

DES, Triple-DES and AES

# Software Security

---

**Steffen Helke**

Chair of Software Engineering

17th December 2018



Brandenburgische  
Technische Universität  
Cottbus - Senftenberg

# Objectives of today's exercise

---

- Getting to know the meaning of *confusion* and *diffusion*, and how these concepts are implemented
- Understanding the advantages of a *Feistel network* and the basic principles of *DES* and *Triple-DES*
- Learning how to perform *attacks* against symmetric encryption systems, like DES and Triple-DES
- Understanding how *AES* works

# DES/AES: Symmetric-key Encryption

System type		Concealment		Authentikation	
Security level		sym.	asym.	sym.	asym.
information theoretical		sym. concealation system	asym. concealation system	sym. authentication system	digital signature system
crypto-graphically strong against...	active attack	Vernam-Chiffre (one-time pad)	X	Authentication codes	X
	passive attack	Pseudo-one-time-pad with $s^2$ -mod-n-Generator			GMR
			System with $s^2$ -mod-n-Generator		
well researched	mathematical chaos		RSA		RSA
		DES/AES		DES/AES	

Source: Andreas Pfitzmann: Security in IT-Networks, 2012

## **DES - Data Encryption Standard**

– Symmetric-key Concelation and Authentication –

# History of DES

---

- At the beginning of the 1970s, an encryption standard was required for general use
- A public call for proposals was initiated by the NBS (*National Bureau of Standard*, USA), today NIST
- Important requirement: The strength of the encryption algorithm should be **based only on the secrecy of the key** and **not on the secrecy of the algorithm** (*Kerckhoffs's principle*)

## Calls

- 1972** NBS publishes a first request for a standard encryption algorithm without success
- 1974** Second request was successful, IBM submitted a new version of the *Lucifer* algorithm
- 1977** Ongoing development with support from the NSA, DES was defined as the encryption standard

# What are the criteria for evaluating chaos-based encryption systems?

---

What is the meaning of confusion and diffusion and how these concepts are usually implemented?

## Diffusion

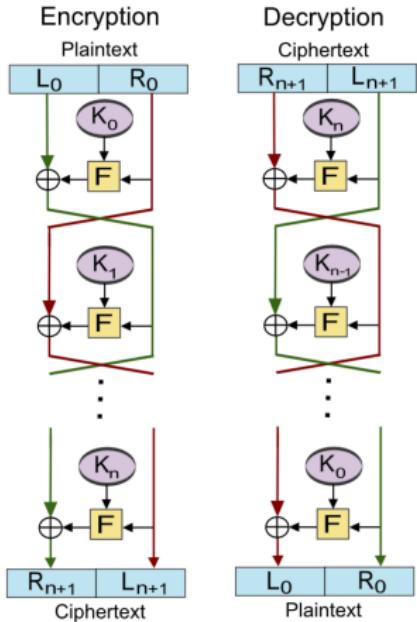
- Effectiveness of changing an **input bit** to as **many output bits as possible**, also known as an *avalanche effect*
- **Implementation** by *permutation* (e.g. using P-boxes), often also expansion permutation used

## Confusion

- Hiding the link between plain and ciphertext
- Implementation by *Substitution* (e.g. using S-boxes)

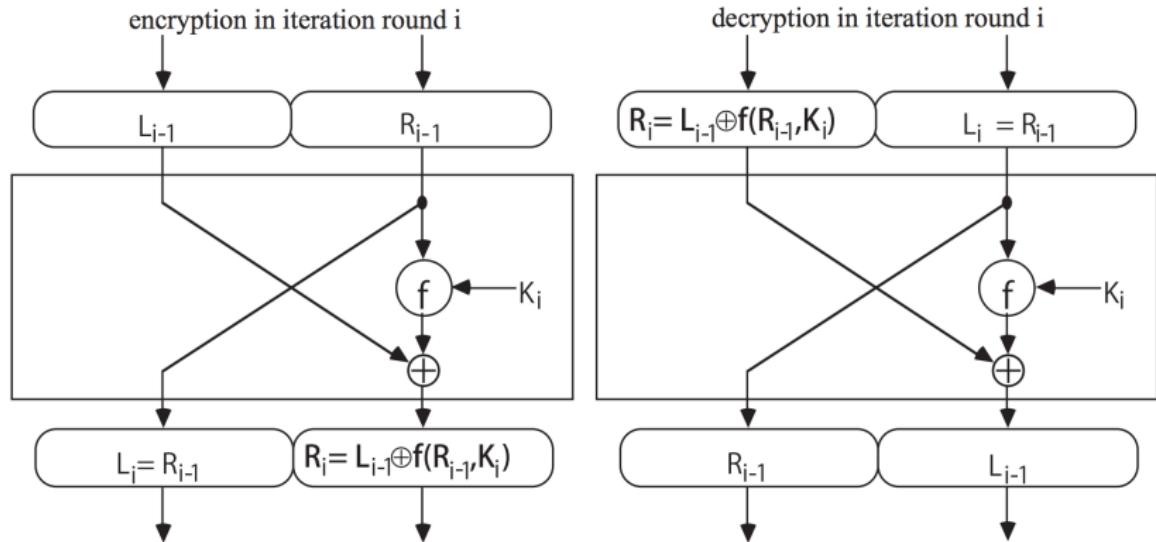
# Basics of Feistel networks

What are the advantages of a Feistel network and which encryption system uses this technology?



- **Idea:** For encryption and decryption the **same network can be used**, only the order of the **round keys** has to be **inverted**
- Function  $F$  has **not to be inverted** for decryption
- DES based on a Feistel network (**16 rounds**)

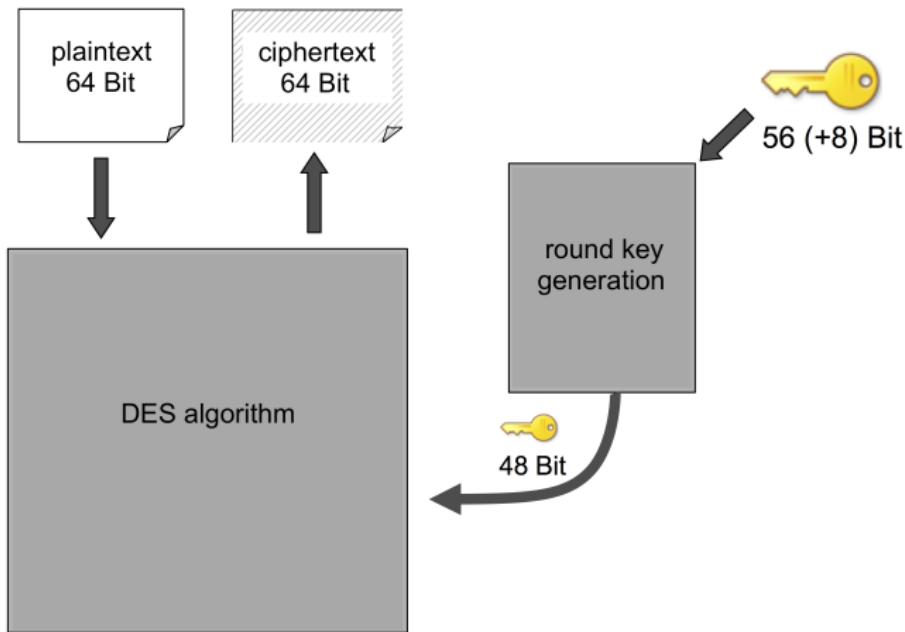
# Decryption Principle of DES



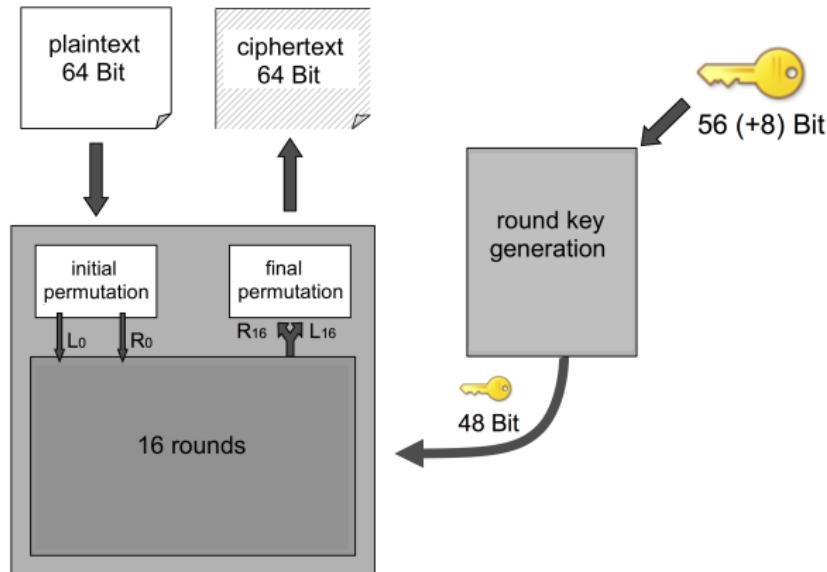
Source: Andreas Pfitzmann: Security in IT-Networks, 2012

# An Abstract View on the DES Algorithm (1)

---



# An Abstract View on the DES Algorithm (2)



→ **Note:** Initial and final permutation have no cryptographic significance!

# Initial and Final Permutation

---

- The input block

$$X = (x_1, x_2, \dots, x_{64})$$

is transformed to a permuted input block

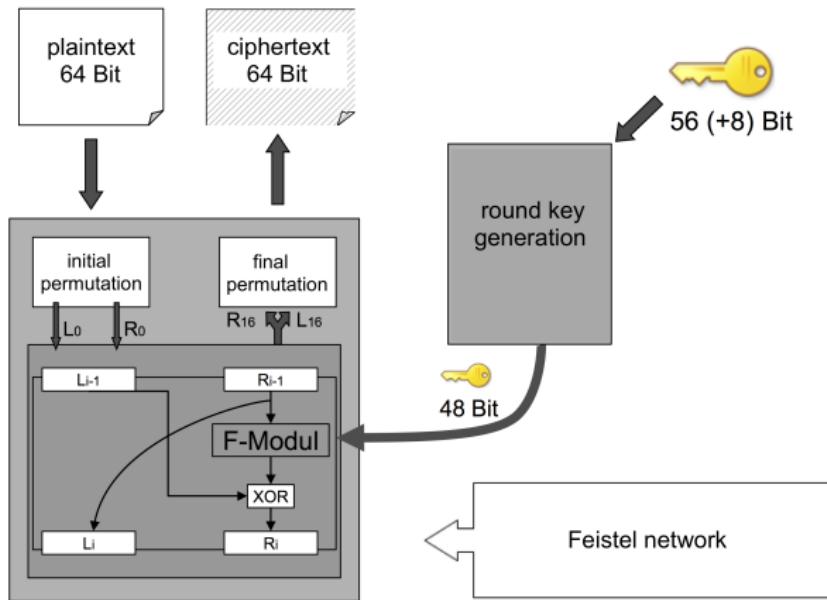
$$IP(X) = (x_{58}, x_{50}, \dots, x_7)$$

according to the following rule

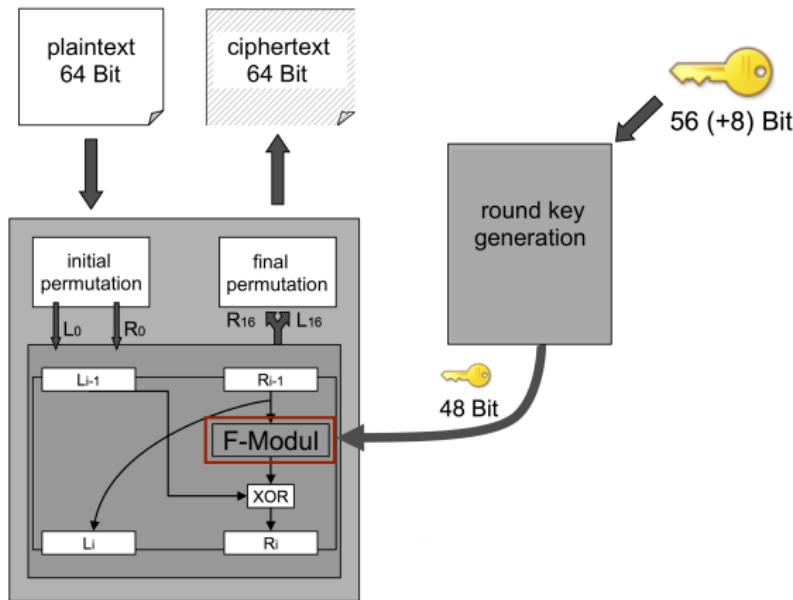
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- The final permutation is defined by the inverted initial permutation  $IP^{-1}$ !

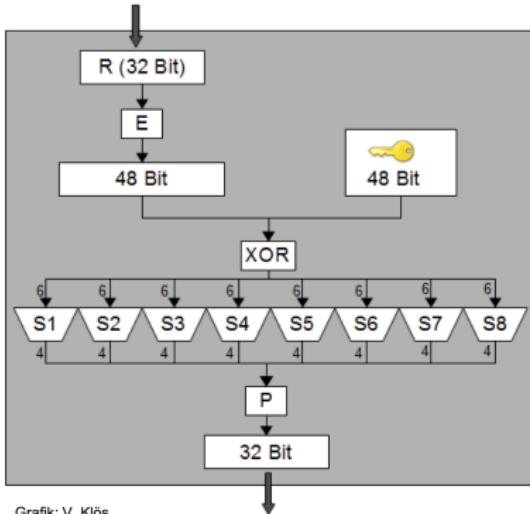
# Feistel Network within the DES Algorithm



# F-Module within the Feistel Network



# Components of the F-Module



Grafik: V. Klös

- $E$ : Expansion permutation
- $S_i$ : Substitution boxes (S-boxes)
- $P$ : Permutation for the next round

# Expansion Permutation

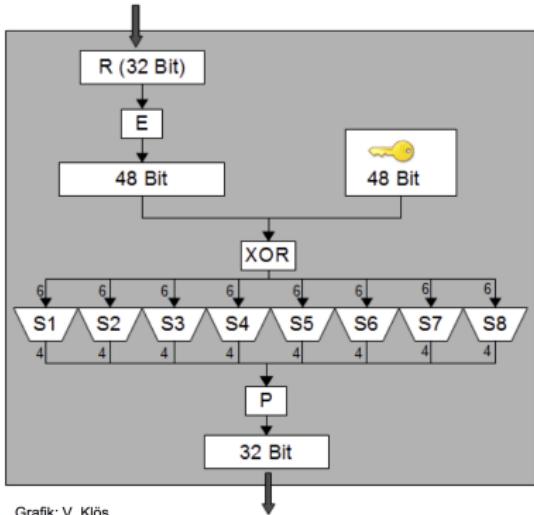
---

- How to expand a 32-bit input block to an 48-bit block?

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- Note: All fields highlighted in gray are added as duplicates of the input block by expansion

# Components of the F-Module



Grafik: V. Klös

- $E$ : Expansion permutation
- $S_i$ : Substitution boxes (S-boxes)
- $P$ : Permutation for the next round

## Substitution using S-Box $S_1$

---

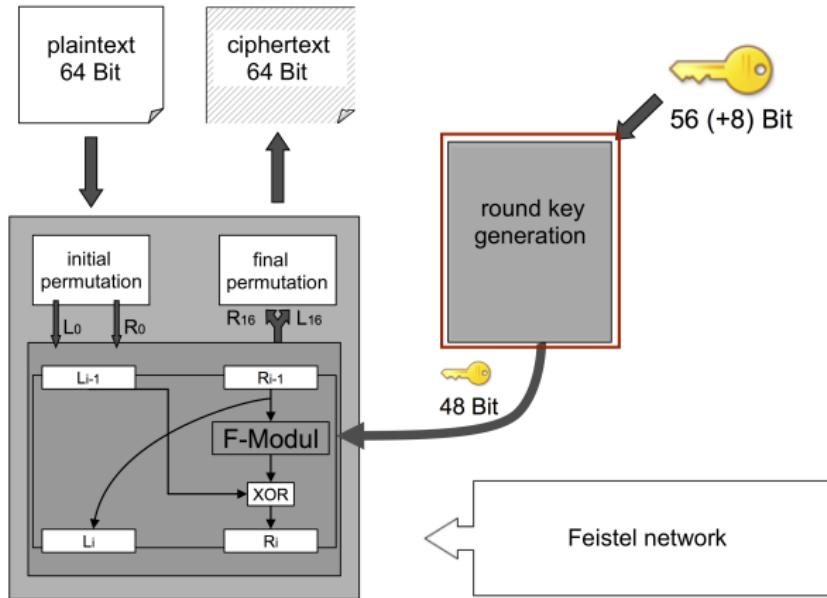
- Note, substitution is not bitwise, but according to the following procedure

- 1 Split the input: 100110
- 2 Calculate the row number:  $10 \hat{=} 2$
- 3 Calculate the column number:  $0011 \hat{=} 3$
- 4 Find the result using the s-box:  $8 \hat{=} 1000$

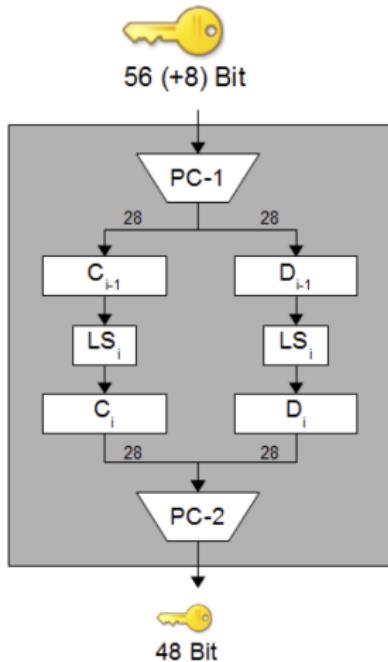
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0

- The substitution results in a compression of the bit block (reduction from 6 bits to 4 bits)

# How to generate round keys?

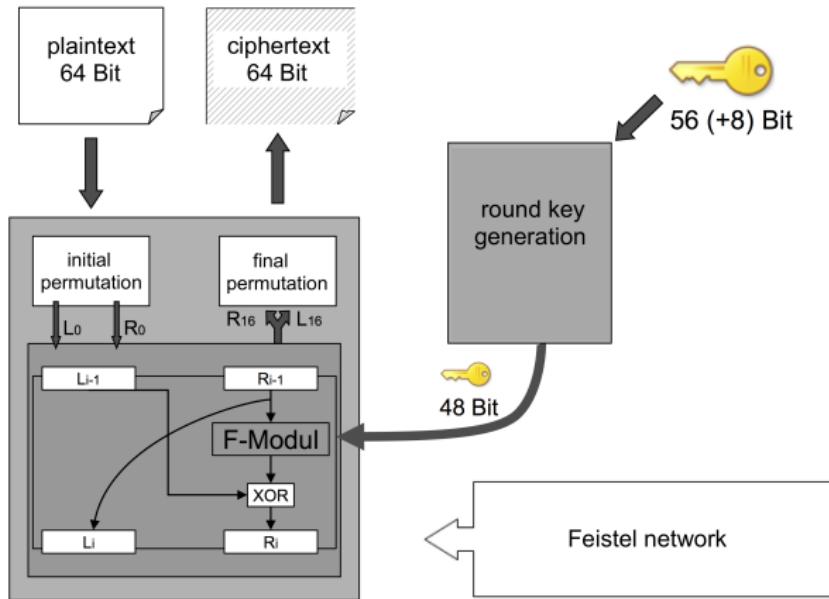


# How to generate round keys?



- $PC - 1$ : Permutated choice  
(56 bits from 64)
- $PC - 2$ : Permutated choice  
(48 bits from 56)
- $LS_i$ : Cyclical left-shift  
operation (by 1 or 2 bits)

# Decryption using DES



- For decrypting, the same algorithm is used, but with the round keys in reverse order!

## **DES - Attacks**

– Brute-Force Attack –

# Brute-Force Attack for DES

How works a brute-force attack? What is an important assumption of this attack?

- Brute-force attack for DES *can be easily implemented* today, since the size of the key space can be tested with  $2^{56}$  in a practicable time

## Procedure for Calculating the Key

- 1 Select a ciphertext  $C$  with the corresponding plaintext  $P$ 
  - Assumption is that you have such a pair, or  $P$  can be checked for plausibility in some other way  
*The quality of seeming reasonable or probable.*
- 2 Check for each key  $K_i$  of the key space, whether
$$DES_{K_i}(P) = C \text{ or } DES_{K_i}^{-1}(C) = P \text{ holds}$$
  - If you succeed, the key is found

## Conclusion

- *DES is considered insecure today and should not be used!*

## How to use the complementarity property of DES?

Why is the complementarity property of DES useful for an attacker?

- Given are two plaintexts  $P$  and  $\bar{P}$ , furthermore the attacker knows the corresponding ciphertexts  $C_1$  and  $C_2$  with

$$C_1 = DES_K(P) \text{ and } C_2 = DES_K(\bar{P})$$

### Procedure for Calculating the Key

- 1 Select a  $K_1$  of the key space and calculate

$$C = DES_{K_1}(P)$$

- If  $C = C_1$  then  $K_1$  is the key you are looking for
  - If  $C = \bar{C}_2$  then  $\bar{K}_1$  is the key you are looking for
- 2 If  $C \neq C_1$  and  $C \neq \bar{C}_2$  than we exclude two keys ( $K_1$  and  $\bar{K}_1$ ) from the key space and try Step 1 using another key

The **cost** of a brute-force attack using the **complementarity property** is **half** that of a **normal brute-force attack**, i.e. it is reduced to  $2^{55}$ !

## Background: Complementarity Property of DES

Why is the following statement correct?

Assuming ...

1  $C1 = DES_K(P)$  and  $C2 = DES_K(\bar{P})$  and

2 for a test key  $K1$  holds  $\bar{C2} = DES_{K1}(P)$

... than the key you are looking for is  $K = \bar{K1}$

### Complementarity Property

$$DES_K(P) = \overline{DES_{\bar{K}}(\bar{P})}$$

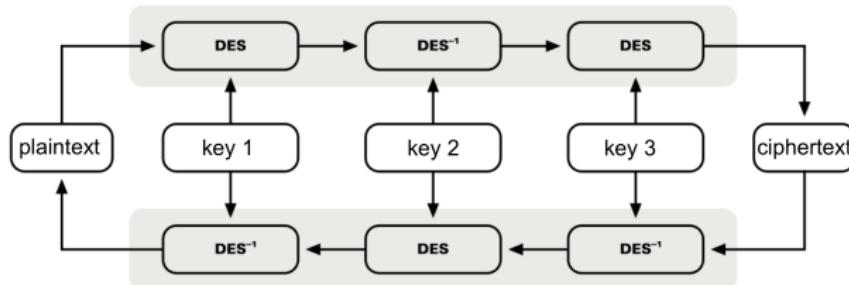
### Proof

$$C2 = DES_K(\bar{P}) = \overline{DES_{\bar{K}}(\bar{\bar{P}})} = \overline{DES_{\bar{K}}(P)}$$

$$\Leftrightarrow \bar{C2} = DES_{\bar{K}}(P)$$

→ If  $\bar{C2} = DES_{K1}(P)$ , than holds  $K1 = \bar{K}$  and  $K = \bar{K1}$

# How much better is Triple-DES?



## Using two different keys

- Order of the used keys:  $K_1-K_2-K_1$
- Size of the key space:  $2^{56} * 2^{56} = 2^{112}$

How much better is  
Triple-DES compared to  
DES? How works a  
Meet-in-the-Middle attack?

## Using three different keys

- Order of the used keys:  $K_1-K_2-K_3$
  - Size of the key space:  $2^{56} * 2^{56} + 2^{56} = 2^{112} + 2^{56}$
- *Meet-in-the-Middle Attack*

# Meet-in-the-Middle Attack for Triple-DES

- Let  $P$  a plaintext and  $C$  a ciphertext with  
$$C = DES_{K_3}(DES_{K_2}^{-1}(DES_{K_1}(P)))$$

How much better is  
Triple-DES compared to  
DES? How works a  
Meet-in-the-Middle  
attack?

## Procedure

- 1 Calculate forward

$$\forall K_1 \bullet C_1 = DES_{K_1}(P)$$

→  $2^{56}$

- 2 Calculate forward

$$\forall C_1, K_2 \bullet DES_{K_2}^{-1}(C_1) = P'$$

→  $2^{56} * 2^{56}$

- 3 Calculate backward and check, whether  
 $P' = P''$  holds

$$\forall K_3 \bullet DES_{K_3}^{-1}(C) = P''$$

→  $2^{56} * 2^{56} + 2^{56}$

## Note

- The attack requires disk space for  $2^{56}$  blocks, i.e. for a block length  
of 64 bits you need 576.460 terabyte!

## **AES – Advanced Encryption Standard**

– Symmetric-key Concelation and Authentication –

# History of AES

---

- Call for proposals in 1997 initiated by the NIST (*National Institute of Standards and Technology*) for the development of a modern symmetric-key encryption standard
- In a first step, 5 promising candidates were selected from 15 submissions
  - MARS (IBM)
  - RC6 (USA)
  - Rijndael (Belgium)
  - Serpent (Europa)
  - Twofish (USA)
- In 2000, the winner was awarded and the Rijndael algorithm was officially named AES

# Characteristics of AES (as contrast to DES)

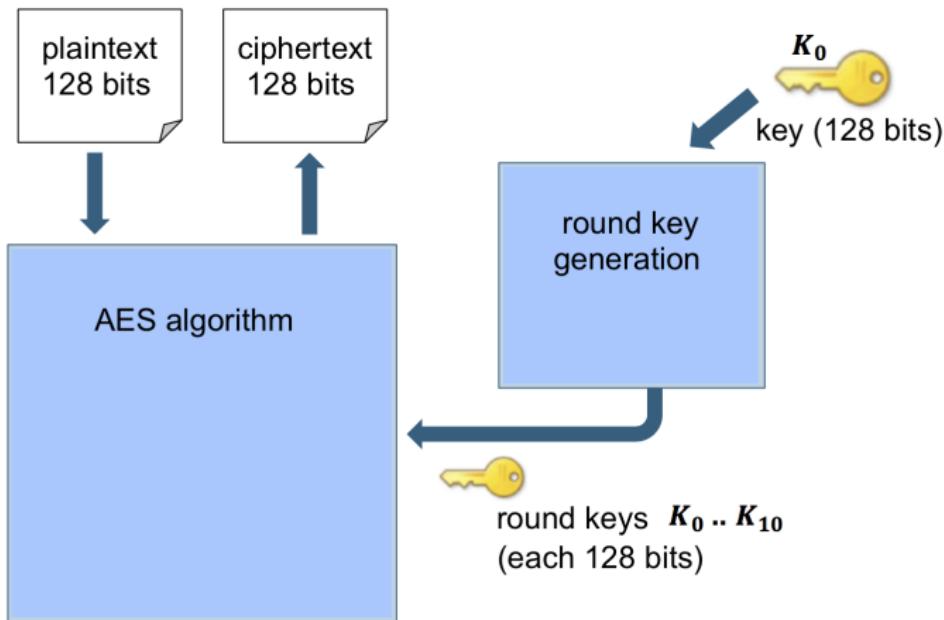
---

How many rounds are required for the AES algorithm?

- Block cipher that can process input blocks with a length of 128 bits
- 3 variants with different key lengths  
→ 128, 192 or 256 bits
- Is based on a substitution-permutation network, not a Feistel network, i.e. encryption function must be inverted for decryption
- Number of rounds depends on key length  
→ 10, 12 or 14 rounds
- Calculations are based on polynomial arithmetic (addition & multiplication) in the galois field  $GF(2^8)$

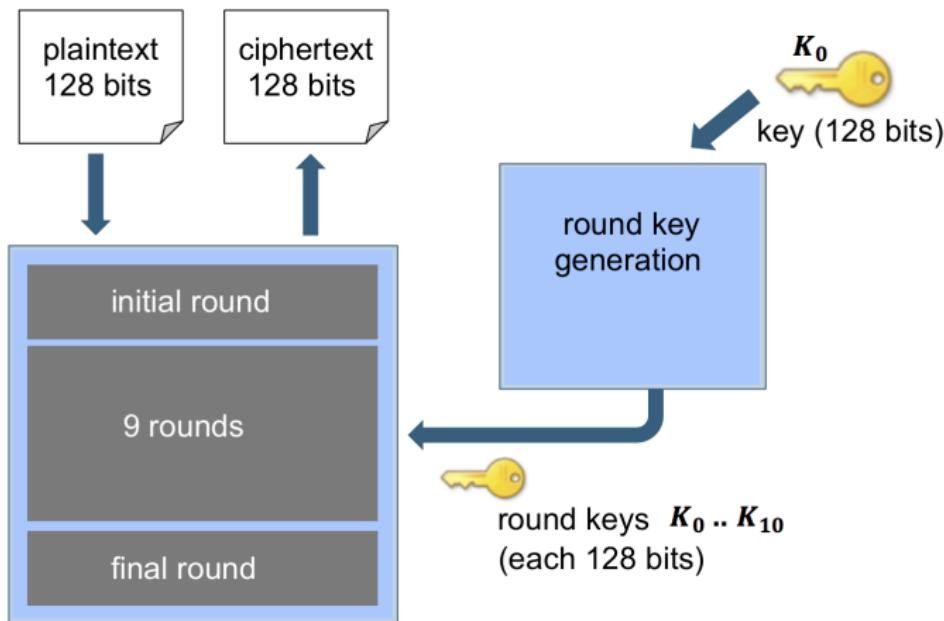
# Abstract View on the 128-AES Algorithm

---



Grafik: B. Ortmann

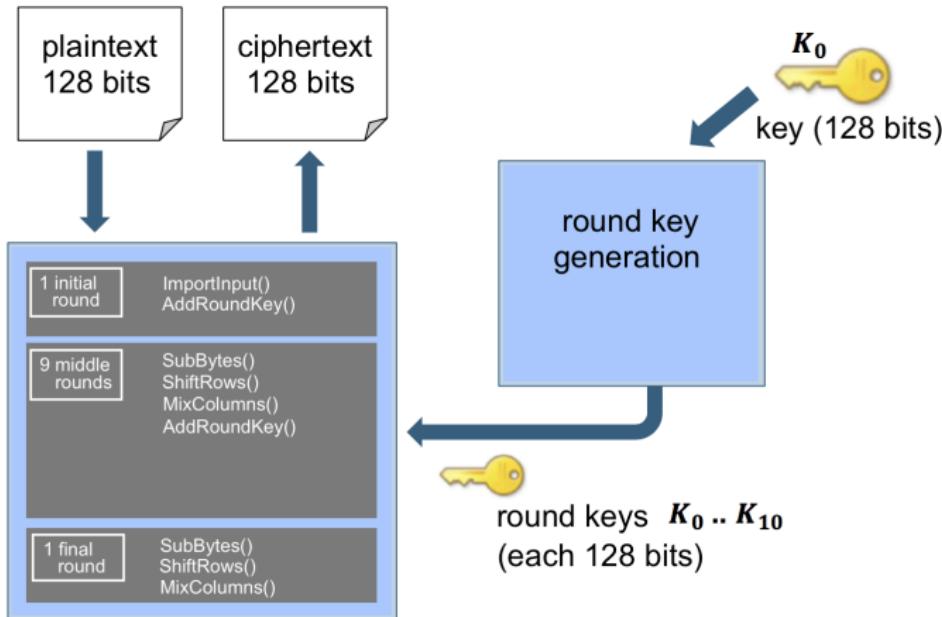
# Refined View on the AES-128 Algorithm



Grafik: B. Ortmann

# Which Operations have to be implemented for AES?

Which four operations need to be implemented for AES? Which of these operations are based on polynomial arithmetic?

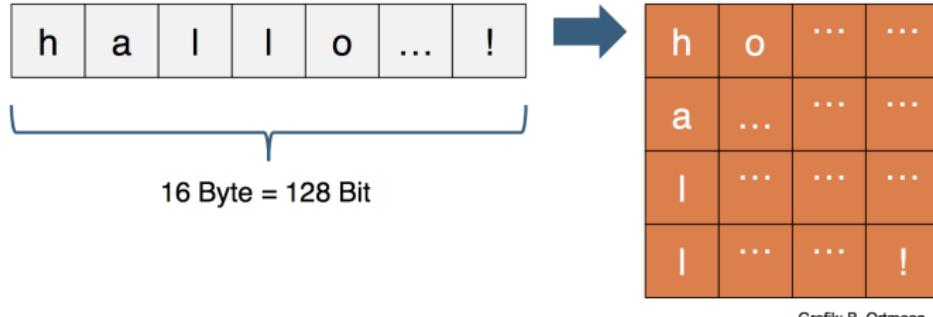


Grafik: B. Ortmann

# How to import the input block?

## Operation *ImportInput()*

- Internal state consists of a  $4 \times 4$  matrix
- Each field of the matrix represents one byte (also often represented as hex)
- Plaintext of 128 bits is imported column by column



Grafik: B. Ortmann

# AES-Operations

---

## Operation *AddRoundKey()*

- Based on *polynomial addition*
- Column-by-column addition of state matrix and round key
- Only this operation depends on the key

## Operation *SubBytes()*

- Performs confusion
- Byte-by-byte substitution using a predefined S-Box

## Operation *ShiftRows()*

- Performs permutations
- Line-by-line cyclic left shifts (each line one shift more)

## Operation *MixColumns()*

- Based on *polynomial multiplication*
- Column-by-column multiplication using a predefined matrix

## **Fundamentals: Polynomial Calculation**

# How to represent bytes using polynomials?

---

- Operations of AES are defined for the *Galois field  $GF(2^8)$*
- i.e. each byte of the state matrix is interpreted as a polynomial

polynomial	binary	hex
0	00000000	00
1	00000001	01
$x$	00000010	02
$x + 1$	00000011	03
$x^2$	00000100	04
$x^2 + 1$	00000101	05
...	...	...
$x^5 + x^3 + x^2 + x + 1$	00101111	2F
...	...	...

# Polynomial Arithmetic for $GF(2^8)$

## How to implement polynomial arithmetic efficiently for AES?

### General Remarks

- Similar to residue classes of the number theory Galois fields are based on *finite field arithmetic*
- For  $GF(2^8)$  we allow only polynomials of the form  $c \cdot x^e$  with  $c \in \{0, 1\}$  and  $e \in \{0, \dots, 7\}$

### Examples

- Polynomial  $(x^3 + x + 1)$  is in  $GF(2^8)$
- However the polynomials  $(x^8 + x + 1)$  and  $(2 \cdot x^3 + x + 1)$  are out of range

### Calculations

- Calculation results that are out of range have to be reduced by a suitable modulo operation
- For addition and subtraction we reduce the result using mod 2
- For multiplication we use an *irreducible reducing polynomial*

# Polynomial Arithmetic for $GF(2^8)$

---

## Example: Addition

- Addition results have to be reduced by mod 2

$$\begin{array}{r} \left( \begin{array}{c} x^3 \\ + x + 1 \end{array} \right) \\ \oplus \left( \begin{array}{c} x^3 \\ + x^2 \end{array} \right) \\ \hline \left( \begin{array}{c} 2 \cdot x^3 \\ + x^2 + x + 1 \end{array} \right) \\ \left( \begin{array}{c} 0 \cdot x^3 \\ + x^2 + x + 1 \end{array} \right) \quad \text{mod 2} \\ \hline \left( \begin{array}{c} x^2 + x + 1 \end{array} \right) \end{array}$$

- Addition in  $GF(2^8)$  corresponds to XOR-operation

$$\begin{array}{r} \left( \begin{array}{c} x^3 \\ + 1 \end{array} \right) \quad (00001001) \\ \oplus \left( \begin{array}{c} x^3 + x^2 + x \end{array} \right) \quad \oplus \quad (00001110) \\ \hline \left( \begin{array}{c} x^2 + x + 1 \end{array} \right) \quad (00000111) \end{array}$$

- Addition and subtraction can be implemented identically

# Polynomial Arithmetic for $GF(2^8)$

## Example: Multiplication

- Multiplication results have to be reduced by an *irreducible polynomial*, for AES polynomial  $(x^8 + x^4 + x^3 + x + 1)$  is used

$$\begin{aligned} p &= (x^6 + x^4 + x^2 + x + 1) \cdot (x^7 + x + 1) \\ &= (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \end{aligned}$$

- Polynomial  $p$  is not in  $GF(2^8)$  → we have to reduce it

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1)$$

- Modulo  $(x^8 + x^4 + x^3 + x + 1)$  is calculated by division

$$\begin{array}{r} (x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) : (x^8 + x^4 + x^3 + x + 1) \\ \oplus (x^{13} \quad + x^9 + x^8 + x^6 + x^5 \quad ) = x^5 + x^3 \\ \hline ( \quad x^{11} \quad \quad \quad + x^4 + x^3 + 1) \\ \oplus ( \quad x^{11} \quad + x^7 + x^6 \quad + x^4 + x^3 \quad ) \\ \hline ( \quad \quad \quad x^7 + x^6 \quad \quad \quad + 1) \end{array}$$

- We obtain for the reduced polynomial  $(x^7 + x^6 + 1)$

<https://www.youtube.com/watch?v=FUdEpUoA4mc>

# Polynomial Arithmetic for $GF(2^8)$

---

## Remarks

- Note, the AES algorithm requires only multiplications with the constants 1, 2 and 3
- This special cases can be implemented more efficiently

How to implement polynomial arithmetic efficiently for AES?

### Efficient Multiplications in $GF(2^8)$ for AES

$$1 \cdot b := b$$

$$2 \cdot b := \begin{cases} (\text{left shift of } b)^1 & \text{if } b < 128 \\ (\text{left shift of } b) \oplus 00011011 & \text{else} \end{cases}$$

$$3 \cdot b := (2 \cdot b) \oplus b$$

---

<sup>1</sup> Note that this is not a cyclical left shift, but a normal left shift where the right side is filled with zeros

## Example: AES Multiplication for *MixColumns()*

---

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} d4 \\ bf \\ 5d \\ 30 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix} = \begin{pmatrix} 04 \\ 66 \\ 81 \\ e5 \end{pmatrix}$$

hex	bin	dec
d4	11010100	212
bf	10111111	191
5d	01011101	93
30	00110000	48

How to calculate  $r_1$ ?

$$r_1 = 2 \cdot d4 \oplus 3 \cdot bf \oplus 1 \cdot 5d \oplus 1 \cdot 30$$

$$ir_1 = 2 \cdot d4 = 10101000 \oplus 00011011 = 10110011$$

$$ir_2 = 3 \cdot bf = 01111110 \oplus 00011011 \oplus bf = 01100101 \oplus 10111111 = 11011010$$

$$ir_3 = 1 \cdot 5d = 01011101$$

$$ir_4 = 1 \cdot 30 = 00110000$$

$$\begin{aligned} r_1 &= ir_1 \oplus ir_2 \oplus ir_3 \oplus ir_4 \\ &= 10110011 \oplus 11011010 \oplus 01011101 \oplus 00110000 = 00000100 \triangleq 04 \end{aligned}$$

# The complete AES algorithm – including round key generation

---

→ Animation to illustrate the AES algorithm

<https://www.youtube.com/watch?v=mlzxpkdXP58>

[http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael\\_Animation\\_v4\\_eng.swf](http://www.formaestudio.com/rijndaelinspector/archivos/Rijndael_Animation_v4_eng.swf)