

# Block Ciphers I

Martin Stanek

Department of Computer Science  
Comenius University  
`stanek@dcs.fmph.uniba.sk`

Cryptology 1 (2020/21)

# Content

## Introduction

- iterated ciphers, Feistel ciphers
- Simon and Speck

## AES (Advanced Encryption Standard)

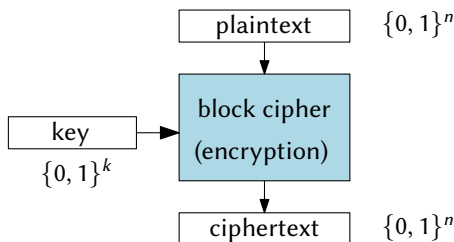
- description
- security

## Multiple encryption

- meet in the middle attacks, 3DES

## Slide attack

# Introduction - Block ciphers



- ▶ encryption/decryption  $E, D : \{0, 1\}^k \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
- ▶  $k$  – key length,  $n$  – block length
- ▶ correctness:  $\forall K \in \{0, 1\}^k \forall m \in \{0, 1\}^n : D_K(E_K(m)) = m$
- ▶  $E_K$  and  $D_K$  are mutually inverse permutations on  $\{0, 1\}^n$

# Block ciphers – examples

- ▶ more versatile than stream ciphers (modes of operation)
- ▶ used more often than stream ciphers
- ▶ AES – block length: 128, key lengths: 128, 192, 256
- ▶ 3DES (also TDEA) – block length: 64, key lengths: 112, 168
- ▶ NIST SP 800-131A rev. 2 (March 2019):
  - ▶ AES acceptable
  - ▶ 3DES (with 168-bits keys) deprecated through 2023, disallowed after 2023
- ▶ ISO standardized the following block ciphers:
  - ▶ ISO/IEC 18033-3:2010
    - 64-bits block: TDEA, MISTY1, CAST-128, HIGHT
    - 128-bits block: AES, Camellia, SEED
  - ▶ ISO/IEC 29192-2:2019 (Lightweight cryptography)
    - 64-bits block: PRESENT
    - 128-bits block: CLEFIA, LEA
- ▶ standardized  $\Rightarrow$  used

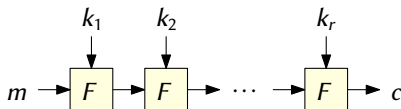
# Block ciphers – remarks

- ▶ alternative view: block cipher as a simple substitution
  - ▶ huge alphabet, frequency analysis impossible
- ▶ short block size – (possibly) easier cryptanalysis
- ▶ extremely short block size
  - ▶ small alphabet
  - ▶ max.  $(2^n)!$  permutations, regardless of key length
- ▶ extremely short key length:
  - ▶ exhaustive key search (brute-force attack)  $\sim 2^k$

# Security

- ▶ exhaustive key search (EKS) complexity  $\sim 2^k$
- ▶ expected EKS complexity  $\sim 2^{k-1}$
- ▶ important assumption: keys with uniform distribution (!)
  - ▶ otherwise enumerate keys by their probabilities (in descending order)
  - ▶ keys often derived from user passwords ( $\Rightarrow$  non-uniformity)
- ▶ (almost) anything with better complexity than EKS is a successful cryptanalytic attack (at least in theory)
- ▶ can be still impractical, because of
  - ▶ complexity, e.g.  $2^{120}$  instead of  $2^{128}$  is still infeasible
  - ▶ assumptions, e.g. CPA with  $2^{90}$  of chosen plaintext blocks encrypted with the same key is rather unrealistic

# Iterated ciphers



- ▶ the most frequently used construction method for block ciphers
- ▶ iteration of round function  $F : \{0, 1\}^{k'} \times \{0, 1\}^n \rightarrow \{0, 1\}^n$
- ▶ structure:
  - ▶ key scheduling/expansion: producing round keys  $k_1, \dots, k_r$  from the key
  - ▶ sequential iteration of  $F$  ( $r$  rounds):  $c = F_{k_r}(\dots F_{k_2}(F_{k_1}(m)) \dots)$
  - ▶ usually with some form of key whitening:  $c = k_{r+1} \oplus F_{k_r}(\dots F_{k_1}(m \oplus k_0) \dots)$
  - ▶ sometimes the first/the last round is different
- ▶ decryption employs inverse round function
- ▶ examples: AES-128 has 10 rounds, PRESENT has 32 rounds

# Feistel ciphers

- ▶ method of constructing a round function (its inverse has the same structure)
- ▶ decryption  $\sim$  encryption (with reversed order of round keys)  $\Rightarrow$  equal speed of encryption and decryption with pre-computed round keys
- ▶ plaintext divided into left and right halves:  $L_0, R_0$
- ▶ iterations (for  $i = 1, \dots, r - 1$ ):

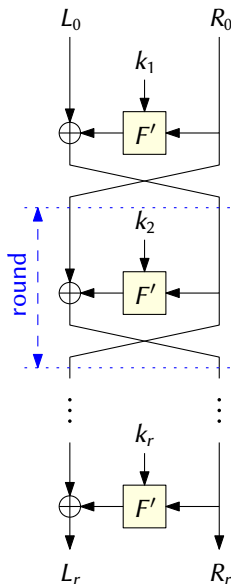
$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F'_{k_i}(R_{i-1})$$

- ▶ last round:

$$L_r = L_{r-1} \oplus F'_{k_r}(R_{r-1})$$

$$R_r = R_{r-1}$$



## Feistel ciphers – decryption

- ▶ using the same structure, changing the order of round keys
- ▶ denote  $L'_0 = L_r$  and  $R'_0 = R_r$  (and other intermediate values  $L'_i, R'_i$ )
- ▶ we can show that  $L'_i = R_{r-i}$  and  $R'_i = L_{r-i}$  for  $i = 1, \dots, r-1$ :
  - ▶ the first round:

$$\begin{aligned}L'_1 &= R'_0 = R_r = R_{r-1} \\ R'_1 &= L'_0 \oplus F'_{k_r}(R'_0) = L_r \oplus F'_{k_r}(R_{r-1}) = L_{r-1}\end{aligned}$$

- ▶ the second round (other rounds similarly):

$$\begin{aligned}L'_2 &= R'_1 = L_{r-1} = R_{r-2} \\ R'_2 &= L'_1 \oplus F'_{k_{r-1}}(R'_1) = R_{r-1} \oplus F'_{k_{r-1}}(R_{r-2}) = L_{r-2}\end{aligned}$$

- ▶ the last rounds (assuming  $L'_{r-1} = R_1$  and  $R'_{r-1} = L_1$ ):

$$\begin{aligned}R'_r &= R'_{r-1} = L_1 = R_0 \\ L'_r &= L'_{r-1} \oplus F'_{k_1}(R'_{r-1}) = R_1 \oplus F'_{k_1}(L_1) = L_0\end{aligned}$$

## Feistel ciphers – remarks

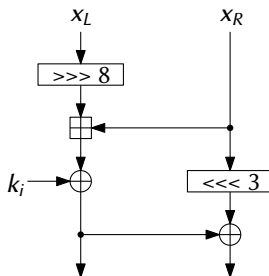
- ▶ examples: DES (3DES), Camellia, Blowfish, etc.
- ▶ generalizations:
  - unbalanced Feistel (splitting block into parts of unequal length)
- ▶ Feistel network is used in other cryptographic constructions, e.g.:
  - ▶ OAEP (Optimal Asymmetric Encryption Padding) for RSA encryption
  - ▶ format preserving encryption
- ▶ theoretical construction:
  - pseudorandom function  $\rightarrow$  pseudorandom permutation

# Simon and Speck

- ▶ lightweight block ciphers
  - ▶ families, variants with various block and key sizes
  - ▶ both ciphers with excellent performance in HW and SW
- ▶ published by NSA (2013)
- ▶ Simon
  - ▶ optimized for hardware, balanced Feistel network
  - ▶ XOR, bitwise AND, ROT (rotation)
- ▶ Speck
  - ▶ optimized for software, ARX cipher (modular addition, XOR, ROT)
- ▶ proposed as ISO standard in 2014
  - ▶ rejected in 2018 by subcommittee ISO/IEC JTC 1/SC 27 (Information security, cybersecurity and privacy protection)
  - ▶ standardized later in 2018 by other subcommittee ISO/IEC JTC 1/SC 31 (Automatic identification and data capture techniques)

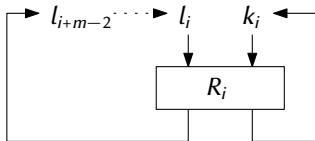
# Speck

- ▶ 10 variants of block/key lengths
  - ▶ starting with 32-bit block and 64-bit key ...
  - ▶ e.g. 128-bit block with 128, 192, or 256-bit key (32, 33, 34 rounds)
- ▶ **SPECK2n**
  - ▶  $2n$ -bit block (two  $n$ -bit words)
  - ▶ round function (round key  $k_i$ ):

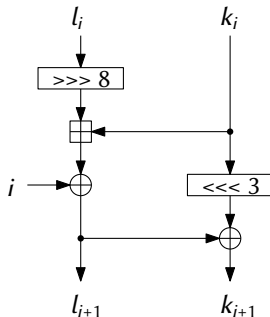


# Speck – key expansion

- ▶ a key  $K$  consists of  $m$  words,  $m \in \{2, 3, 4\}$  ( $m = |K|/n$ )
  - ▶ for example:  $m = 2$  for SPECK128/128,  $m = 4$  for SPECK128/256
- ▶  $K = (l_{m-2}, \dots, l_0, k_0)$
- ▶ round function is used for key expansion:



general scheme



case  $m = 2$

# AES (Advanced Encryption Standard)

- ▶ DES deficiency: short key length (56 bits)
- ▶ public standardization process for new encryption standard (1997–2000)
- ▶ requirements: block cipher, block length 128 bits, key lengths 128, 192, 256 bits
- ▶ Rijndael – winning algorithm (Vincent Rijmen, Joan Daemen)
- ▶ NIST standardized AES in 2001 (other standardizations followed)
- ▶ the most important symmetric cipher today
- ▶ used (almost) everywhere

# AES

- ▶ **not** a Feistel cipher
- ▶ different number of rounds depending on key length:  
AES-128 10 rounds, AES-192 12 rounds, AES-256 14 rounds
- ▶ slight performance degradation for longer key lengths

	1 thread (millions AES/s)	
	with AES-NI	no AES-NI
AES-128	42.8	7.1
AES-192	36.1	6.0
AES-256	31.4	5.3

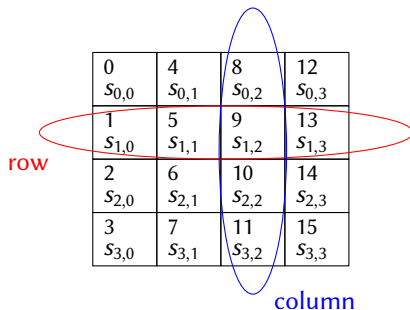
platform: i7-2600 @ 3.40 GHz (4 cores/8 threads, AES-NI)

implementation: openssl 1.0.1

overall encryption performance AES-128: 0.75 GB/s (1 thread, AES-NI)

# AES – state and operations

- ▶ state (plaintext, internal state, ciphertext):  $4 \times 4$  array of bytes:



The diagram shows a 4x4 state matrix. A red oval highlights the second row, with the word "row" in red text to its left. A blue oval highlights the third column, with the word "column" in blue text below it.

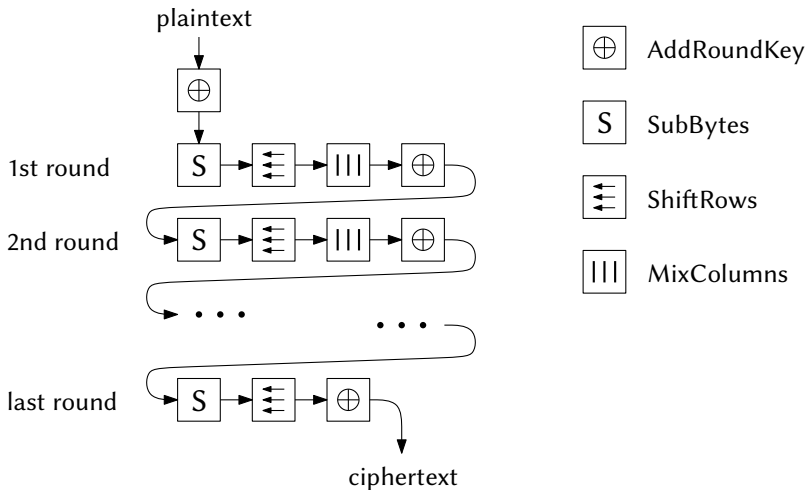
0 $s_{0,0}$	4 $s_{0,1}$	8 $s_{0,2}$	12 $s_{0,3}$
1 $s_{1,0}$	5 $s_{1,1}$	9 $s_{1,2}$	13 $s_{1,3}$
2 $s_{2,0}$	6 $s_{2,1}$	10 $s_{2,2}$	14 $s_{2,3}$
3 $s_{3,0}$	7 $s_{3,1}$	11 $s_{3,2}$	15 $s_{3,3}$

- ▶ 4 basic operations (invertible):
  1. AddRoundKey – XOR the state with 128-bit round key
  2. SubBytes – replace each byte using a fixed permutation (S-box)
  3. ShiftRows – cyclically shift each row of the state
  4. MixColumns – multiply each column by a fixed matrix

# AES – details of operations

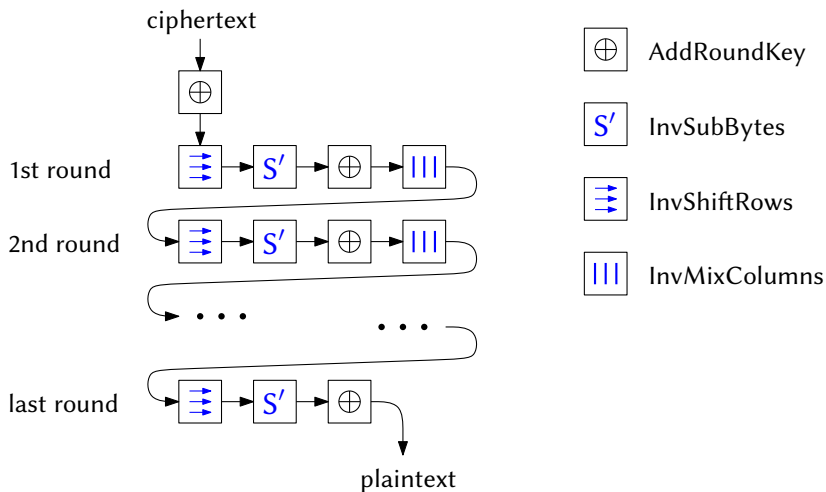
1. AddRoundKey: fast mix of round key in; self-inverse
2. SubBytes:  $s_{i,j} = S(s_{i,j})$  for all  $0 \leq i, j \leq 3$ 
  - ▶ the only nonlinear operation in AES
  - ▶ carefully chosen (a linear/affine ciphers are easy to break)
  - ▶ invertible: inverse permutation on  $\{0, 1\}^8$
3. ShiftRows:
  - ▶ 1st row is not shifted
  - ▶ 2nd/3rd/4th row: bytes are cyclically shifted to the left by 1/2/3 bytes
  - ▶ example:  $(s_{1,0}, s_{1,1}, s_{1,2}, s_{1,3}) \mapsto (s_{1,1}, s_{1,2}, s_{1,3}, s_{1,0})$
  - ▶ invertible: shift to the right
4. MixColumns
  - ▶ fixed (invertible !) matrix  $M$
  - ▶ good diffusion properties (small difference on input “amplifies”)

# AES – encryption structure



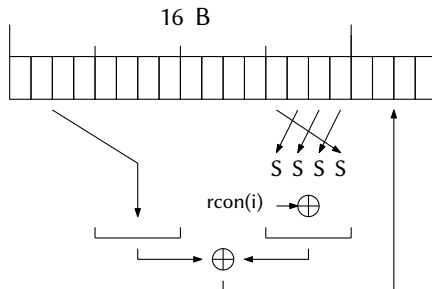
# AES – decryption structure

inverse operations: InvShiftRows, InvMixColumns, InvSubBytes



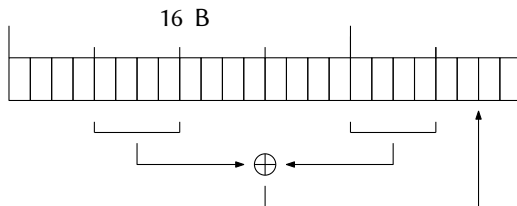
# AES – key expansion (for 128 bit key) 1

- ▶ AES-128  $\Rightarrow$  10 rounds  $\Rightarrow$  11 round keys (i.e.  $11 \cdot 16 = 176$  B)
- ▶ first 16 B (first round key) is the encryption key
- ▶  $\text{rcon}(i)$  – round constant
- ▶ 1st 4-byte word in each new round key:



## AES – key expansion (for 128 bit key) 2

- ▶ for the 2nd, 3rd and 4th 4-byte word in each round key:



- ▶ round keys are formed from consecutive bytes of the expanded key
- ▶ slightly different key expansion for key length 256

# AES – security

- ▶ exhaustive key search complexity  $\sim 2^{128}$  or  $2^{192}$  or  $2^{256}$
- ▶ best key recovery attacks
  - ▶ Bogdanov et al. 2011, KPA:

	time	data
AES-128	$2^{126.2}$	$2^{88}$
AES-192	$2^{189.7}$	$2^{80}$
AES-256	$2^{254.4}$	$2^{40}$

- ▶ Tao and Wu 2015, KPA:

	time	data
AES-128	$2^{126.1}$	$2^{56}$
AES-192	$2^{189.9}$	$2^{48}$
AES-256	$2^{254.3}$	$2^{40}$

# Multiple encryption

- ▶ multiple encryption (cascade encryption)
  - ▶ using the same or different ciphers, usually with independent keys

$$E_{k_1, k_2}(p) = E'_{k_2}(E_{k_1}^*(p))$$

- ▶ possible goals:
  - ▶ increasing the key space
  - ▶ security (what if a cipher is broken ... use two or three distinct)
- ▶ some ciphers cannot be strengthened (the key space does not increase), regardless of cascade length
  - ▶ examples: simple substitution, Vigenere, permutation, Vernam, etc.

$$\forall k_1, k_2 \exists k \forall p : E_{k_1}(E_{k_2}(p)) = E_k(p)$$

- ▶ independence of keys can be crucial
  - ▶ example: using the same key in double Vernam cipher  $\Rightarrow$  no encryption

## 3DES (TDEA)

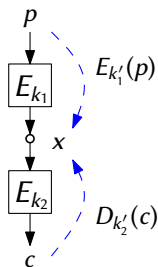
- ▶ 3DES is defined as a cascade of the length 3:
  - ▶ encryption:  $E_{k_3}(D_{k_2}(E_{k_1}(p)))$
  - ▶ decryption:  $D_{k_1}(E_{k_2}(D_{k_3}(c)))$
- ▶ keying options (and corresponding key length):
  - ▶ option 1: independent keys (168 bits)
  - ▶ option 2:  $k_1 = k_3$  (112 bits)
  - ▶ option 3:  $k_1 = k_2 = k_3$  (56 bits)
- ▶ EDE mode (instead of EEE mode) and keying option 3 ensures backward compatibility with DES
- ▶ real strength (bit security) of 3DES:
  - ▶ option 1: 112 bits (meet in the middle attack)
  - ▶ option 2: 80 bits (assuming  $2^{40}$  known plaintext/ciphertext pairs)

# Meet in the middle attack (MITM)

- ▶ disadvantage of multiple encryption – slower than single encryption
- ▶ why not “double encryption” – MITM attack
  - ▶ MITM is generally applicable to multiple encryption schemes
  - ▶ MITM is known plaintext attack (several pairs  $(p_i, c_i)$  known)

$$c = E_{k_2}(E_{k_1}(p))$$

1.  $\forall k'_2$ : compute  $x = D_{k'_2}(c)$  and store  $(x, k'_2)$  in a hash table indexed by  $x$
2.  $\forall k'_1$ : compute  $x = E_{k'_1}(p)$ 
  - 2.1 find entry(ies)  $(x, k'_2)$  in the table
  - 2.2 verify a candidate key(s)  $(k'_1, k'_2)$  using other plaintext/ciphertext pairs



# MITM – complexity

- ▶ assume key length  $k$  and block length  $n$
- ▶ expected number of required plaintext/ciphertext pairs  $\lceil 2k/n \rceil$ 
  - ▶  $\approx 2^{2k}/2^n$  “valid” key pairs for a single  $(p, c)$  pair
  - ▶  $\approx 2^{2k}/2^{tn}$  for  $t$  plaintext/ciphertext pairs
  - ▶ from  $1 \sim 2^{2k}/2^{tn}$  we get  $t \sim 2k/n$
- ▶ time complexity  $O(2^k)$ 
  - ▶ first cycle  $2^k$  iterations; second cycle  $2^k$  iterations
  - ▶ single hash table operation  $O(1)$
- ▶ space complexity  $O(2^k)$ 
  - ▶ each key  $k'_2$  produces one fixed-length entry in the hash table
  - ▶ second cycle in constant memory
- ▶ easily generalized for longer cascades
  - ▶ example: MITM on 3DES with 3 keys – time  $2^{112}$  and space  $2^{56}$

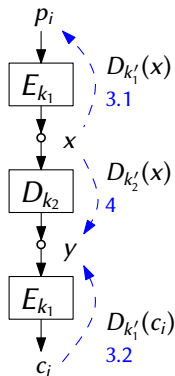
# A KPA on two-key triple encryption

- ▶ assume  $t$  plaintext/ciphertext pairs  $(p_i, c_i)_{i=1}^t$
- ▶ 3DES with keying option 2:  $c = E_{k_1}(D_{k_2}(E_{k_1}(p)))$

1. build a hash table T1 with  $(p_i, c_i)_{i=1}^t$  indexed by plaintexts

repeat steps 2–4 until the key pair is recovered:

2. choose random  $x$
3.  $\forall k'_1$ :
  - 3.1 compute  $p' = D_{k'_1}(x)$  and find  $(p' = p_i, c_i) \in T1$  (if exists)
  - 3.2 compute  $y = D_{k'_1}(c_i)$ , put  $(y, k'_1)$  into hash table T2 indexed by  $y$
4.  $\forall k'_2$ : compute  $D_{k'_2}(x)$ , find entry in T2, verify  $(k'_1, k'_2)$  pair



# Analysis of the attack

- ▶ time complexity:
  - ▶ building T1:  $O(t)$
  - ▶ for single  $x$ : step 3  $O(2^k)$ , step 4  $O(2^k)$
  - ▶ attack succeeds iff  $x$  is an intermediate value for some  $(p_i, c_i)$
  - ▶ expected number of tries to “guess” correct  $x$  (non-repeated guesses):
    - ▶  $(2^n - t)$  random variables  $X_z$  (for incorrect “middle” values)
    - ▶  $X_z$  is 0/1 r.v. (0 iff  $z$  is not guessed before any correct  $x$ )
    - ▶  $\text{Ex}$  denotes an expected value of random variable,  $\text{Ex}(X_z)$  is constant for all  $z$
    - ▶  $\text{Ex}(X_z) = 1/(t + 1)$  (counting arrangements of  $z$  and  $t$  correct values)
    - ▶ guesses:  $1 + \text{Ex}(\sum_z X_z) = 1 + (2^n - t) \cdot \text{Ex}(X_z) \approx (2^n - t)/(t + 1) \approx 2^{n-\lg t}$
  - ▶ let's summarize:  $O(t + 2^k \cdot 2^{n-\lg t}) = O(t + 2^{k+n-\lg t})$
- ▶ space complexity:  $O(t + 2^{k-n} \cdot t)$ 
  - ▶ T1:  $O(t)$  fixed-length entries;
  - ▶ T2 (new one for each  $x$ ): expected number of entries  $\approx (2^k/2^n) \cdot t$
  - ▶ space for T2 can be reused (not needed after finishing with  $x$ )
- ▶ easily adjustable for EEE (instead of EDE) mode of triple encryption

# Examples

- ▶ 3DES with two key option:
  - ▶ parameters:  $k = 56, n = 64, t = 2^{40}$
  - ▶ time complexity:  $O(t + 2^{k+n-\lg t}) \approx 2^{120-40} = 2^{80}$
  - ▶ space complexity:  $O(t + 2^{k-n} \cdot t) \approx 2^{40}$
- ▶ Triple AES-128 (not used in practice) with two-key option:
  - ▶ parameters:  $k = 128, n = 128, t = 2^{60}$
  - ▶ time complexity:  $O(t + 2^{k+n-\lg t}) \approx 2^{256-60} = 2^{196}$
  - ▶ space complexity:  $O(t + 2^{k-n} \cdot t) \approx 2^{60}$
- ▶ different trade-offs for different  $t$  values

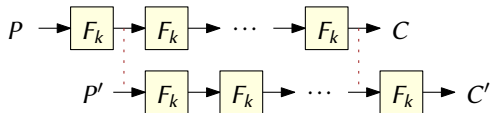
# Data requirements of KPA/CPA

- ▶ assumption: block length  $n = 128$
- ▶ only the ciphertext is considered for size computation, and for calculation of transmission time

data	size [TB]	time for 1Gb/s
$2^{40}$	17.6	39 hours
$2^{60}$	$1.8 \cdot 10^8$	4676 years
$2^{80}$	$1.9 \cdot 10^{13}$	$4.9 \cdot 10^9$ years
$2^{100}$	$2.0 \cdot 10^{19}$	$5.1 \cdot 10^{15}$ years

# Slide attack 1

- ▶ iterated ciphers
  - ▶ easy to change the number of rounds
  - ▶ usually more rounds  $\sim$  increased security
- ▶ Biryukov, Wagner (1999)
  - ▶ general attack on iterated cipher with identical round transform
  - ▶ arbitrary number of rounds
  - ▶ other variants exist
- ▶ cipher:  $C = F_k \circ F_k \circ \dots \circ F_k(P)$
- ▶ *slid pair* is a known pair of  $(P, C)$  and  $(P', C')$  such that  $P' = F_k(P)$  and  $C' = F_k(C)$



## Slide attack 2

- ▶ we assume that  $F_k$  is “weak”:
  - ▶ easy to compute  $k$  from equations  $y_0 = F_k(x_0)$ ,  $y_1 = F_k(x_1)$
  - ▶ usually very easy; for example, try this for Speck2n or AES
- ▶ KPA attack
  - ▶ approx.  $2^{n/2}$  of known plaintext-ciphertext pairs expecting  $\approx 1$  slid pair (birthday paradox)
  - ▶ testing all combinations if there is a slid pair  $(P, C), (P', C')$   
Is there  $k$  such that  $P' = F_k(P) \wedge C' = F_k(C) ? \dots (\approx 2^n)$
  - ▶ one slid pair recovers approx.  $n$  bits of the key
- ▶ Why bother when time complexity is  $O(2^n)$ ?
  - ▶ single round (slide attack) vs. full cipher (brute-force)
  - ▶ other improvements depending on  $F$

## Slide attack 3

- ▶ KPA and CPA slide attacks much better with Feistel ciphers
  - ▶ single round ... half of the block does not change
  - ▶  $\approx 2^{n/4}$  plaintext-ciphertext pairs for finding a slid pair
  - ▶ i.e. complexity is  $O(2^{n/2})$
- ▶ advanced variants of slide attack exist
- ▶ pay attention to key scheduling