0}^{m-1} c_i x^i, \quad c_i \equiv a_i + b_i \bmod 2$$

and the difference is computed according to:

$$C(x) = A(x) - B(x) = \sum_{i=0}^{m-1} c_i x^i, \quad c_i \equiv a_i - b_i \equiv a_i + b_i \bmod 2.$$

Note: addition and subtraction for the coefficients are the same operation (XOR).

Example:

$$\begin{array}{rl} A(x) = & x^7 + x^6 + x^4 + \quad 1 \\ B(x) = & \quad\quad x^4 + x^2 + 1 \\ \hline C(x) = & x^7 + x^6 + \quad x^2 \end{array}$$

Note that the coefficients are in GF(2), so for example, $x^4 + x^4 = 2x^4 = 0 \cdot x^4 = 0$ since $2 \equiv 0 \; mod \; 2$.

Multiplication in $GF(2^m)$

- To do multiplication in $GF(2^m)$, we first do the usual polynomial multiplication, but the coefficient is computed modulo 2.
- Example: consider two elements of $GF(2^3)$, $A(x) = x^2 + x + 1$ and $B(x) = x + 1$. Doing the standard polynomial multiplication we get

$$C(x) = A(x) \cdot B(x) = x^3 + 1$$

- Note however, $C(x)$ has degree 3, so not in $GF(2^3)$.
- We need to reduce the polynomial by computing its remainder modulo a "prime" polynomial, which is *irreducible*.
- A polynomial is irreducible if it cannot be factored into two non-constant polynomials:
- E.g., $x^4 + x^3 + x + 1$ is reducible since

$$x^4 + x^3 + x + 1 = (x^2 + x + 1)(x^2 + 1)$$

- But $x^4 + x + 1$ is irreducible.

Multiplication in $GF(2^m)$

Let $A(x), B(x) \in GF(2^m)$ *and let*

$$P(x) \equiv \sum_{i=0}^{m} p_i x^i, \quad p_i \in GF(2)$$

be an irreducible polynomial. Multiplication of the two elements $A(x), B(x)$ *is performed as*

$$C(x) \equiv A(x) \cdot B(x) \bmod P(x).$$

For AES, $m = 8$ and $P(x) = x^8 + x^4 + x^3 + x + 1$.

The concept of a remainder when dividing against an irreducible polynomial is similar to the concept of remainder in modular arithmetic:

$$F(x) \equiv G(x) \, mod \, P(x)$$

holds if and only if there is a $H(x)$ such that $F(x) = H(x) \cdot P(x) + G(x)$

Multiplication in $GF(2^m)$

- Example: multiply $A(x) = x^3 + x^2 + 1$ and $B(x) = x^2 + x$ in $GF(2^4)$, using the irreducible polynomial $P(x) = x^4 + x + 1$.

(See Example 4.6 in Chapter 4 of ”Understanding Cryptography” by Paar & Pelzl for a detailed calculation.)

Inversion in $GF(2^m)$

- For a given finite field $GF(2^m)$ and the corresponding irreducible reduction polynomial $P(x)$, the inverse A^{-1} of a non-zero element $A \in GF(2^m)$ is defined as:

$$A^{-1}(x) \cdot A(x) = 1 \, mod \, P(x)$$

- For small fields, such as $GF(2^8)$, multiplicative inverses are often implemented using a lookup table:

Inversion in $GF(2^m)$

Multiplicative inverse table in $GF(2^8)$ for bytes xy used within the AES S-Box

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

For a byte XY (in HEX notation), its inverse is at row X and column Y.

Inversion in $GF(2^m)$

- Example: use the inverse table for $GF(2^8)$ to find the inverse of

$$\mathrm{A(x)} = x^7 + x^6 + x.$$

- First, find the bit vector that represents the polynomial:

$$x^7 + x^6 + x = 1 \cdot x^7 + 1 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 0$$

- So the polynomial A(x) is represented as the bit vector (1100 0010).
- In HEX, this is (C2)
- The left 4-bit corresponds to the X (row) of the table, and the right 4-bit corresponds to the Y (column) of the table.
- So the inverse of A(x) is given by the entry in row C and column 2 in the table, which is 2F

$$(2F)_{hex} = (00101111)_2 = x^5 + x^3 + x^2 + x + 1.$$

Content of this Chapter

- Overview of block ciphers
- Overview of the AES algorithm
- Galois Field
- **Internal structure of AES**
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- Decryption
- Practical issues

Internal Structure of AES

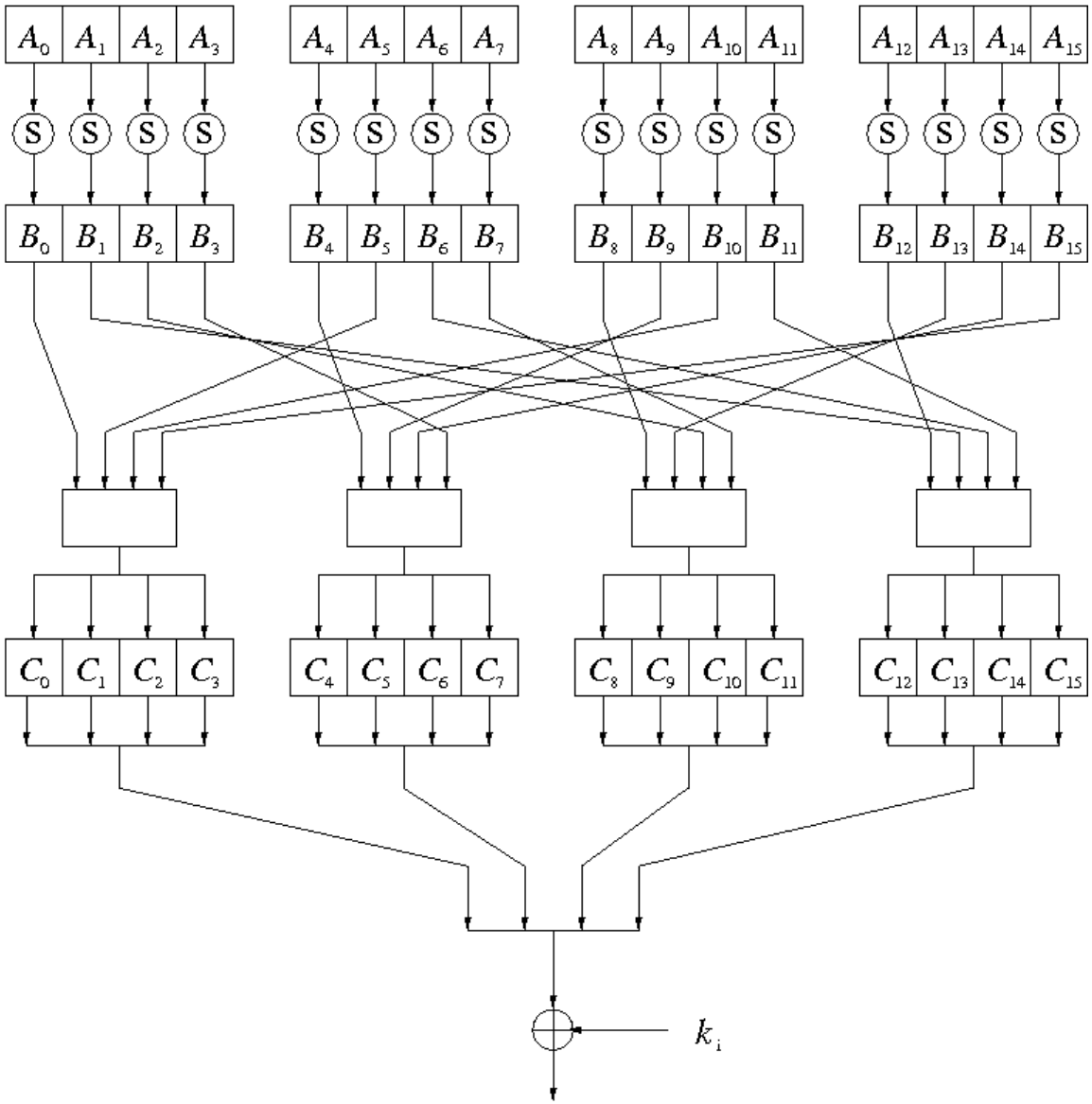
- AES is a byte-oriented cipher
- The state *A* (i.e., the 128-bit data path) can be arranged in a 4x4 matrix:

A_0	A_4	A_8	A_{12}
A_1	A_5	A_9	A_{13}
A_2	A_6	A_{10}	A_{14}
A_3	A_7	A_{11}	A_{15}

  with $A_0, \ldots, A_{15}$ denoting the 16-byte input of AES
- Each byte A_i represents a polynomial in $GF(2^8)$.

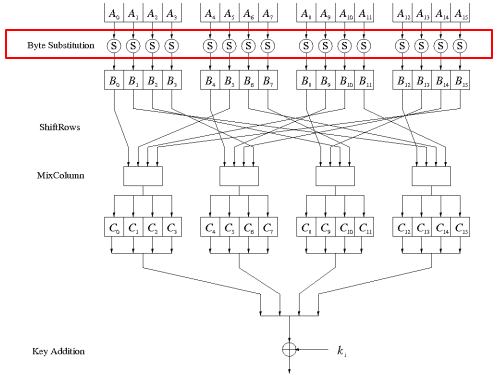
Internal Structure of AES

- Round function for rounds 1,2,…,n_{r-1}:



- Note: In the last round, the MixColumn transformation is omitted.

Byte Substitution Layer (S-Box)



- The Byte Substitution layer consists of 16 **S-Boxes** with the following properties:

  The S-Boxes are

  - **identical**
  - the only **nonlinear** elements of AES, i.e., ByteSub(A_i) + ByteSub(A_j) ≠ ByteSub(A_i + A_j), for i,j = 0,…,15
  - **bijective**, i.e., there exists a one-to-one mapping of input and output bytes

    ⇒ S-Box can be uniquely reversed

- In software implementations, the S-Box is usually realized as a lookup table

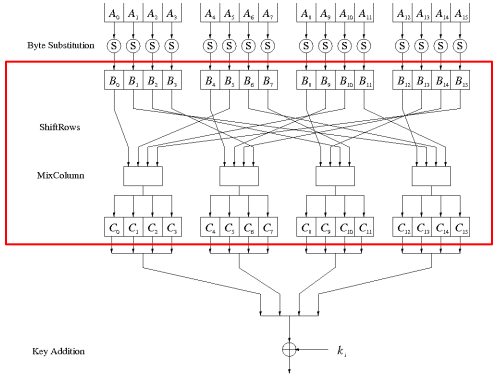
Byte Substitution Layer (S-Box)

AES S-Box: Substitution values in hexadecimal notation for input byte (xy)

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

The lookup table encodes a function that composes an inversion operation and an affine mapping in $GF(2^8)$.

Diffusion Layer



The Diffusion layer

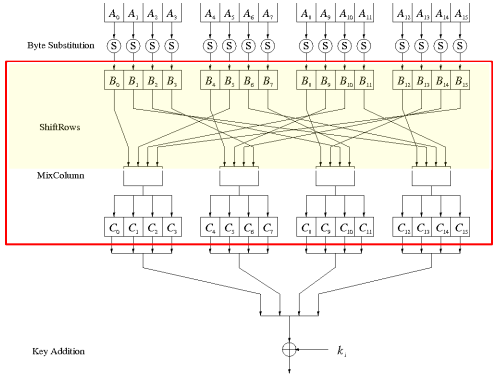
- provides diffusion over all input state bits

- consists of two sublayers:
  - **ShiftRows Sublayer**: Permutation of the data on a byte level
  - **MixColumn Sublayer**: Matrix operation which combines ("mixes") blocks of four bytes

- performs a linear operation on state matrices *A*, *B*, i.e.,

$$\text{DIFF}(A) + \text{DIFF}(B) = \text{DIFF}(A + B)$$

ShiftRows Sublayer



- Rows of the state matrix are shifted cyclically:

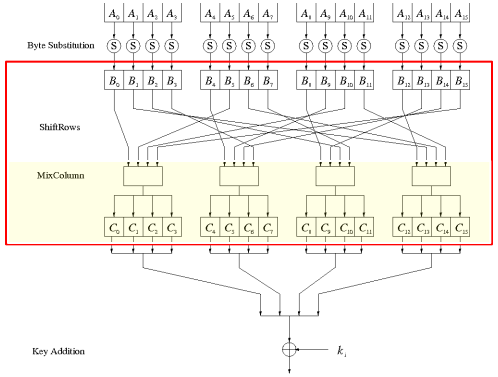
Input matrix

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_5	B_9	B_{13}	B_1	← one position left shift
B_{10}	B_{14}	B_2	B_6	← two positions left shift
B_{15}	B_3	B_7	B_{11}	← three positions left shift

MixColumn Sublayer



- Linear transformation which mixes each column of the state matrix
- Each 4-byte column is considered as a vector and multiplied by a fixed 4x4 matrix, e.g.,

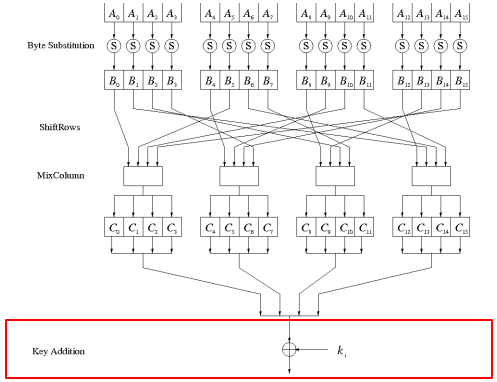
$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

where 01, 02 and 03 are given in hexadecimal notation. For example, C_0 is given by the equation:

$$C_0 = 02 \cdot B_0 + 03 \cdot B_5 + 01 \cdot B_{10} + 01 \cdot B_{15} \; mod \; P(x)$$

- All arithmetic is done in the Galois field $GF(2^8)$, i.e., **each byte is treated as a polynomial in $GF(2^8)$.**

Key Addition Layer



- Inputs:
  - 16-byte state matrix *C*
  - 16-byte subkey k_i
- Output: $C \oplus k_i$.
- The subkeys are generated in the ***key schedule.***

Key Schedule

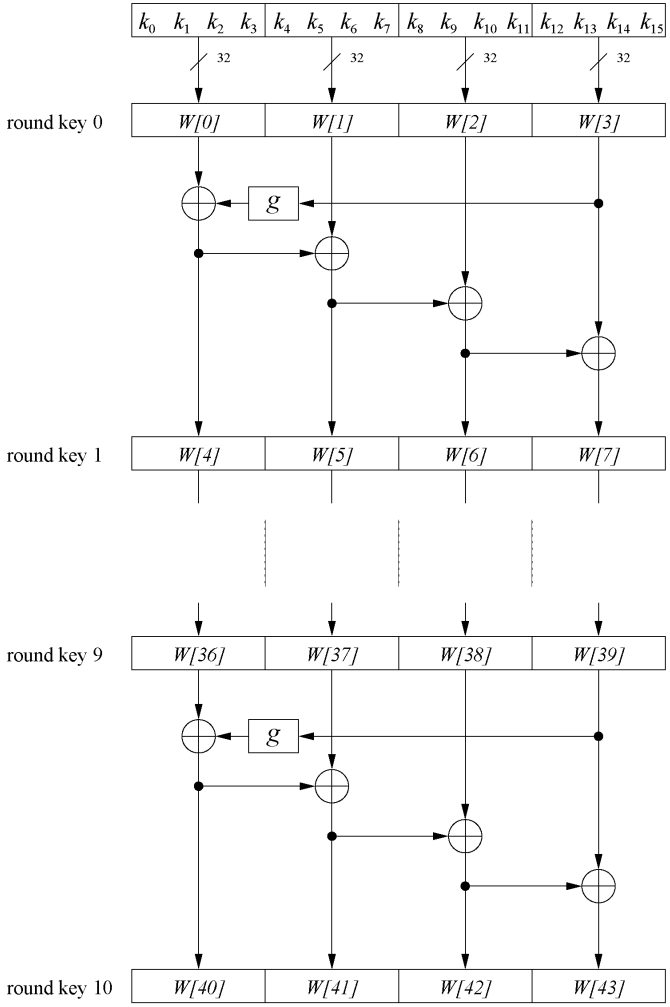
- Subkeys are derived recursively from the original 128/192/256-bit input key
- Each round has 1 subkey, plus 1 subkey at the beginning of AES

Key length (bits)	Number of subkeys
128	11
192	13
256	15

- Subkey is used both at the input and output of AES
  $\Rightarrow$ # subkeys = # rounds + 1
- There are different key schedules for the different key sizes

Key Schedule

Example: Key schedule for 128-bit key AES



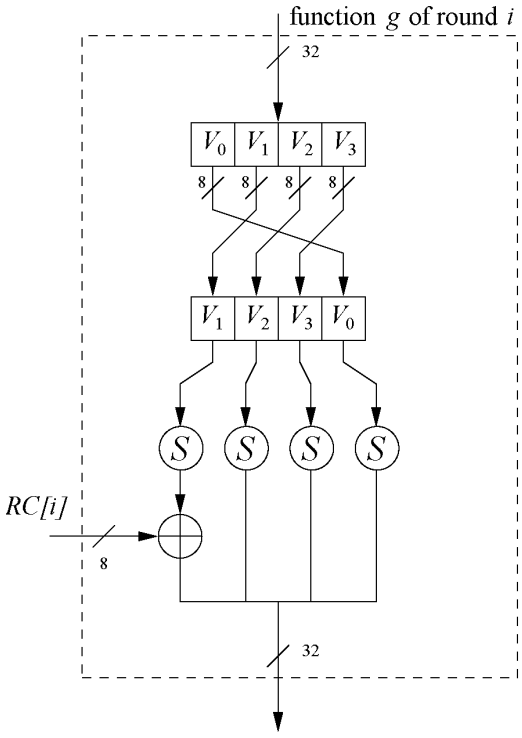
- Word-oriented: 1 word = 32 bits
- 11 subkeys are stored in *W[0]…W[3], W[4]…W[7], … , W[40]…W[43]*
- First subkey *W[0]…W[3]* is the original AES key

Key Schedule

- Function *g* rotates its four input bytes and performs a bytewise S-Box substitution
  $\Rightarrow$ nonlinearity

- The round coefficient *RC* is only added to the leftmost byte and varies from round to round:

$$RC[1] = x^0 = (00000001)_2$$
$$RC[2] = x^1 = (00000010)_2$$
$$RC[3] = x^2 = (00000100)_2$$
$$\ldots$$
$$RC[10] = x^9 = (00110110)_2$$

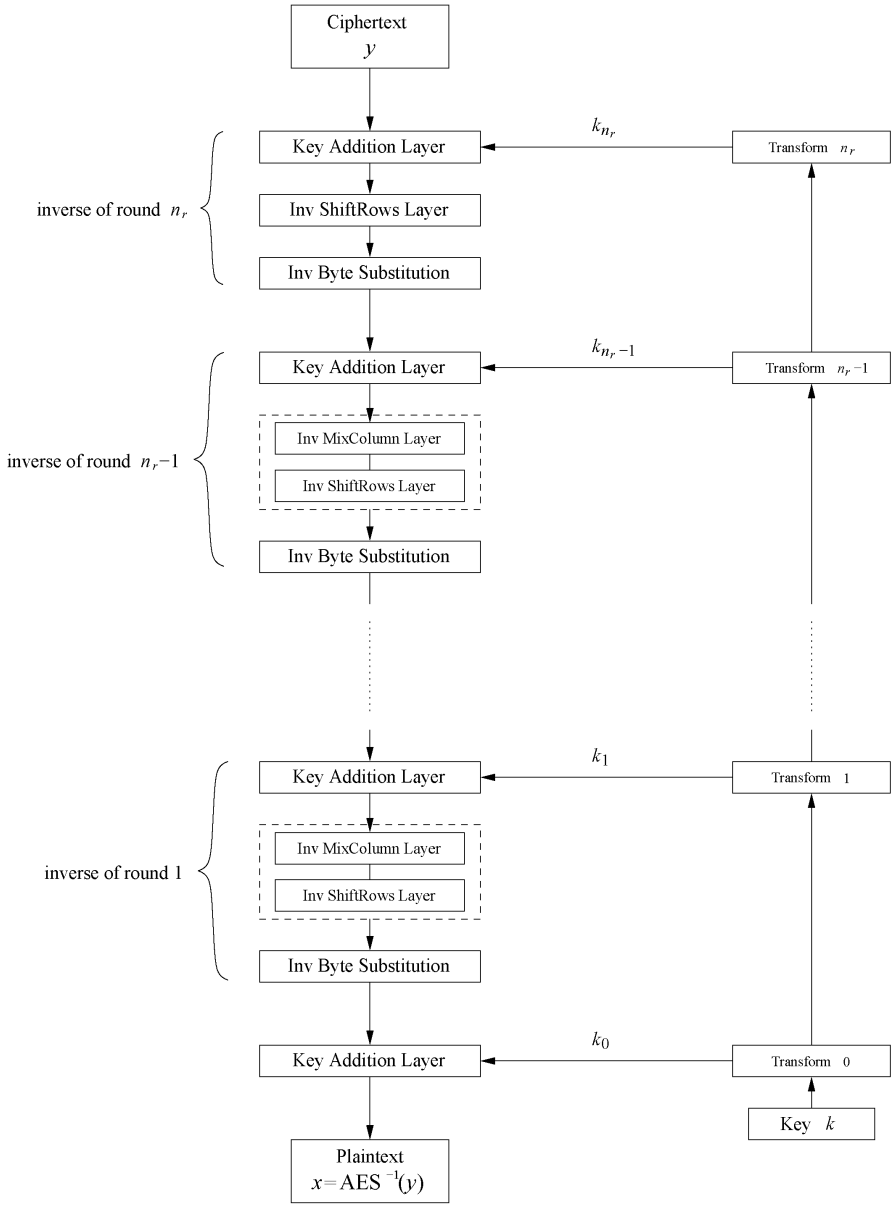


- x^i represents an element in a Galois field $GF(2^8)$.

Content of this Chapter

- Overview of the AES algorithm
- Internal structure of AES
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- **Decryption**
- Practical issues

Decryption



All layers must be inverted for decryption:

- MixColumn layer → **Inv MixColumn layer**
- ShiftRows layer→ **Inv ShiftRows layer**
- Byte Substitution layer → **Inv Byte Substitution layer**
- Key Addition layer is its own inverse

Decryption

- **Inv MixColumn layer**:
  - To reverse the MixColumn operation, each column of the state matrix C must be multiplied with the **inverse of the 4x4 matrix**, e.g.,

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \cdot \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

where 09, 0*B, 0D* and 0*E* are given in hexadecimal notation

- Again, all arithmetic is done in the Galois field $GF(2^8)$.

Decryption

- **Inv ShiftRows layer**:
  - All rows of the state matrix *B* are shifted to the opposite direction:

Input matrix

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Output matrix

B_0	B_4	B_8	B_{12}	no shift
B_{13}	B_1	B_5	B_9	→ one position right shift
B_{10}	B_{14}	B_2	B_6	→ two positions right shift
B_7	B_{11}	B_{15}	B_3	→ three positions right shift

Decryption

- **Inv Byte Substitution layer**:
  - Since the S-Box is bijective, it is possible to construct an inverse, such that

$$A_i = S^{-1}(B_i) = S^{-1}(S(A_i))$$

    $\Rightarrow$ The inverse S-Box is used for decryption. It is usually realized as a lookup table

- **Decryption key schedule**:
  - Subkeys are needed in reversed order (compared to encryption)
  - In practice, for encryption and decryption, the same key schedule is used. This requires that all subkeys must be computed before the encryption of the first block can begin

Content of this Chapter

- Overview of the AES algorithm
- Internal structure of AES
  - Byte Substitution layer
  - Diffusion layer
  - Key Addition layer
  - Key schedule
- Decryption
- **Practical issues**

Implementation in Software

- One requirement of AES was the possibility of an efficient software implementation
- Straightforward implementation is well suited for 8-bit processors (e.g., smart cards), but inefficient on 32-bit or 64-bit processors
- A more sophisticated approach: Merge all round functions (except the key addition) into one table look-up
  - This results in four tables with 256 entries, where each entry is 32 bits wide
  - One round can be computed with 16 table look-ups
- Typical SW speeds are more than 1.6 Gbit/s on modern 64-bit processors

Security

- **Brute-force attack:** Due to the key length of 128, 192 or 256 bits, a brute-force attack is not possible
- **Analytical attacks:** There is no analytical attack known that is better than brute-force
- **Side-channel attacks:**
  - Several side-channel attacks have been published
  - Note that side-channel attacks do not attack the underlying algorithm but the implementation of it

Lessons Learned

- AES is a modern block cipher which supports three key lengths of 128, 192 and 256 bit. It provides excellent long-term security against brute-force attacks.
- AES has been studied intensively since the late 1990s and no attacks have been found that are better than brute-force.
- Its basic operations use Galois field arithmetic and provide strong diffusion and confusion.
- AES is part of numerous open standards such as IPsec or TLS, in addition to being the mandatory encryption algorithm for US government applications. It seems likely that the cipher will be the dominant encryption algorithm for many years to come.
- AES is efficient in software and hardware.

Encryption Mode

Content of this Chapter

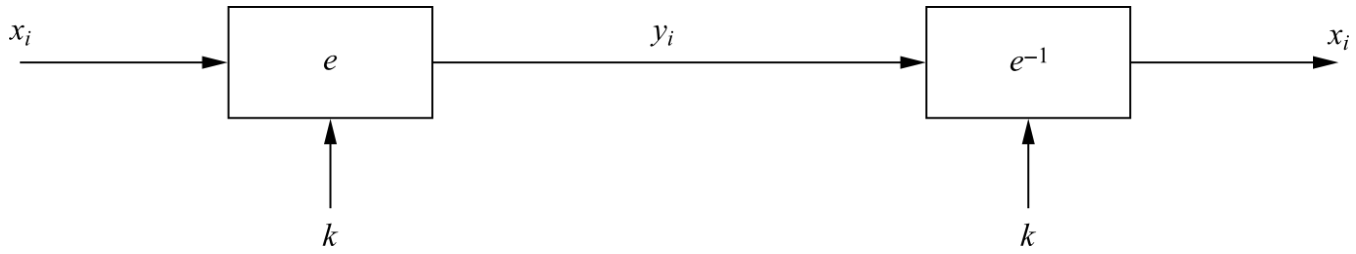
- Encryption with Block Ciphers: Modes of Operation
  - Electronic Code Book mode (ECB)
  - Cipher Block Chaining mode (CBC)
  - Output Feedback mode (OFB)
  - Cipher Feedback mode (CFB)
  - Counter mode (CTR)

Encryption with Block Ciphers

- There are several ways of encrypting long plaintexts, e.g., an e-mail or a computer file, with a block cipher ("modes of operation")
  - Electronic Code Book mode (ECB)
  - Cipher Block Chaining mode (CBC)
  - Output Feedback mode (OFB)
  - Cipher Feedback mode (CFB)
  - Counter mode (CTR)
  - Galois Counter Mode (GCM)
- All of the 6 modes have one desired goal:
  - In addition to confidentiality, they should provide authenticity and integrity:
    - Is the message really coming from the original sender? (authenticity)
    - Was the ciphertext altered during transmission? (integrity)
  - However, not all modes provide integrity or authenticity.

Electronic Code Book mode (ECB)

- $e_k(x_i)$ denote the encryption of a b-bit plaintext block x_i with key k
- $e_k^{-1}(y_i)$ denote the decryption of b-bit ciphertext block y_i with key k
- Messages which exceed b bits are partitioned into b-bit blocks
- **Each Block is encrypted separately**



Encryption: $y_i = e_k(x_i), \quad i \geq 1$

Decryption: $x_i = e_k^{-1}(y_i) = e_k^{-1}(e_k(x_i)), \quad i \geq 1$

ECB: advantages/disadvantages

- Advantages
  - no block synchronization between sender and receiver is required
  - bit errors caused by noisy channels only affect the corresponding block but not succeeding blocks
  - Block cipher operating can be parallelized
    - advantage for high-speed implementations

- Disadvantages
  - ECB encrypts highly deterministically
    - identical plaintexts result in identical ciphertexts
    - an attacker recognizes if the same message has been sent twice
    - plaintext blocks are encrypted independently of previous blocks
      - an attacker may reorder ciphertext blocks which results in valid plaintext

Substitution Attack on ECB

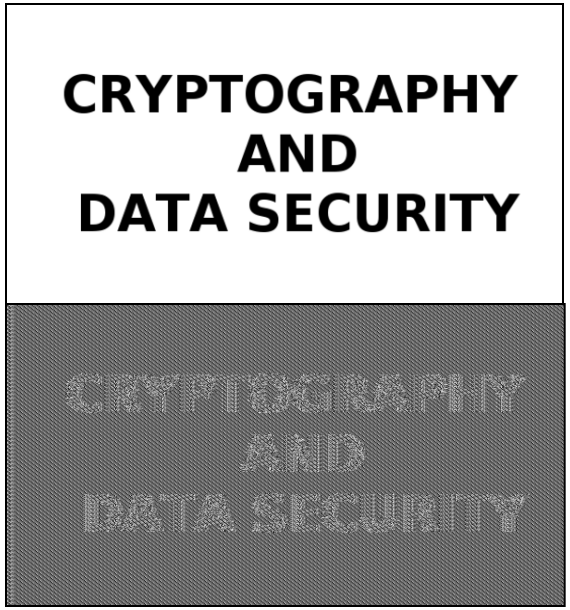
- Once a particular plaintext to ciphertext block mapping $x_i \rightarrow y_i$ is known, a sequence of ciphertext blocks can easily be manipulated
- Suppose an *electronic bank transfer*

Block #	1	2	3	4	5
	Sending Bank A	Sending Account #	Receiving Bank B	Receiving Account #	Amount $

  - the encryption key between the two banks does not change too frequently
  - The attacker sends $1.00 transfers from his account at bank A to his account at bank B repeatedly
    - He can check for ciphertext blocks that repeat, and he stores blocks 1,3 and 4 of these transfers
  - He now simply replaces block 4 of other transfers with the block 4 that he stored before
    - *all transfers* from some account of bank A to some account of bank B are redirected to go into the attacker's B account!

Example of encrypting bitmaps in ECB mode

- Identical plaintexts are mapped to identical ciphertexts



- Statistical properties in the plaintext are preserved in the ciphertext

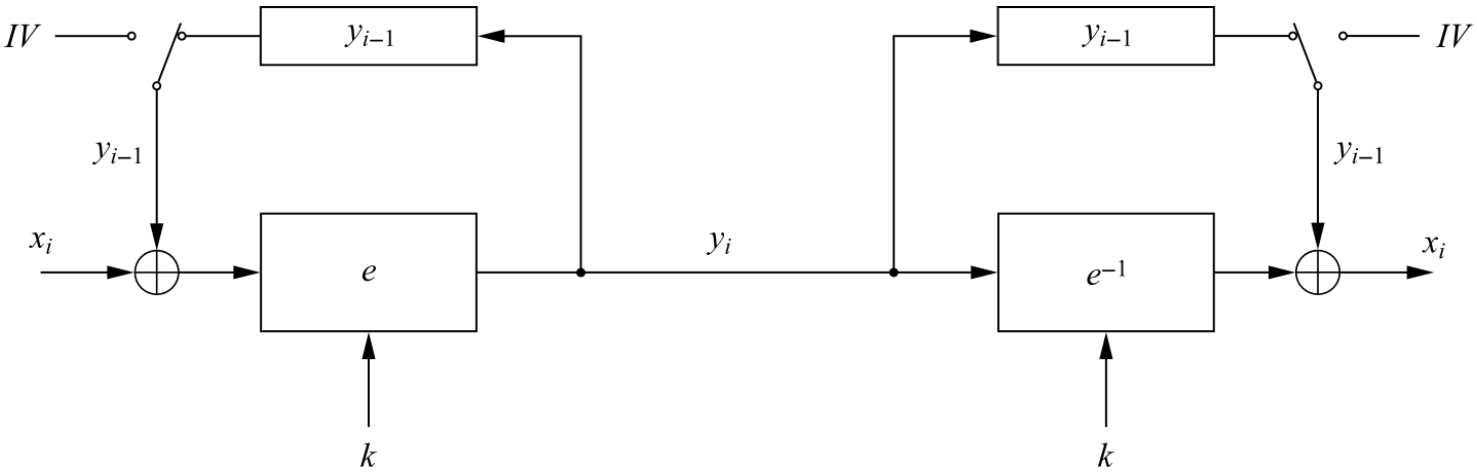
Cipher Block Chaining mode (CBC)

- There are two main ideas behind the CBC mode:
  - The encryption of all blocks are “chained together”
    - ciphertext y_i depends not only on block x_i but on all previous plaintext blocks as well
  - The encryption is randomized by using an initialization vector (IV)

***Encryption* (*first block*)**:	$y_1 = e_k(x_1 \oplus \text{IV})$
***Encryption* (*general block*)**:	$y_i = e_k(x_i \oplus y_{i-1}), \ i \geq 2$
***Decryption* (*first block*)**:	$x_1 = e_k^{-1}(y_1) \oplus \text{IV}$
***Decryption* (*general block*)** :	$x_i = e_k^{-1}(y_i) \oplus y_{i-1}, \ i \geq 2$

Cipher Block Chaining mode (CBC)

- For the first plaintext block x_1 there is no previous ciphertext
  - an IV is added to the first plaintext to make each CBC encryption nondeterministic
  - the first ciphertext y_1 depends on plaintext x_1 and the IV
- The second ciphertext y_2 depends on the IV, x_1 *and* x_2
- The third ciphertext y_3 depends on the IV and x_1, x_2 *and* x_3, and so on

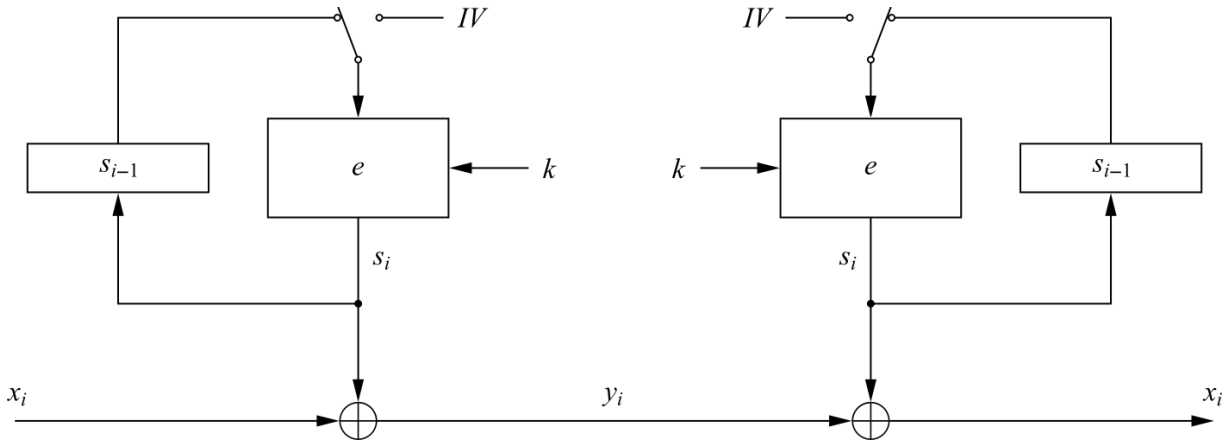


Substitution Attack on CBC

- Suppose the last example (*electronic bank transfer*)
- If the IV is properly chosen for every wire transfer, the attack will not work at all
- If the IV is kept the same for several transfers, the attacker would recognize the transfers from his account at bank A to back B
- If we choose a new IV every time we encrypt, the CBC mode becomes a probabilistic encryption scheme, i.e., two encryptions of the same plaintext look entirely different
- It is not needed to keep the IV *secret*!
- Typically, the IV should be a non-secret nonce (value used only once)

Output Feedback mode (OFB)

- It is used to build a *synchronous* **stream cipher** from a block cipher
- The key stream is not generated bitwise but instead in a blockwise fashion
- The output of the cipher gives us key stream bits S_i with which we can encrypt plaintext bits using the XOR operation



***Encryption* (*first block*)**: $s_1 = e_k(\mathrm{IV})$ *and* $y_1 = s_1 \oplus x_1$

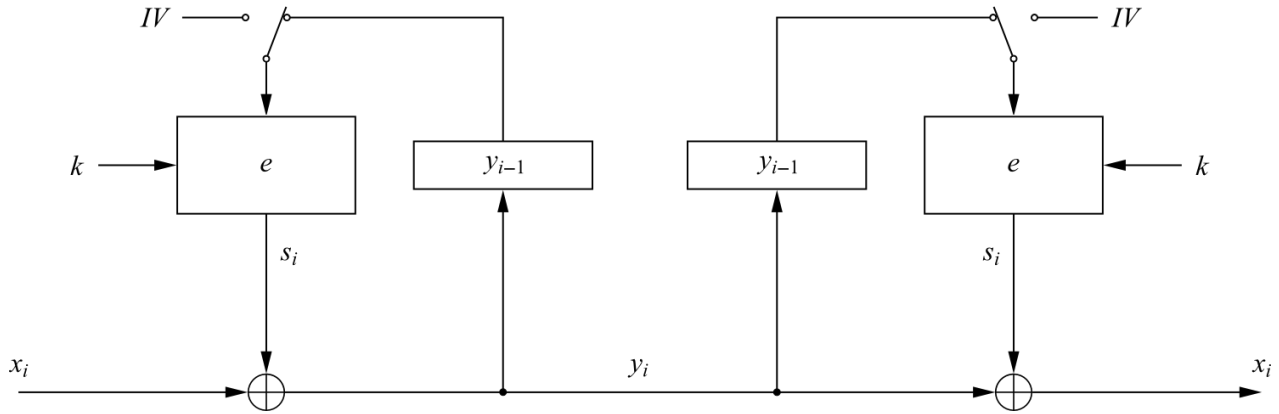
***Encryption* (*general block*)**: $s_i = e_k(s_{i-1})$ *and* $y_i = s_i \oplus x_i$, $i \geq 2$

***Decryption* (*first block*)**: $s_1 = e_k(\mathrm{IV})$ *and* $x_1 = s_1 \oplus y_1$

***Decryption* (*general block*)** : $s_i = e_k(s_{i-1})$ *and* $x_i = s_i \oplus y_i$, $i \geq 2$

Cipher Feedback mode (CFB)

- It uses a block cipher as a building block for an asynchronous **stream cipher** (similar to the OFB mode), more accurate name: “Ciphertext Feedback Mode”
- The key stream S_i is generated in a blockwise fashion and is also a function of the ciphertext
- As a result of the use of an IV, the CFB encryption is also nondeterministic

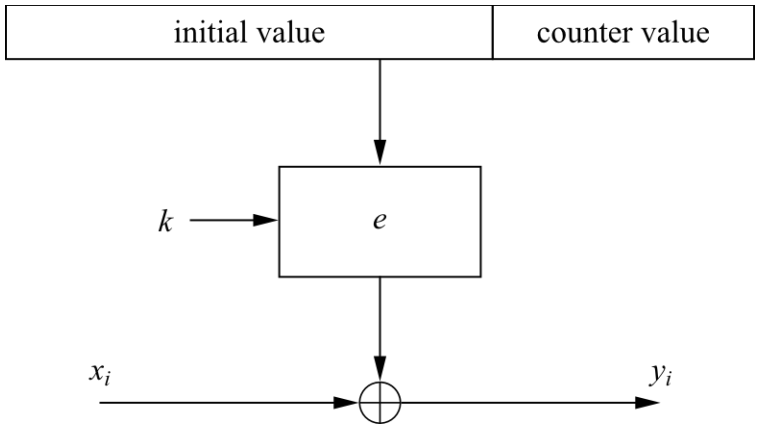


***Encryption* (*first block*)**: $y_1 = e_k(\text{IV}) \oplus x_1$
***Encryption* (*general block*)**: $y_i = e_k(y_{i-1}) \oplus x_i, \quad i \geq 2$
***Decryption* (*first block*)**: $x_1 = e_k(\text{IV}) \oplus y_1$
***Decryption* (*general block*)** : $x_i = e_k(y_{i-1}) \oplus y_i, \quad i \geq 2$

- It can be used in situations where short plaintext blocks are to be encrypted

Counter mode (CTR)

- It uses a block cipher as a **stream cipher** (like the OFB and CFB modes)
- The key stream is computed in a block wise fashion
- The input to the block cipher is a counter which assumes a different value every time the block cipher computes a new key stream block



- Unlike CFB and OFB modes, the CTR mode can be parallelized since the 2nd encryption can begin before the 1st one has finished
  - Desirable for high-speed implementations, e.g., in network routers

$$\textbf{\textit{Encryption}}: \quad y_i = e_k(\text{IV} \,||\, \text{CTR}_i) \oplus x_i, \quad i \geq 1$$
$$\textbf{\textit{Decryption}}: \quad x_i = e_k(\text{IV} \,||\, \text{CTR}_i) \oplus y_i, \quad i \geq 1$$

Lessons Learned

- There are many ways to encrypt with a block cipher. Each mode of operation has some advantages and disadvantages
- Several modes turn a block cipher into a stream cipher
- There are modes that perform encryption together together with authentication, i.e., a cryptographic checksum protects against message manipulation
- The straightforward ECB mode has security weaknesses, independent of the underlying block cipher
- The counter mode allows parallelization of encryption and is thus suited for high speed implementations