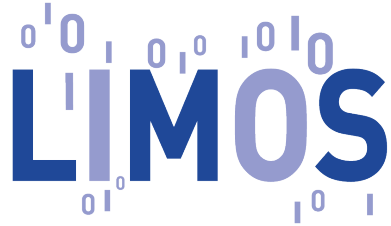
SSI Lecture 3
Symmetric Encryption

Pascal Lafourcade



2022-2023

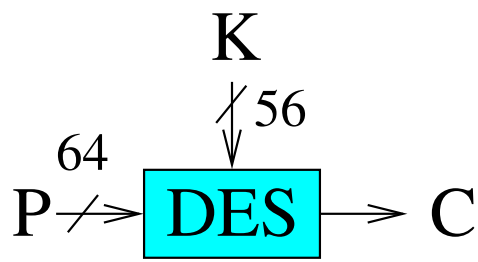
Outline

Outline

Data Encryption Standard, (call in 1973)

Lucifer designed in 1971 by Horst Feistel at IBM.

- Block cipher, encrypting 64-bit blocks
  Uses 56 bit keys, expressed as 64 bit numbers (8 bits parity checking)



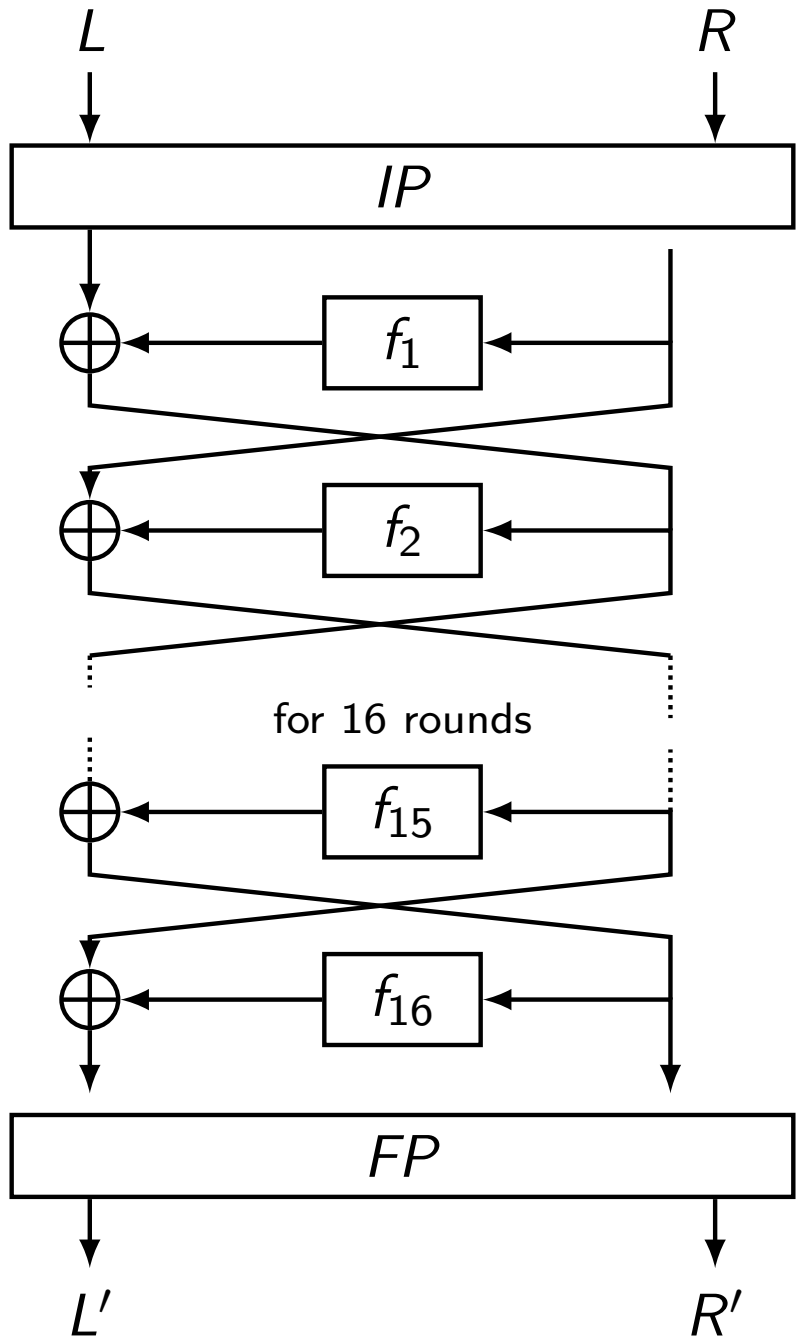
- First cryptographic standard.
  - 1977 US federal standard (US Bureau of Standards)
  - 1981 ANSI private sector standard

DES — overall form

- 16 rounds Feistel cipher + key-scheduler.
- Key scheduling algorithm derives subkeys K_i from original key K.
- Initial permutation at start, and inverse permutation at end.
- f consists of two permutations and an s-box substitution.

$L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$

DES — overall form



DES — Subkey generation

First, produce two subkeys K1 and K2:

$$K1 = P8(LS1(P10(key)))$$

$$K2 = P8(LS2(LS1(P10(key))))$$

where P8, P10, LS1 and LS2 are bit substitution operators.

- P10 : 10 bits to 10 bits

3	5	2	7	4	10	1	9	8	6

- P8 : 10 bits to 8 bits

6	3	7	4	8	5	10	9

- LS1 ("left shift 1 bit" on 5 bit words) : 10 bits to 10 bits

2	3	4	5	1	7	8	9	10	6

- LS2 ("left shift 2 bit" on 5 bit words) : 10 bits to 10 bits

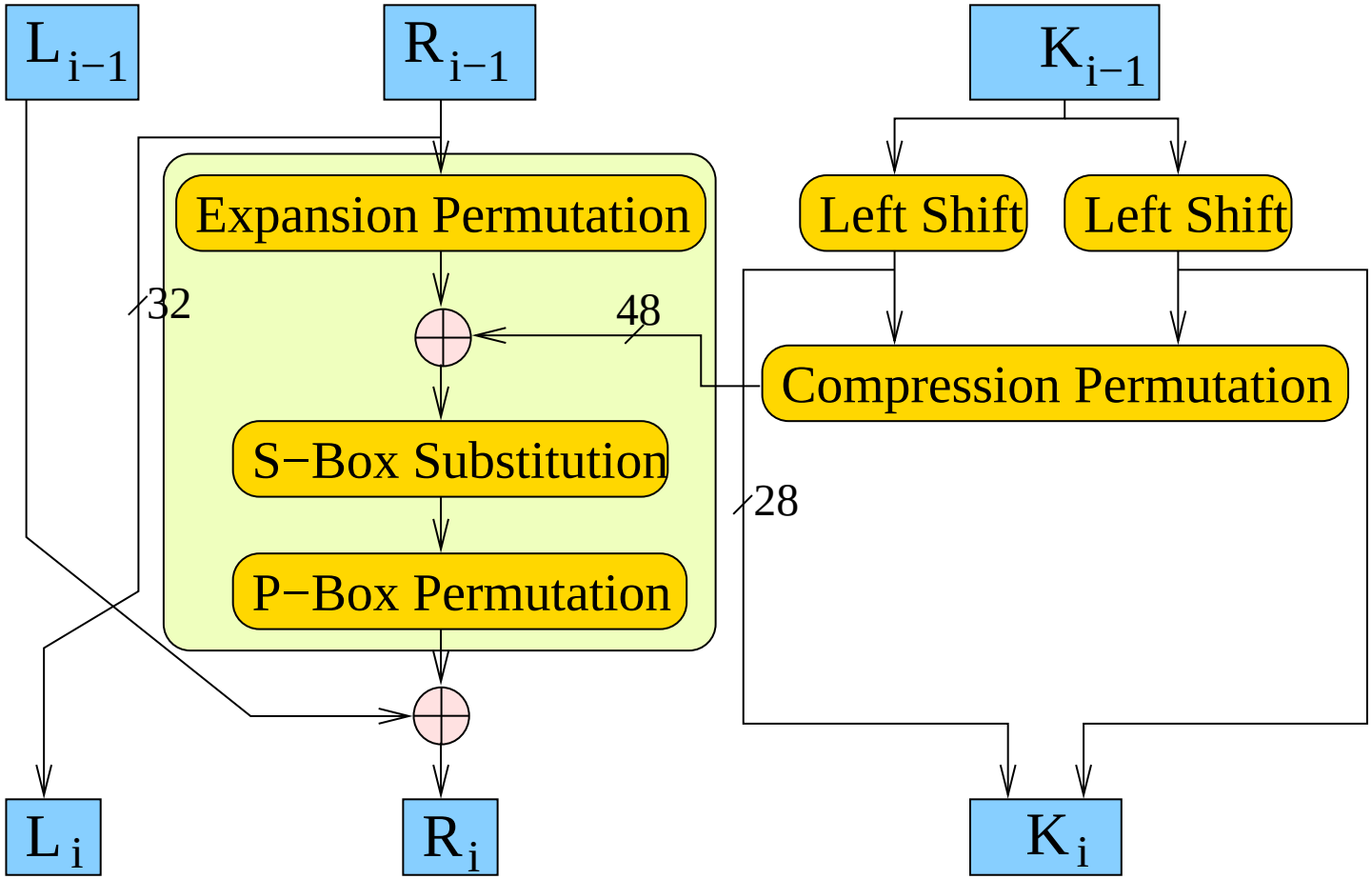
3	4	5	1	2	8	9	10	6	7

DES — Before round subkey

Each half of the key schedule state is rotated left by a number of places.

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

DES — 1 round



$(b_1 b_6, b_2 b_3 b_4 b_5)$, C_j represents the binary value in the row $b_1 b_6$ and column $b_2 b_3 b_4 b_5$ of the S_j box.

S-Boxes: S1, S2, S3, S4

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

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

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

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

S-Boxes: S5, S6, S7 and S8

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

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

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

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

Permutation P

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

Decryption DES

Use inverse sequence key.

- $IP(C) = IP(IP^{-1}(R_{16}||L_{16})$
- $L'_0 = R_{16}$ and $R'_0 = L_{16}$

$$L'_1 = R'_0 = L_{16} = R_{15}$$

$$R'_1 = L'_0 \oplus f(R'_0, K'_0)$$

$$R'_1 = R_{16} \oplus f(L_{16}, K_{15})$$

$$R'_1 = R_{16} \oplus f(R_{15}, K_{15})$$

$$R'_1 = L_{15}$$

Recall $L_{i+1} = R_i$ and $R_{i+1} = L_i \oplus f(R_i, K_i)$

Property of DES

DES exhibits the complementation property, namely that

$$E_K(P) = C \Leftrightarrow E_{\overline{K}}(\overline{P}) = \overline{C}$$

where $\overline{x}$ is the bitwise complement of x. E_K denotes encryption with key K. Then P and C denote plaintext and ciphertext blocks respectively.

Anomalies of DES

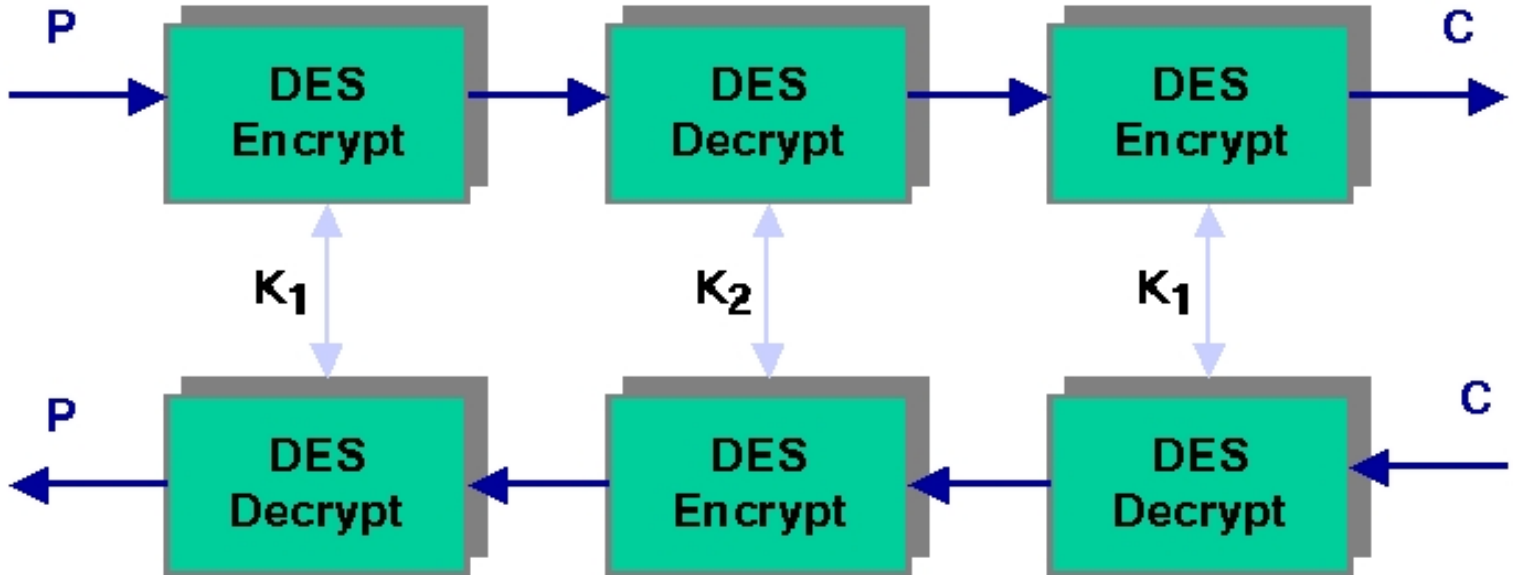
- Existence of 6 pairs of *semi-weak keys*: $E_{k_1}(E_{k_2}(x)) = x$.
  - 0x011F011F010E010E and 0x1F011F010E010E01
  - 0x01E001E001F101F1 and 0xE001E001F101F101
  - 0x01FE01FE01FE01FE and 0xFE01FE01FE01FE01
  - 0x1FE01FE00EF10EF1 and 0xE01FE01FF10EF10E
  - 0x1FFE1FFE0EFE0EFE and 0xFE1FFE1FFE0EFE0E
  - 0xE0FEE0FEF1FEF1FE and 0xFEE0FEE0FEF1FEF1

Security of DES

- No security proofs or reductions known
- Main attack: exhaustive search
  - 7 hours with 1 million dollar computer (in 1993).
  - 7 days with \$10,000 FPGA-based machine (in 2006).
- Mathematical attacks
  - Not know yet.
  - But it is possible to reduce key space from 2^{56} to 2^{43} using (linear) cryptanalysis.
    - To break the full 16 rounds, differential cryptanalysis requires 2^{47} chosen plaintexts (Eli Biham and Adi Shamir).
    - Linear cryptanalysis needs 2^{43} known plaintexts (Matsui, 1993)

Triple DES

- Use three stages of encryption instead of two.



- Compatibility is maintained with standard DES ($K_2 = K_1$).
- No known practical attack
  $\Rightarrow$ brute-force search with 2^{112} operations.

Advanced Encryption Standard

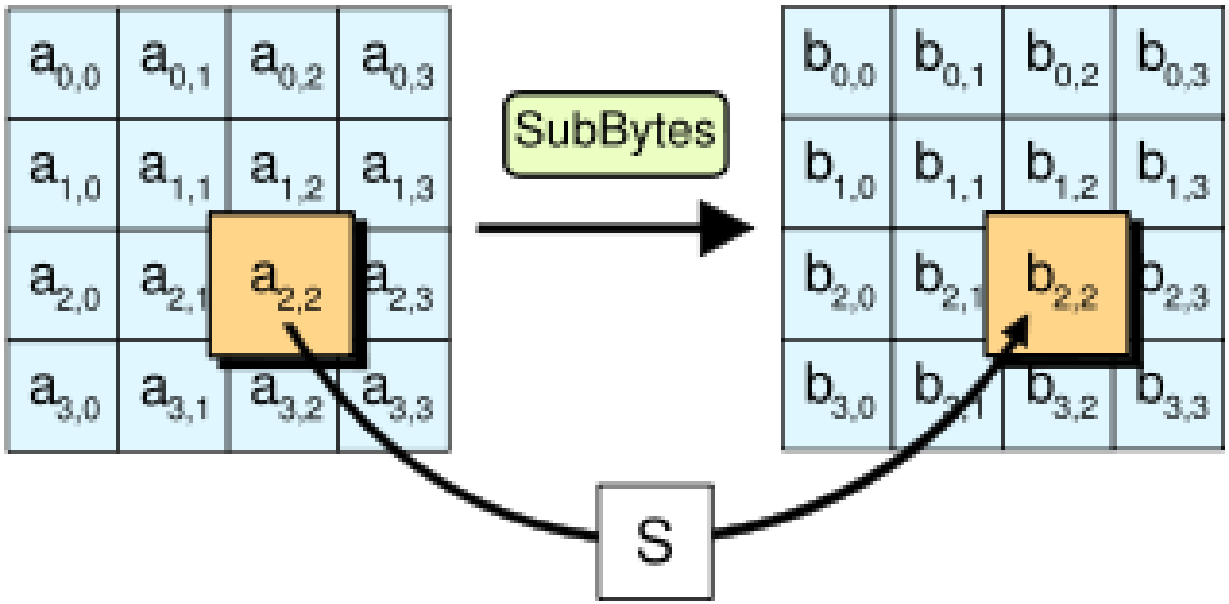
- Block cipher, approved for use by US Government in 2002. Very popular standard, designed by two Belgian cryptographers. Joan Daemen et Vincent Rijmen
- Block-size = 128 bits, Key size = 128, 192, or 256 bits.
- Uses various substitutions and transpositions + key scheduling, in different rounds.
- Algorithm believed secure. Only attacks are based on side channel analysis, i.e., attacking implementations that inadvertently leak information about the key.

Key Size	Round Number
128	10
192	12
256	14

AES: High-level cipher algorithm

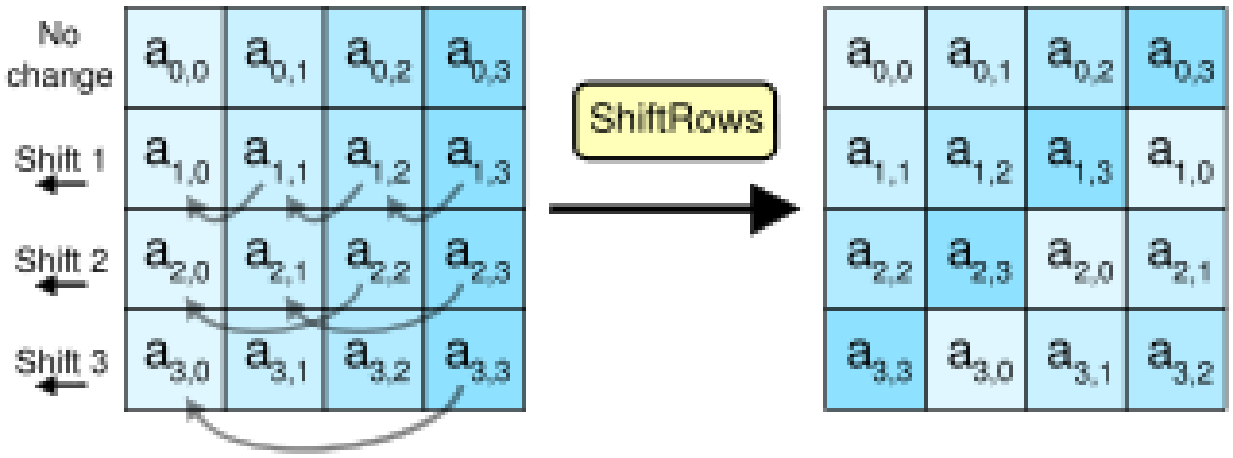
- KeyExpansion using Rijndael's key schedule
- Initial Round: AddRoundKey
- Rounds:
  1. SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.
  2. ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.
  3. MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
  4. AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.
- Final Round (no MixColumns)
  1. SubBytes
  2. ShiftRows
  3. AddRoundKey

AES: SubBytes



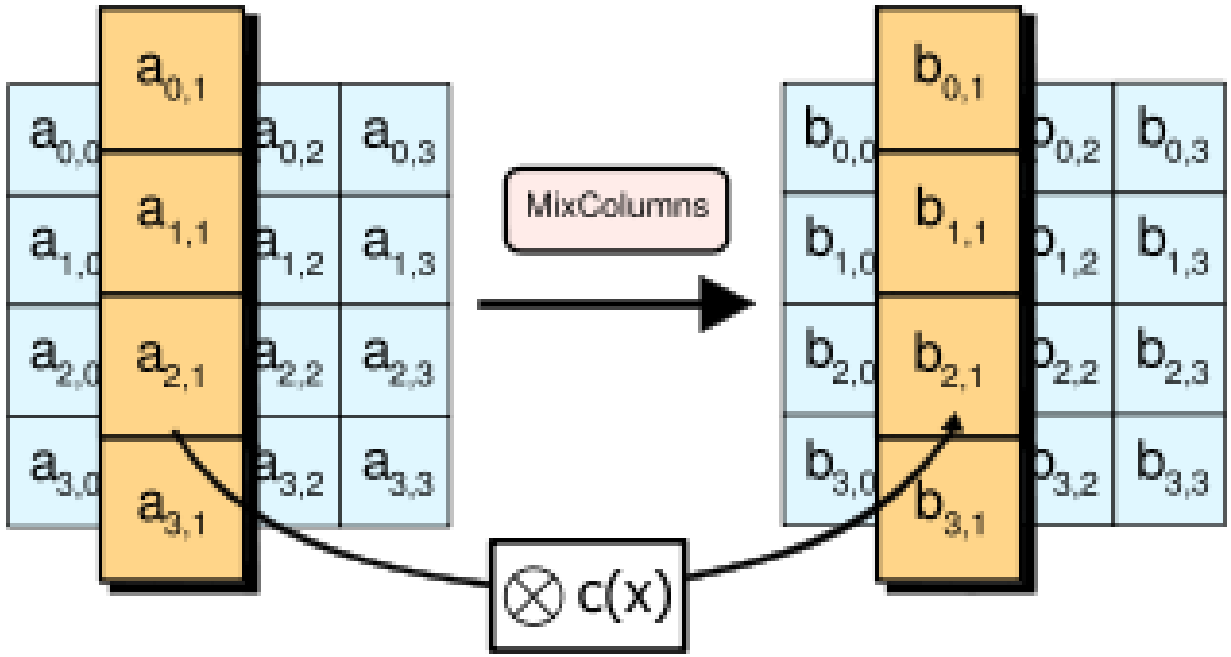
SubBytes: a non-linear substitution step where each byte is replaced with another according to a lookup table.

AES: ShiftRows



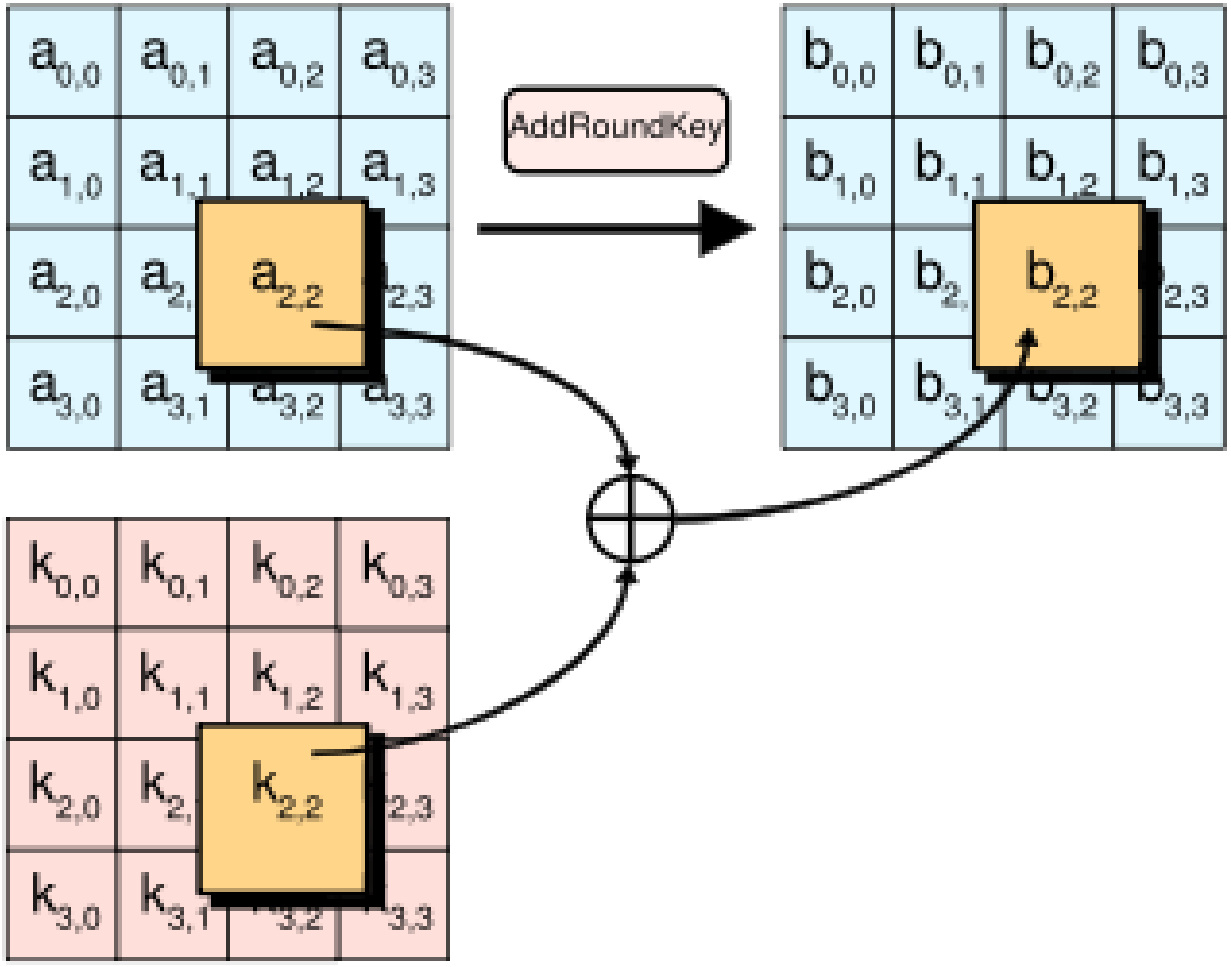
ShiftRows: a transposition step where each row of the state is shifted cyclically a certain number of steps.

AES: MixColumns



MixColumns: a mixing operation which operates on the columns of the state, combining the four bytes in each column
$C(x) = 3x^3 + x^2 + x + 2 Modulo : x^4 + 1$

AES: AddRoundKey



AddRoundKey: each byte of the state is combined with the round key; each round key is derived from the cipher key using a key schedule.

AES: Attacks

Not yet efficient Cryptanalysis on complete version, but Niels Ferguson proposed in 2000 an attack on a version with 7 rounds and 128 bits key.
But
Marine Minier, Raphael C.-W. Phan, Benjamin Pousse: Distinguishers for Ciphers and Known Key Attack against Rijndael with Large Blocks. AFRICACRYPT 2009: 60-76
Samuel Galice, Marine Minier: Improving Integral Attacks Against Rijndael-256 Up to 9 Rounds. AFRICACRYPT 2008: 1-15
Side channel attacks using on optimized version (2005)

- Timing.
- Cache Default.
- Electric Consumptions.
- ..

There exists algebraic attacks ...

IDEA: International Data Encryption Algorithm 1991

Designed by Xuejia Lai and James Massey of ETH Zurich, used in Pretty Good Privacy (PGP) v2.0
8.5 rounds
IDEA uses a message of 64-bit blocks and a 128-bit key,

Key schedule

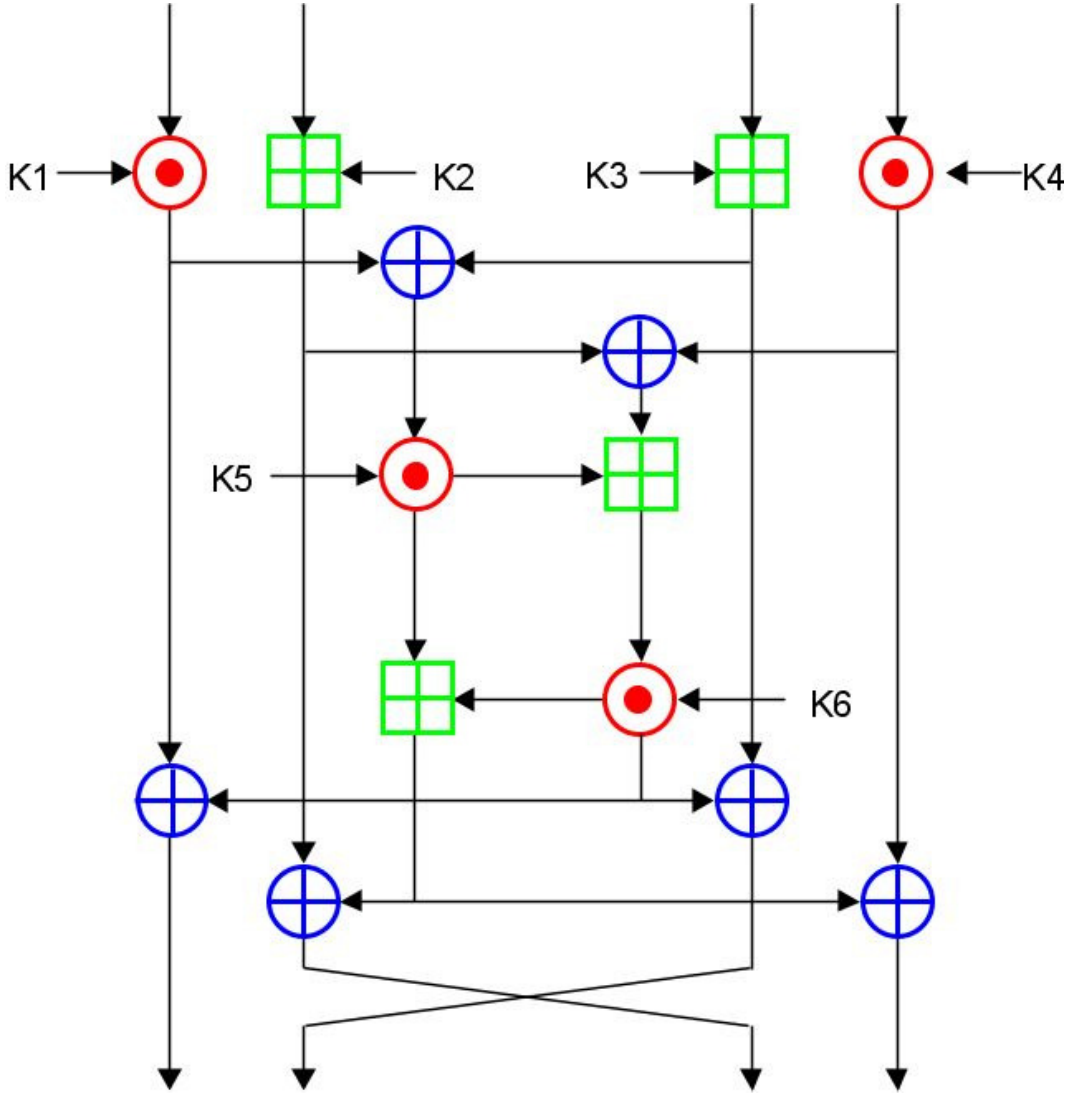
- K1 to K6 for the first round are taken directly as the first 6 consecutive blocks of 16 bits.
- This means that only 96 of the 128 bits are used in each round.
- 128 bit key undergoes a 25 bit rotation to the left, i.e. the LSB becomes the 25th LSB.

IDEA

Notation

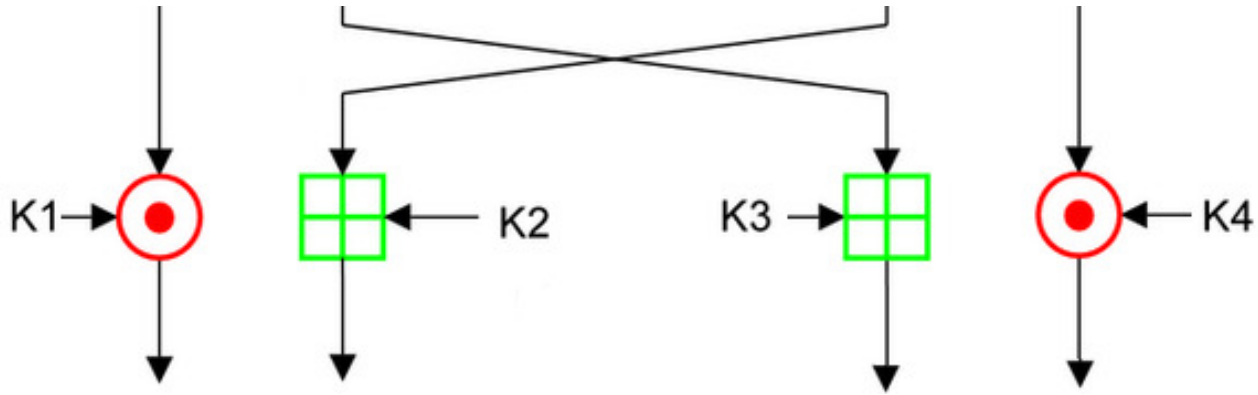
- Bitwise eXclusive OR (denoted with a blue $\oplus$).
- Addition modulo 216 (denoted with a green $\boxplus$).
- Multiplication modulo 216+1, where the all-zero word (0x0000) is interpreted as 216 (denoted by a red $\odot$).

IDEA



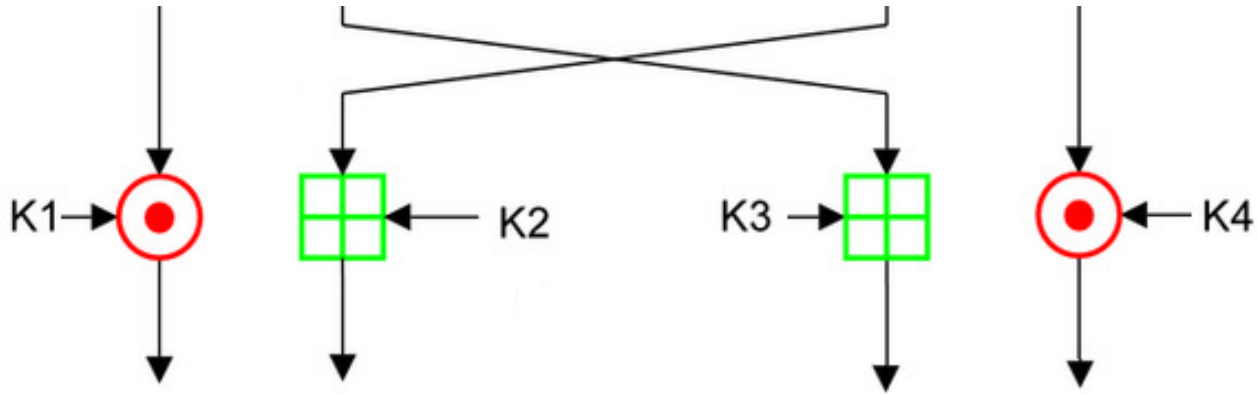
IDEA

After the eight rounds comes a final "half round".



IDEA

After the eight rounds comes a final "half round".



The best attack which applies to all keys can break IDEA reduced to 6 rounds (the full IDEA cipher uses 8.5 rounds) Biham, E. and Dunkelman, O. and Keller, N. "A New Attack on 6-Round IDEA".

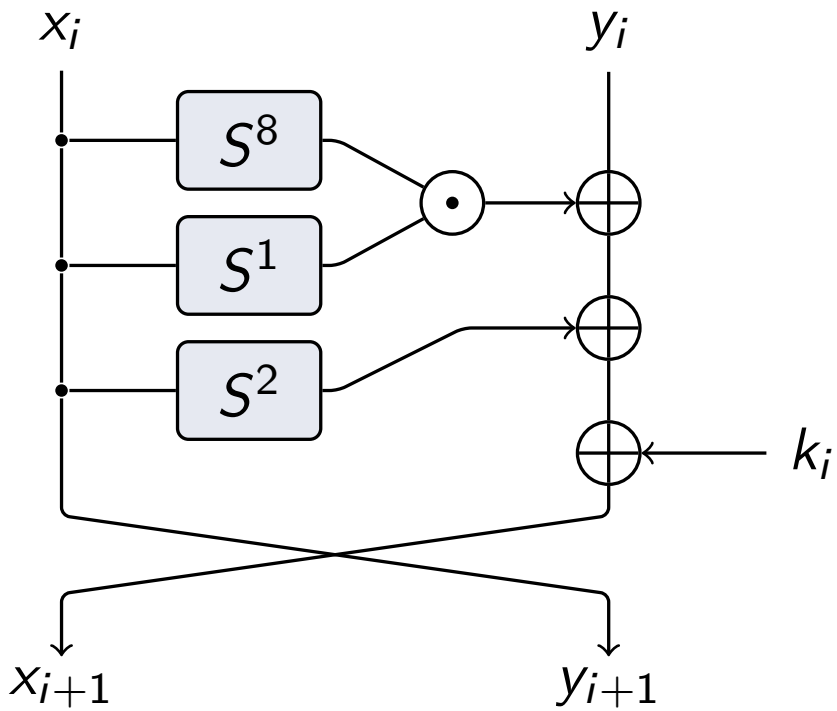
• Blowfish, invented by Schneier to be fast, compact, easy to implement, and to have variable key length (up to 448 bits),

SIMON

Proposed by NSA in June 2013.

Block size (bits)	Key size (bits)	Rounds
32	64	32
48	72	36
	96	36
64	96	42
	128	44
96	96	52
	144	54
128	128	68
	192	69
	256	72

One round of SIMON



S^8 = shift left by 8 bits
S^2 = shift left by 2 bits
S^1 = shift left by 1 bits
Using AND bitwise $\odot$ and XOR $\oplus$.

SIMON key schedule, $c = 2^n - 4$

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+1}\right), & m = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+2}\right), & m = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+3} \oplus k_{i+1}\right), & m = 4 \end{cases}$$

$(z_j)_i$ is defined by sequence of bits for each parameters.

Others Symmetric Encryption Schemes

Blowfish, Serpent, Twofish, 3-Way, ABC, Akelarre, Anubis, ARIA, BaseKing, BassOmatic, BATON, BEAR and LION, C2, Camellia, CAST-128, CAST-256, CIKS-1, CIPHERUNICORN-A, CIPHERUNICORN-E, CLEFIA, CMEA, Cobra, COCONUT98, Crab, CRYPTON, CS-Cipher, DEAL, DES-X, DFC, E2, FEAL, FEA-M, FROG, G-DES, GOST, Grand Cru, Hasty Pudding Cipher, Hierocrypt, ICE, IDEA, IDEA NXT, Intel Cascade Cipher, Iraqi, KASUMI, KeeLoq, KHAZAD, Khufu and Khafre, KN-Cipher, Ladder-DES, Libelle, LOKI97, LOKI89/91, Lucifer, M6, M8, MacGuffin, Madryga, MAGENTA, MARS, Mercy, MESH, MISTY1, MMB, MULTI2, MultiSwap, New Data Seal, NewDES, Nimbus, NOEKEON, NUSH, Q, RC2, RC5, RC6, REDOC, Red Pike, S-1, SAFER, SAVILLE, SC2000, SEED, SHACAL, SHARK, Skipjack, SMS4, Spectr-H64, Square, SXAL/MBAL, TEA, Treyfer, UES, Xenon, xmx, XTEA, XXTEA, Zodiac.

Meet-in-the-middle Attack

Double DES with k_1 and k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$

$P = DEC_{k_1}(DEC_{k_2}(C))$

Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$

Meet-in-the-middle Attack

Double DES with k_1 and k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$

$P = DEC_{k_1}(DEC_{k_2}(C))$

Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$

One Observation

$$DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)])$$
$$= ENC_{k_1}(P)$$

Meet-in-the-middle Attack

Double DES with k_1 and k_2

$C = ENC_{k_2}(ENC_{k_1}(P))$
$P = DEC_{k_1}(DEC_{k_2}(C))$
Brute force attaque : $2^{k1} * 2^{k2} = 2^{k1+k2}$

One Observation

$$DEC_{k_2}(C) = DEC_{k_2}(ENC_{k_2}[ENC_{k_1}(P)])$$
$$= ENC_{k_1}(P)$$

MITM Attack

- $ENC_{k_1}(P)$ for all values of k_1
- $DEC_{k_2}(C)$ for all possible values of k_2,

for a total of $2^{|k_1|} + 2^{|k_2|}$

Outline

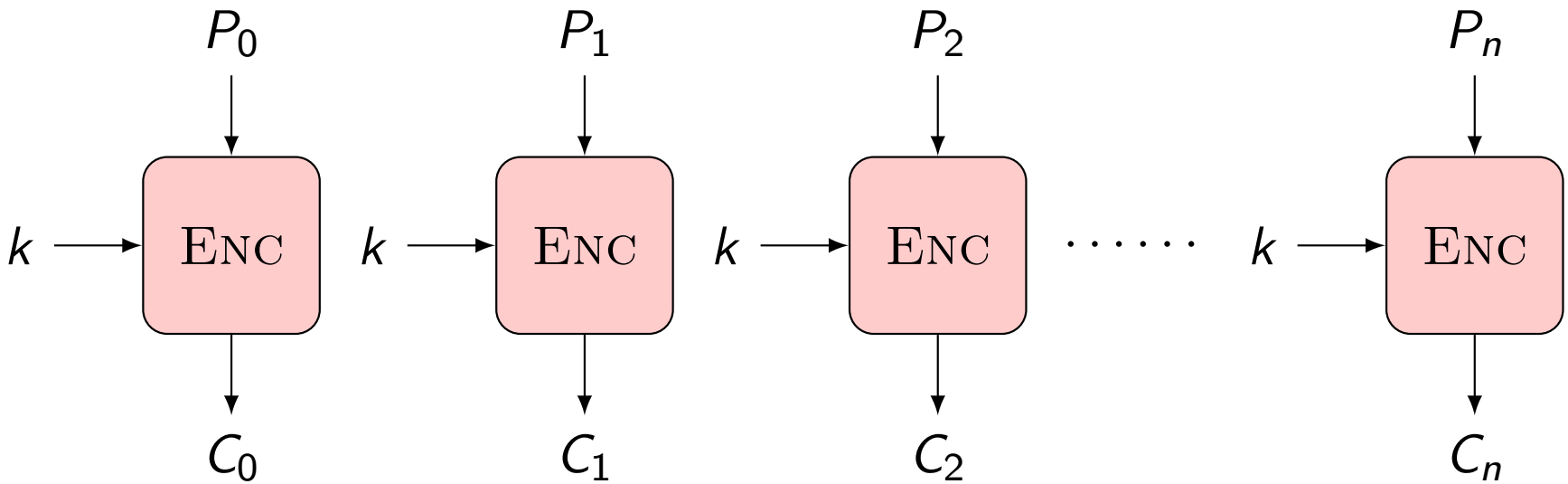
Electronic Book Code (ECB)

Each block of the same length is encrypted separately using the same key K. In this mode, only the block in which the flipped bit is contained is changed. Other blocks are not affected.

ECB Encryption Algorithm

algorithm $E_K(M)$
if ($|M|$ mod $n \neq 0$ or $|M| = 0$) then return FAIL
Break M into n-bit blocks $M[1] \ldots M[m]$
for $i = 1$ to m do $C[i] = E_K(M[i])$
$C = C[1] \ldots C[m]$
return C

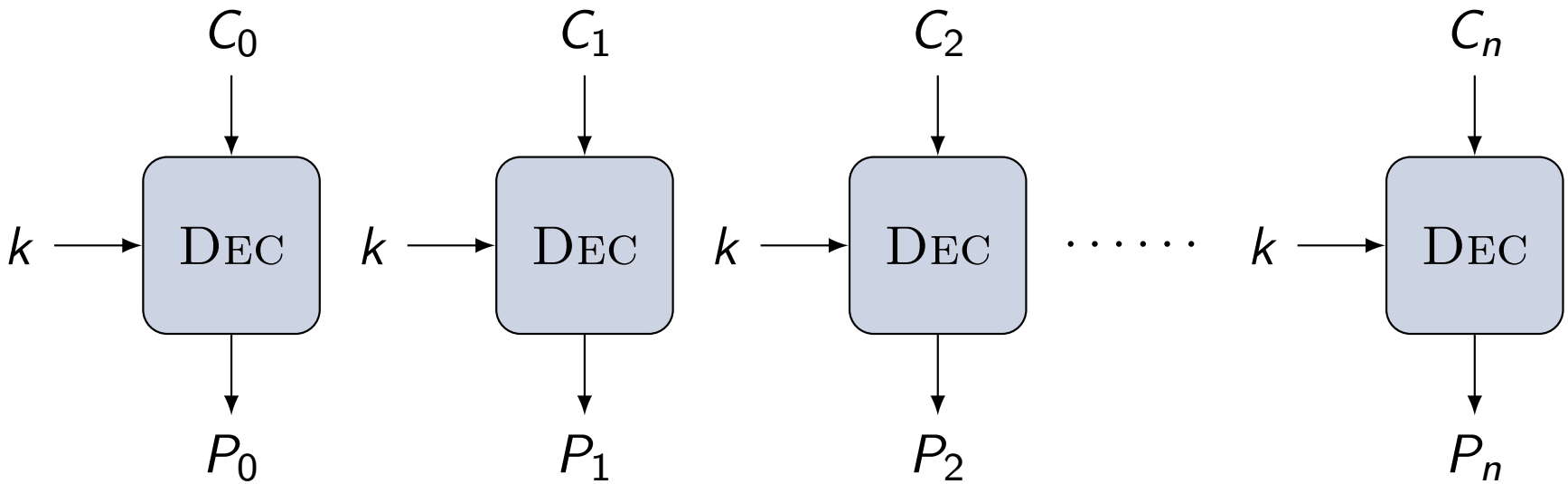
ECB Encryption



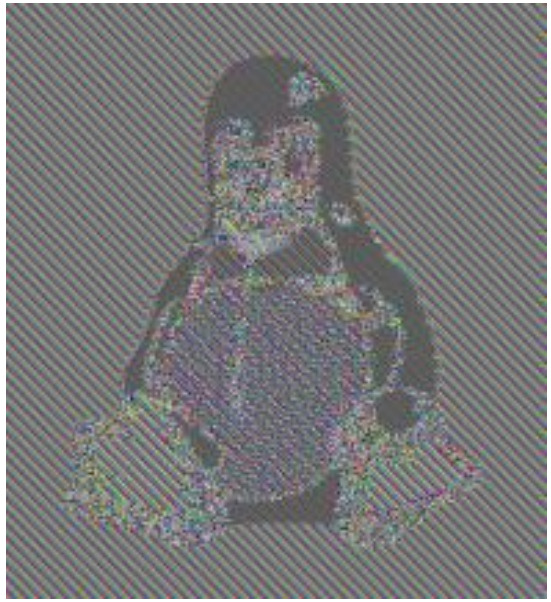
ECB Decryption Algorithm

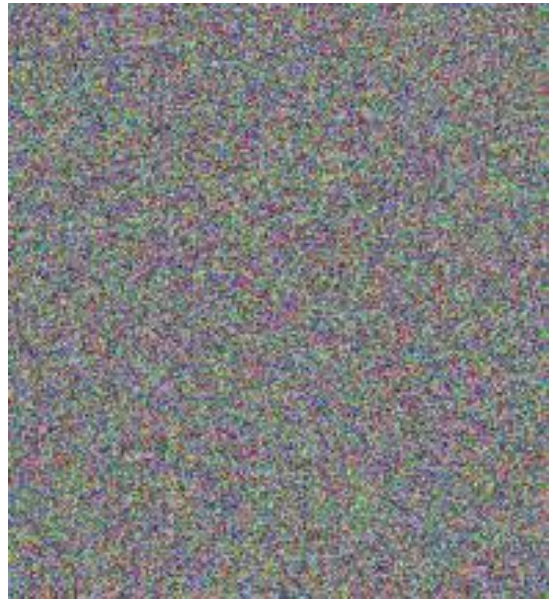
algorithm $D_K(C)$
if ($|C|$ mod $n \neq 0$ or $|C| = 0$) then return FAIL
Break C into n-bit blocks $C[1] \ldots C[m]$
for $i = 1$ to m do $M[i] = D_K(C[i])$
$M = M[1] \ldots M[m]$
return M

ECB Decryption



ECB vs Others





Cipher-block chaining (CBC)

If the first block has index 1, the mathematical formula for CBC encryption is

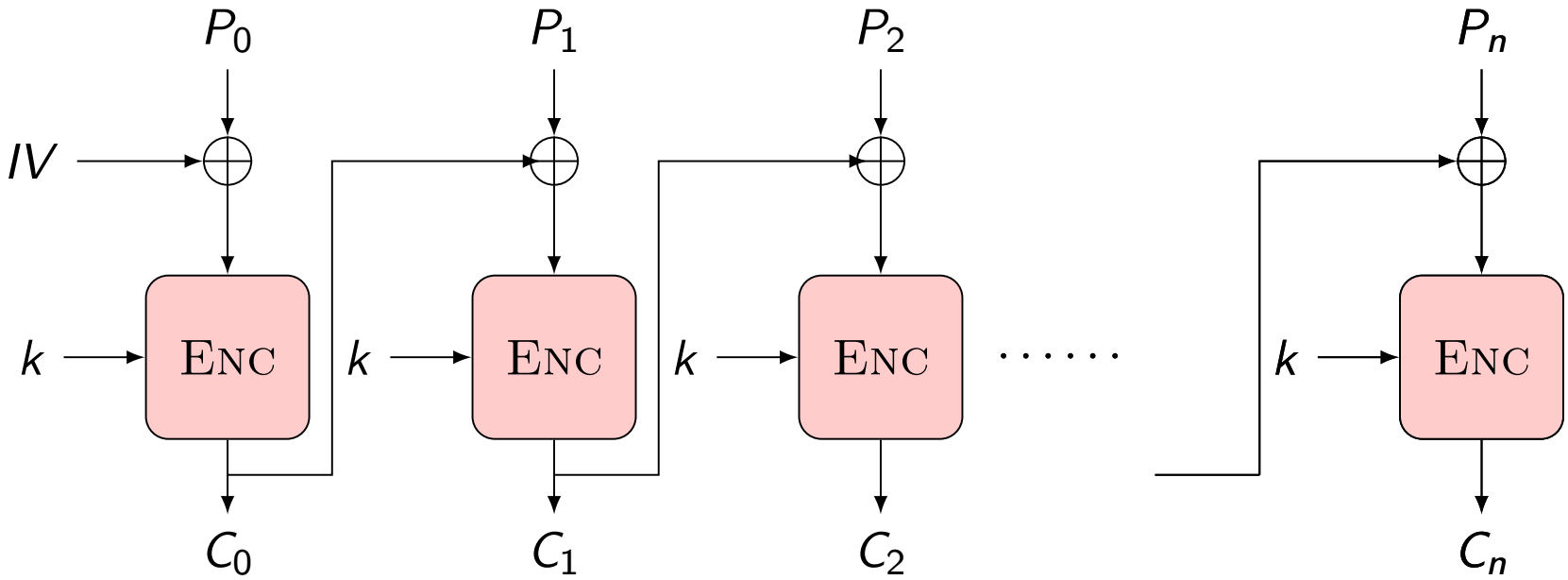
$$C_i = E_K(P_i \oplus C_{i-1}), C_0 = IV$$

while the mathematical formula for CBC decryption is

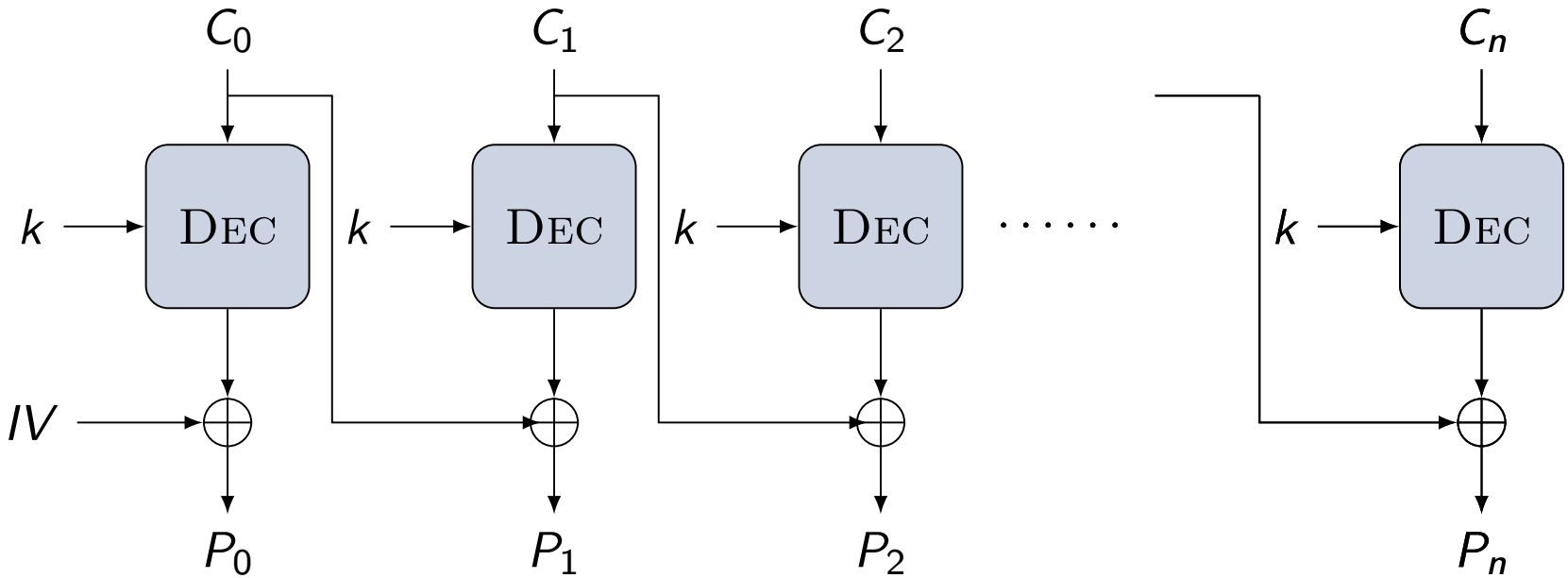
$$P_i = D_K(C_i) \oplus C_{i-1}, C_0 = IV$$

CBC has been the most commonly used mode of operation.

CBC Encryption



CBC Decryption



The cipher feedback (CFB)

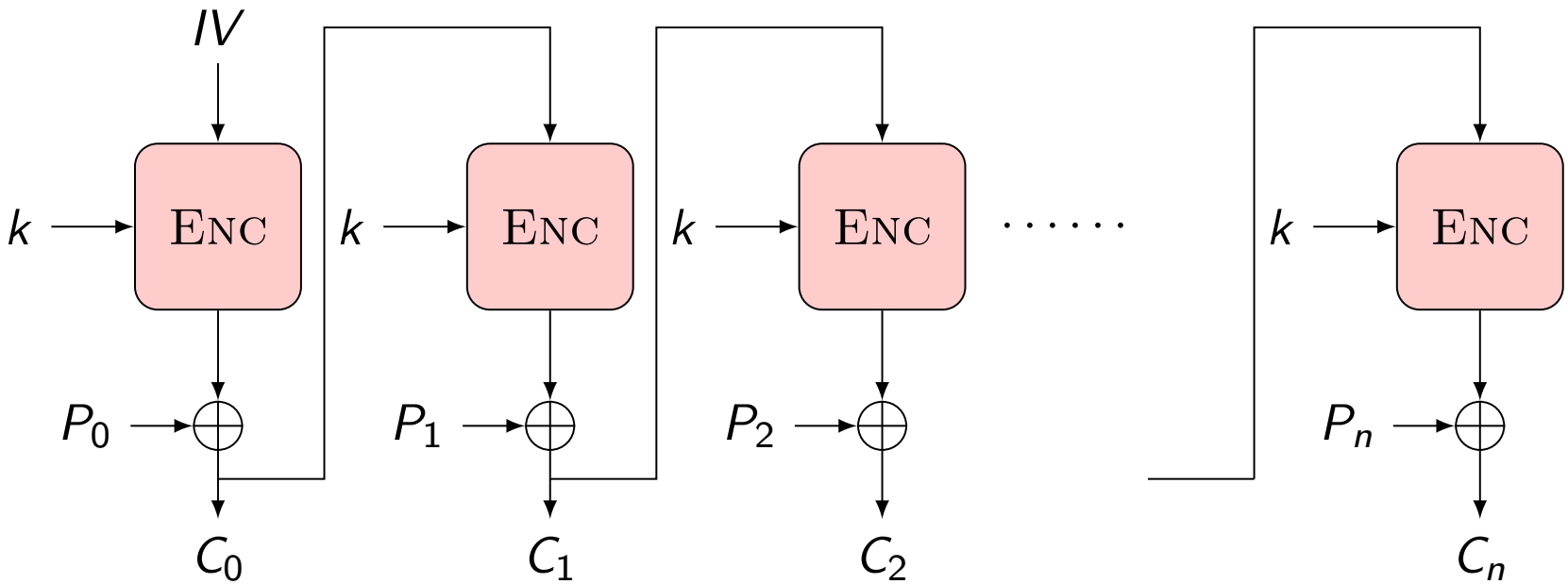
A close relative of CBC:

$$C_i = E_K(C_{i-1}) \oplus P_i$$

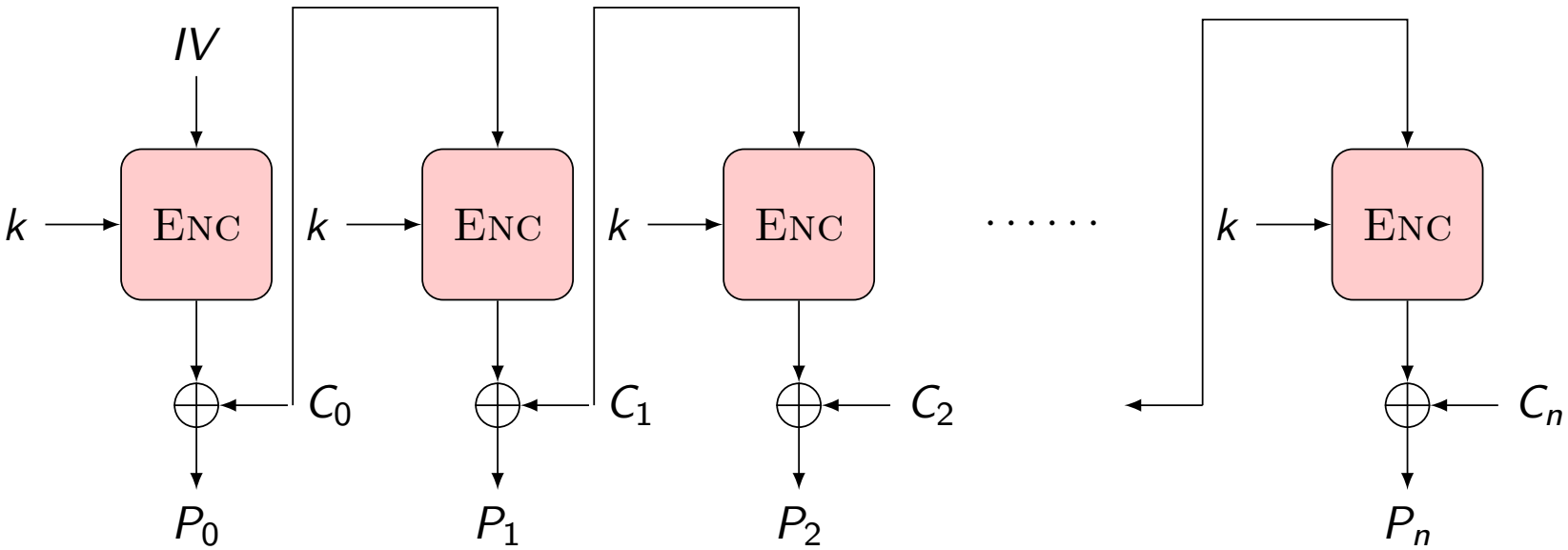
$$P_i = E_K(C_{i-1}) \oplus C_i$$

$$C_0 = \text{IV}$$

CFB Encryption



CFB Decryption



Output feedback (OFB)

Because of the symmetry of the XOR operation, encryption and decryption are exactly the same:

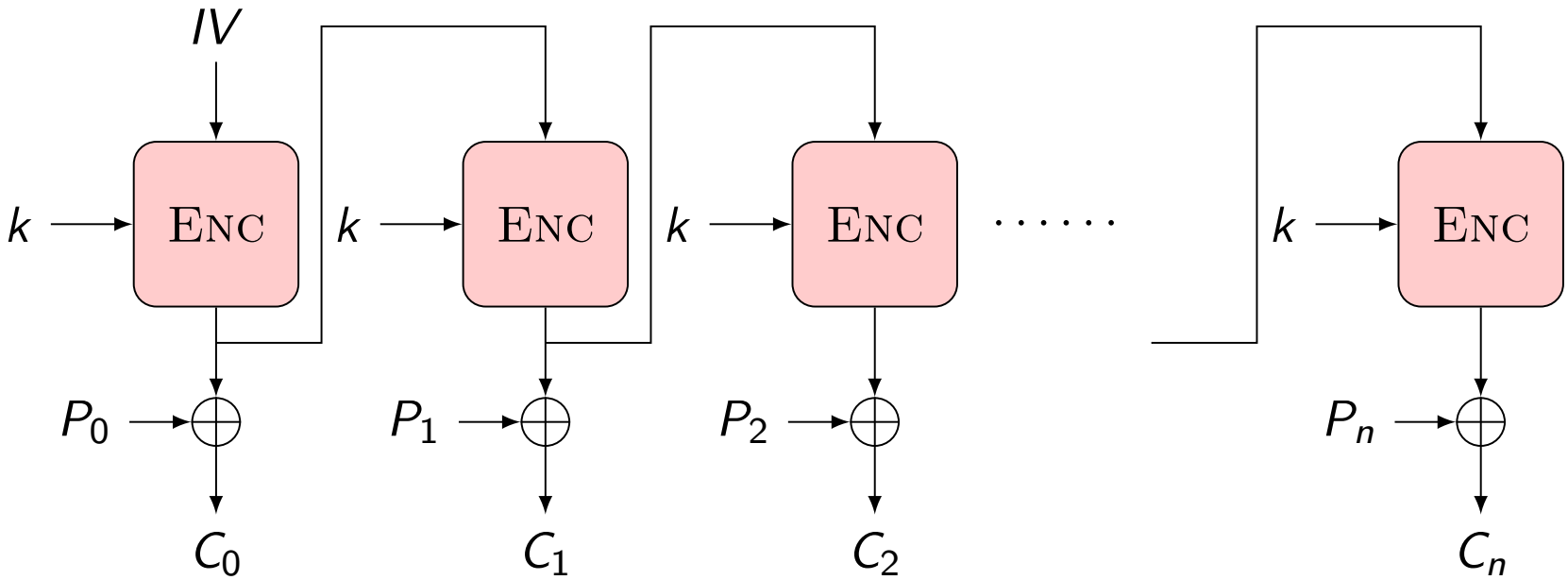
$$C_i = P_i \oplus O_i$$

$$P_i = C_i \oplus O_i$$

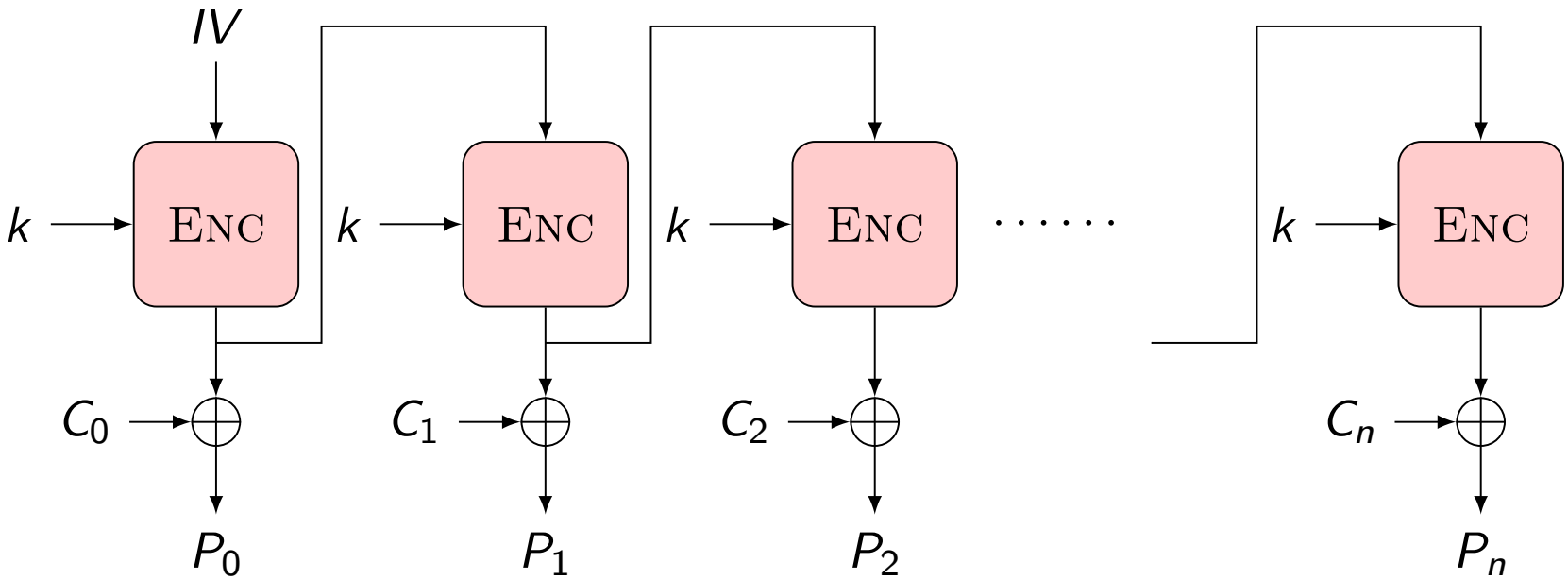
$$O_i = E_K(O_{i-1})$$

$$O_0 = \text{IV}$$

OFB encryption



OFB Decryption



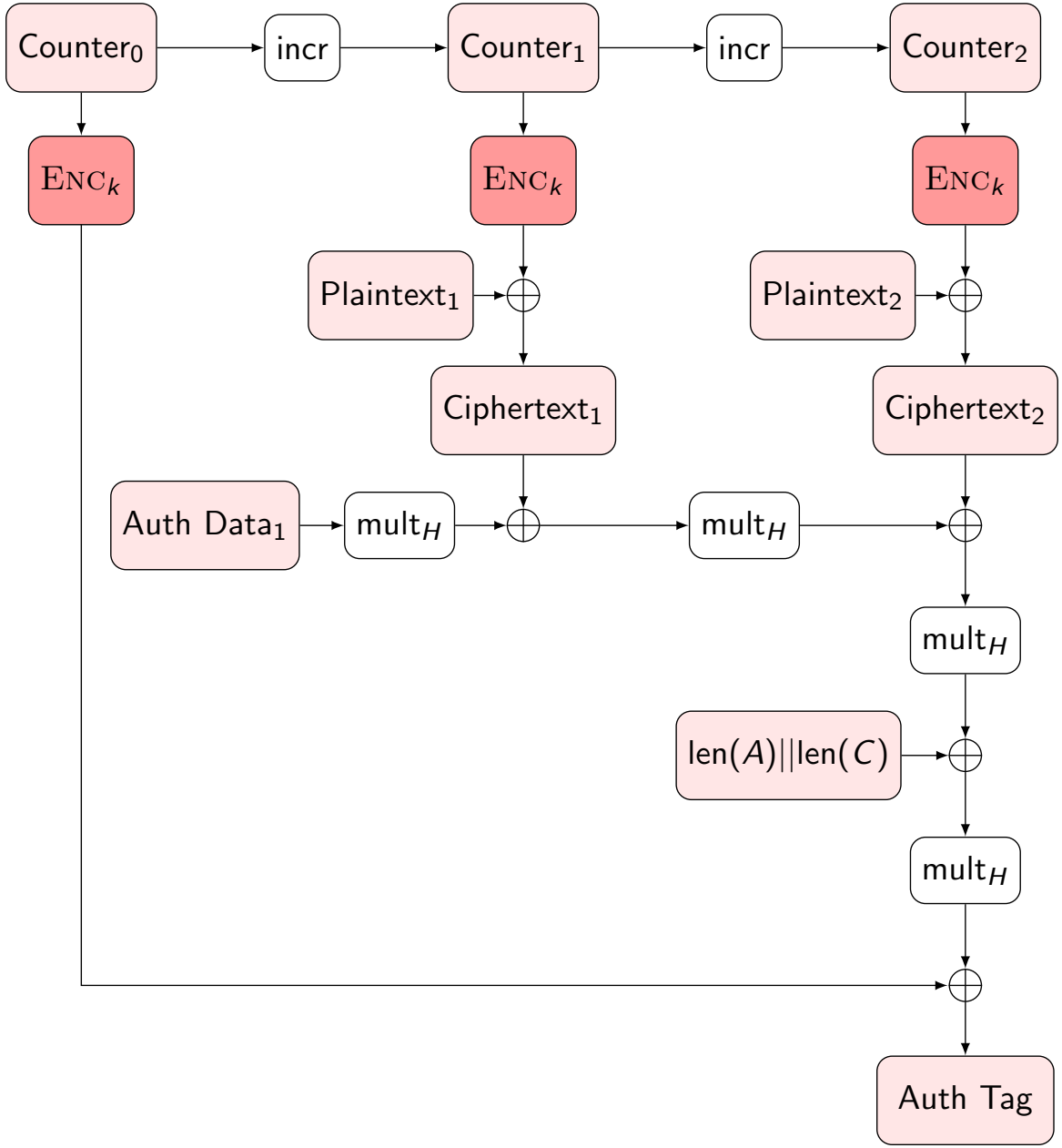
Counter Mode (CTR)

$$C_0 = IV$$

$$C_i = P_i \oplus E_k(IV + i - 1)$$

$$P_i = C_i \oplus E_k(IV + i - 1)$$

GCM Galois/Counter Mode by D. McGrew and J. Viega



GCM

$GF(2^{128})$ est défini par $x^{128} + x^7 + x^2 + x + 1$

$$S_i = \begin{cases} A_i & \text{for } i = 1, \ldots, m-1 \\ A_m^* \parallel 0^{128-v} & \text{for } i = m \\ C_{i-m} & \text{for } i = m+1, \ldots, m+n-1 \\ C_n^* \parallel 0^{128-u} & \text{for } i = m+n \\ \text{len}(A) \parallel \text{len}(C) & \text{for } i = m+n+1 \end{cases}$$

where $len(A)$ and $len(C)$ are the 64-bit representations of the bit lengths of A and C, respectively, $v = len(A) \mod 128$ is the bit length of the final block of A, $u = len(C) \mod 128$ is the bit length of the final block of C.

$$X_i = \sum_{j=1}^{i} S_j \cdot H^{i-j+1} = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus S_i) \cdot H & \text{for } i = 1, \ldots, m+n+1 \end{cases}$$

Outline

"Classifications" of Hash Functions

Unkeyed Hash function

- Modification Code Detection (MDC)
- Data integrity
- Fingerprints of messages
- Other applications

Keyed Hash function

- Message Authentication Code (MAC)
- Password Verification in uncrypted password-image files.
- Key confirmation or establishment
- Time-stamping
- Others applications

Hash Functions

A hash function H takes as input a bit-string of any finite length and returns a corresponding 'digest' of fixed length.

$$h : \{0,1\}^* \rightarrow \{0,1\}^n$$

$$H(Alice) \quad = \quad$$ 

Definition (Pre-image resistance (One-way) OWHF)

Given an output y, it is computationally infeasible to compute x such that

$$h(x) = y$$

Properties of hash functions

2nd Pre-image resistance (weak-collision resistant) CRHF

Given an input x, it is computationally infeasible to compute x' such that

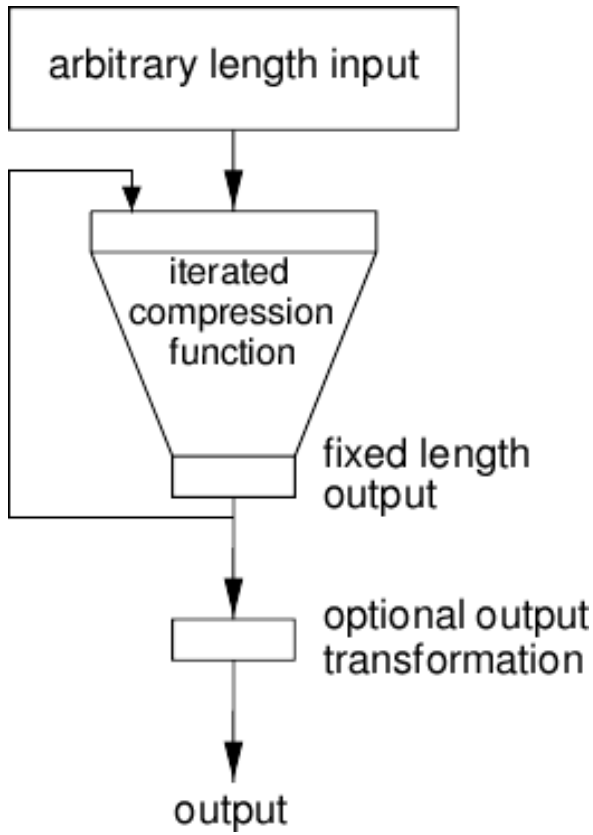
$$h(x') = h(x)$$

Collision resistance (strong-collision resistant)

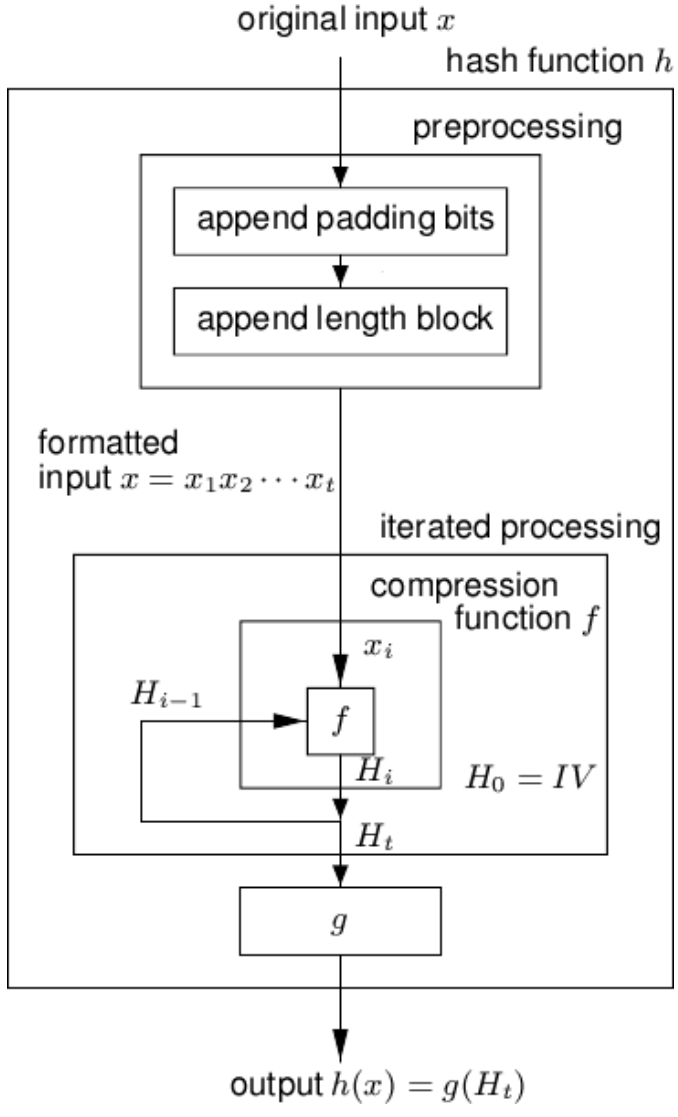
It is computationally infeasible to compute x and x' such that

$$h(x) = h(x')$$

Basic construction of hash functions



Basic construction of hash functions



Basic construction of hash functions (Merkle-Damgård)

$$f : \{0,1\}^m \to \{0,1\}^n$$

1. Break the message x to hash in blocks of size $m - n$:

$$x = x_1 x_2 \ldots x_t$$

2. Pad x_t with zeros as necessary.
3. Define x_{t+1} as the binary representation of the bit length of x.
4. Iterate over the blocks:

$$\begin{aligned} H_0 &= 0^n \\ H_i &= f(H_{i-1} || x_i) \\ h(x) &= H_{t+1} \end{aligned}$$

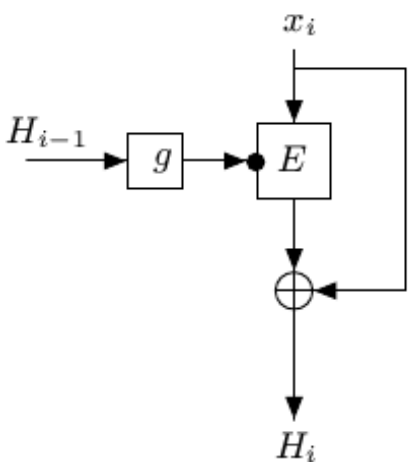
Basic construction of hash functions

Theorem

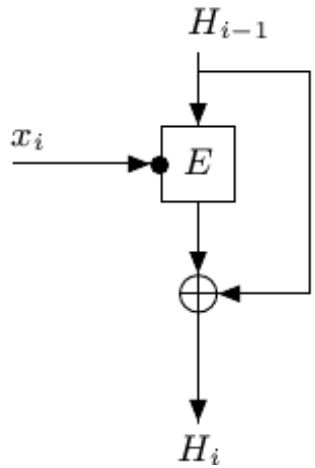
If the compression function f is collision resistant, then the obtained hash function h is collision resistant.

Hash functions based on (MDC) block ciphers

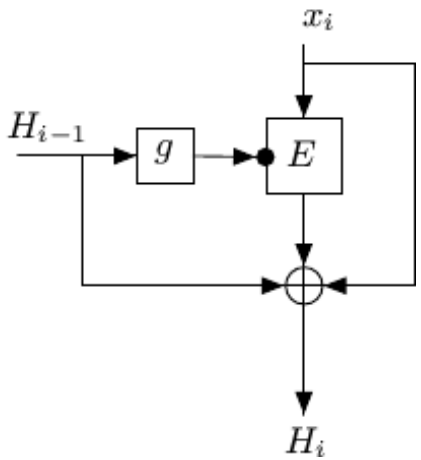
Matyas-Meyer-Oseas



Davies-Meyer

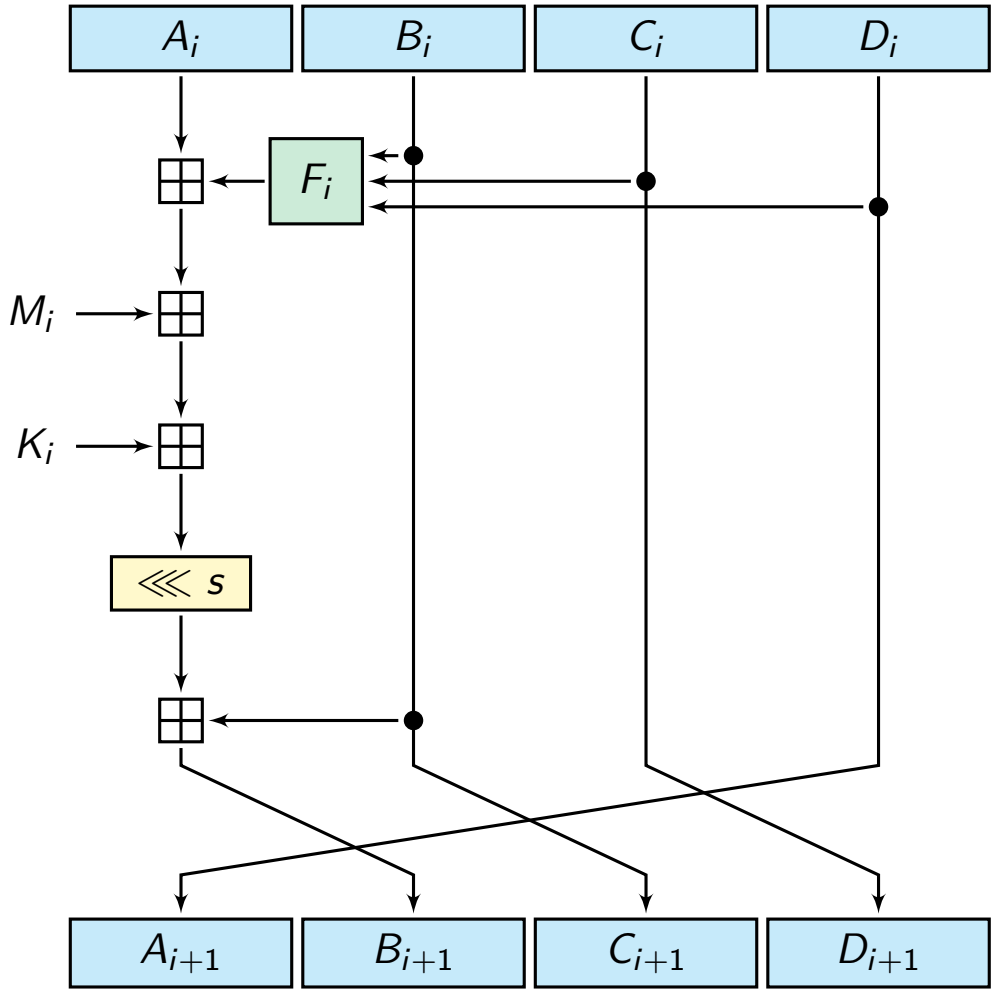


Miyaguchi-Preneel



MD5 by Ron Rivest in 1991

For each 512-bit block of plaintext



Ki denotes a 32-bit constant, different for each operation Addition denotes addition modulo 2^{32}.

MD5 by Ron Rivest in 1991

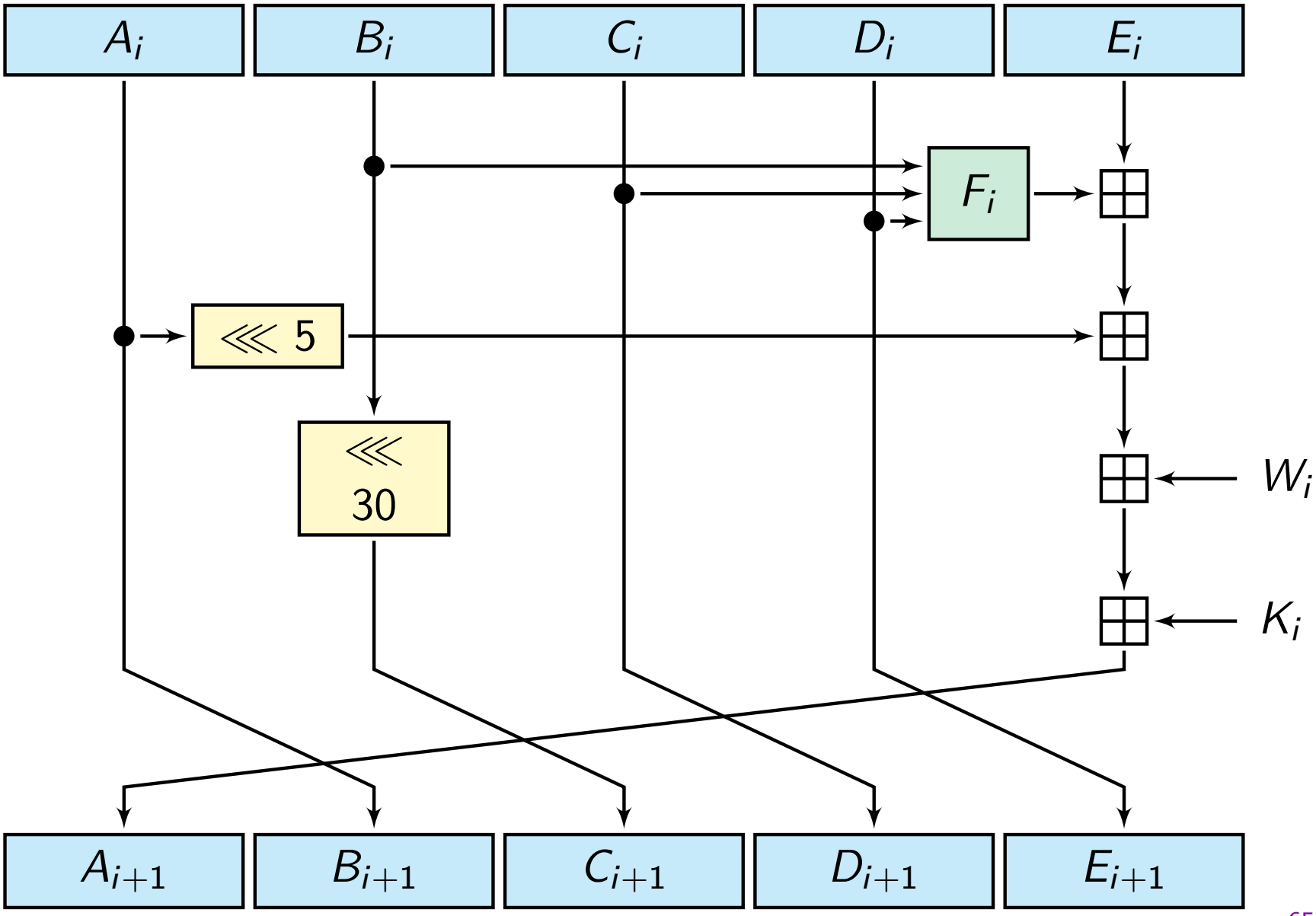
There are 4 possible functions F
A different one is used in each round:

- $F(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$
- $G(B, C, D) = (B \wedge D) \vee (C \wedge \neg D)$
- $H(B, C, D) = B \oplus C \oplus D$
- $I(B, C, D) = C \oplus (B \vee \neg D)$

MD5 Cryptanalysis

- In 1993, Den Boer and Bosselaers gave a "pseudo-collision" two different initialization vectors of compression function which produce an identical digest.
- In 1996, Dobbertin announced a collision of the compression function of MD5.
- 17 August 2004, collisions for the full MD5 by Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu.
- On 1 March 2005, Arjen Lenstra, Xiaoyun Wang, and Benne de Weger demonstrated construction of two X.509 certificates with different public keys and the same MD5 hash value.
- A few days later, Vlastimil Klima able to construct MD5 collisions in a few hours on a single notebook computer.
- On 18 March 2006, Klima published an algorithm that can find a collision within one minute on a single notebook computer, using a method he calls tunneling.
- On 24 December 2010, Tao Xie and Dengguo Feng announced the first published single-block (512 bit) MD5 collision.

SHA-1



Collision for PDF

SHA1(TP/SHA1/a.pdf)=
a5a678701d8b2ab07c96d101b3331fb4992f0980
SHA1(TP/SHA1/b.pdf)=
a5a678701d8b2ab07c96d101b3331fb4992f0980

List of Hash Functions

Algorithm	Output size	Internal state size	Block size	Length size	Word size	Collision
HAVAL	256/.../128	256	1024	64	32	Yes
MD2	128	384	128	No	8	Almost
MD4	128	128	512	64	32	Yes
MD5	128	128	512	64	32	Yes
PANAMA	256	8736	256	No	32	Yes
RadioGatún	Arbitrarily long	58 words	3 words	No	1-64	No
RIPEMD	128	128	512	64	32	Yes
RIPEMD	128/256	128/256	512	64	32	No
RIPEMD	160/320	160/320	512	64	32	No
SHA-0	160	160	512	64	32	Yes
SHA-1	160	160	512	64	32	With flaws
SHA-256/224	256/224	256	512	64	32	No
SHA-512/384	512/384	512	1024	128	64	No
Tiger(2)	192/160/128	192	512	64	64	No
WHIRLPOOL	512	512	512	256	8	No

SHA-3 Zoo

64 Submissions, 54 selected,

1. * BLAKE Jean-Philippe Aumasson
2. Blue Midnight Wish Svein Johan Knapskog
3. CubeHash Daniel J. Bernstein preimage
4. ECHO Henri Gilbert
5. Fugue Charanjit S. Jutla
6. * Grøstl Lars R. Knudsen
7. Hamsi Özgül Küçk
8. * JH Hongjun Wu preimage
9. * Keccak The Keccak Team
10. Luffa Dai Watanabe
11. Shabal Jean-François Misarsky
12. SHAvite-3 Orr Dunkelman
13. SIMD Gaëtan Leurent
14. * Skein Bruce Schneier

SHA-3 = Keccak (sponge + compression)

Authors

- Guido Bertoni (Italy) of STMicroelectronics,
- Joan Daemen (Belgium) of STMicroelectronics,
- Michaël Peeters (Belgium) of NXP Semiconductors, and
- Gilles Van Assche (Belgium) of STMicroelectronics.

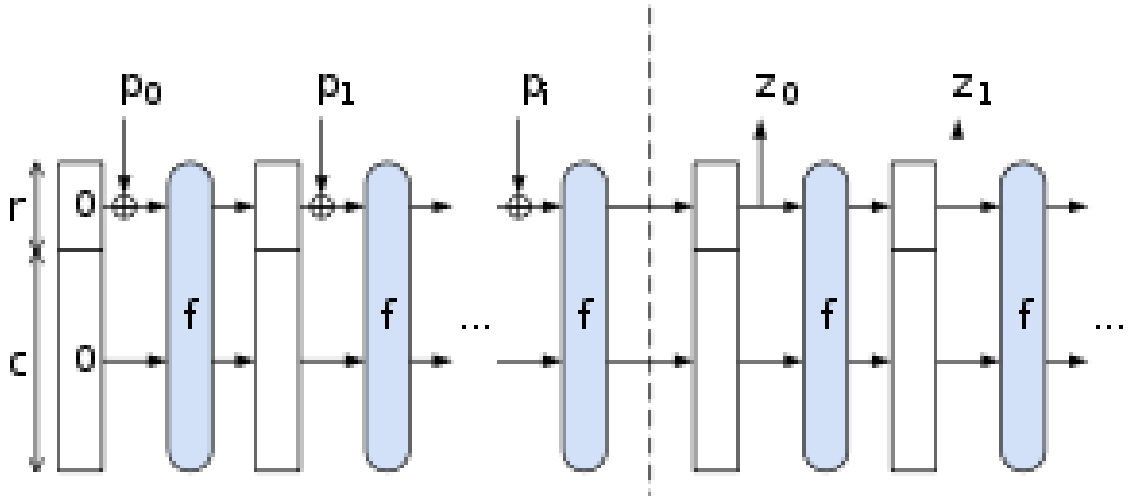
SHA-3 = Keccak

$$h : \{1,0\}^* \to \{1,0\}^n$$

- MD5: $n = 128$ (Ron Rivest, 1992)
- SHA-1: $n = 160$ (NSA, NIST, 1995)
- SHA-2: $n \in \{224, 256, 384, 512\}$ (NSA, NIST, 2001)
- SHA-3: n is arbitrary (NSA, NIST, 2012)

SHA-3 = Keccak is a sponge based hash

$$H(P_0|P_1|\ldots|P_i) = Z_0|Z_1|\ldots|Z_l$$



$b = r + c$

- r bits of rate
- c bits of capacity (security parameter)

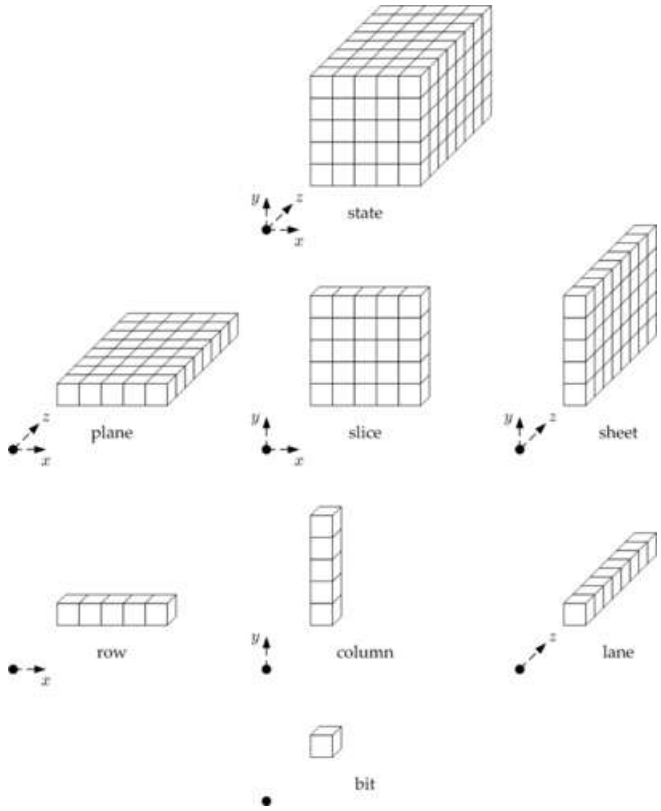
Inside Keccak

- 7 permutations: $b \in \{25, 50, 100, 200, 400, 800, 1600\}$
- ... from toy over lightweight to high-speed ...
- SHA-3 instance: r = 1088 and c = 512
  - permutation width: 1600
  - security strength 256: post-quantum sufficient
- Lightweight instance: r = 40 and c = 160
  - permutation width: 200
  - security strength 80: same as (initially expected from) SHA-1

SHA-3 = Keccak f Setting

Defined for word of size, $w = 2^l$ bits (if $l = 6$ 64-bit words)

State is $5 \times 5 \times w$ array of bits (a[i][j][k])



- state = 5×5 lanes, each containing 2^l bits
- (5×5)-bit slices, 2^l of them

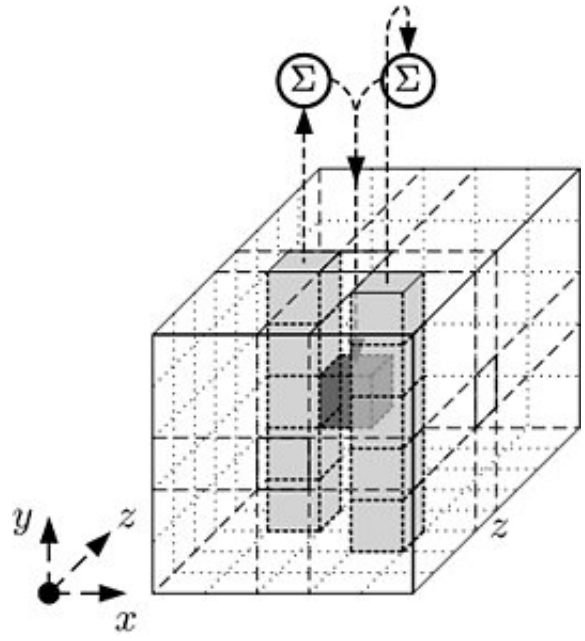
SHA-3 = Keccak

The basic block permutation function consists of $12 + 2 \times l$ iterations of following sub-rounds.

1. step Θ
2. step ρ
3. step π
4. step χ
5. step ι

Keccak Θ

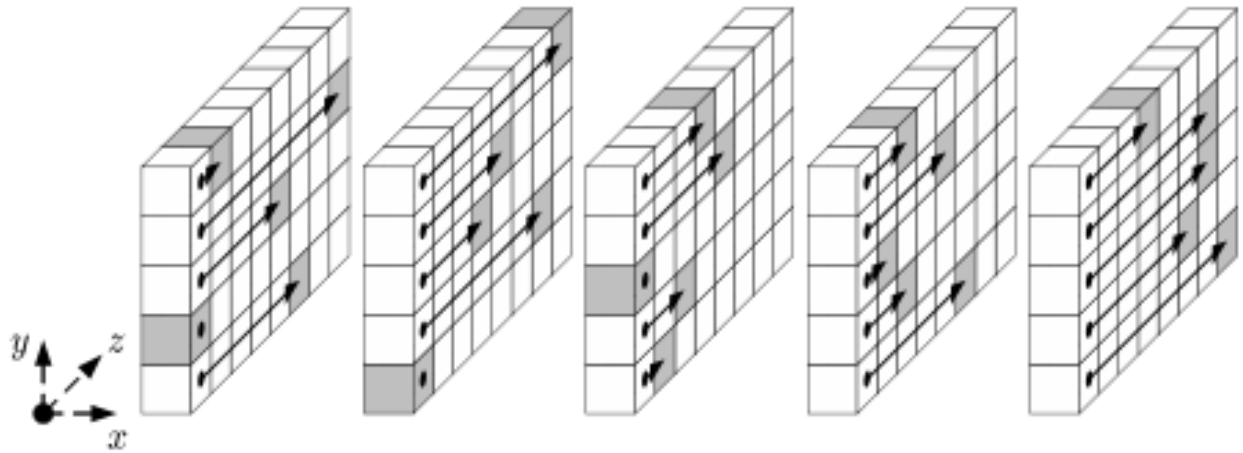
1. Compute the parity of each of the 5-bit columns
2. $\oplus$ the sum of a[x-1][][z] and of a[x+1][][z-1] into a[x][y][z].



$$a[i][j][k] \oplus = parity(a[0..4][j-1][k]) \oplus parity(a[0..4][j+1][k-1])$$

Keccak ρ

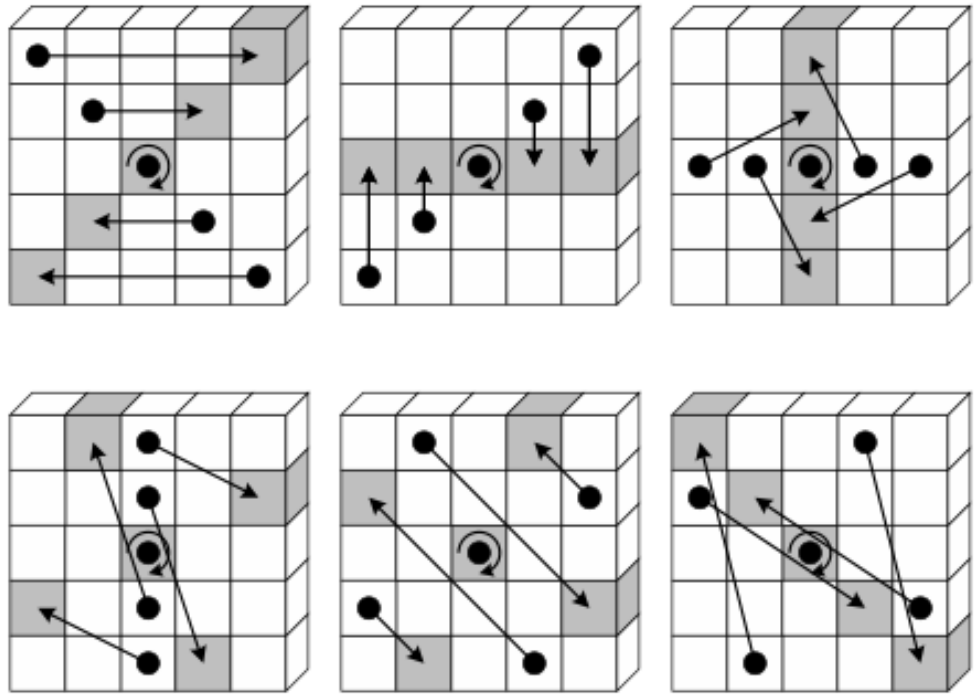
Bitwise rotate each of the 25 words by a different rotation.



$a[0][0]$ is not rotated, and for all $0 \leq t < 24$
$a[i][j][k] = a[i][j][k - (t+1)(t+2)/2]$, where
$\begin{pmatrix} i \\ j \end{pmatrix} = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}^t \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Keccak π

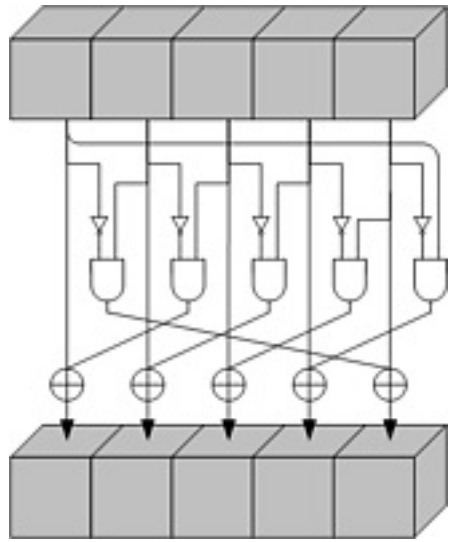
Permute the 25 words in a fixed pattern.



$$a[i][j] = a[j][2i + 3j]$$

Keccak χ

Bitwise combine along rows, using $a = a \oplus (\neg b \& c)$.



$$a[i][j][k] \oplus = \neg a[i][j+1][k] \& a[i][j+2][k]$$

This is the only non-linear operation in SHA-3.

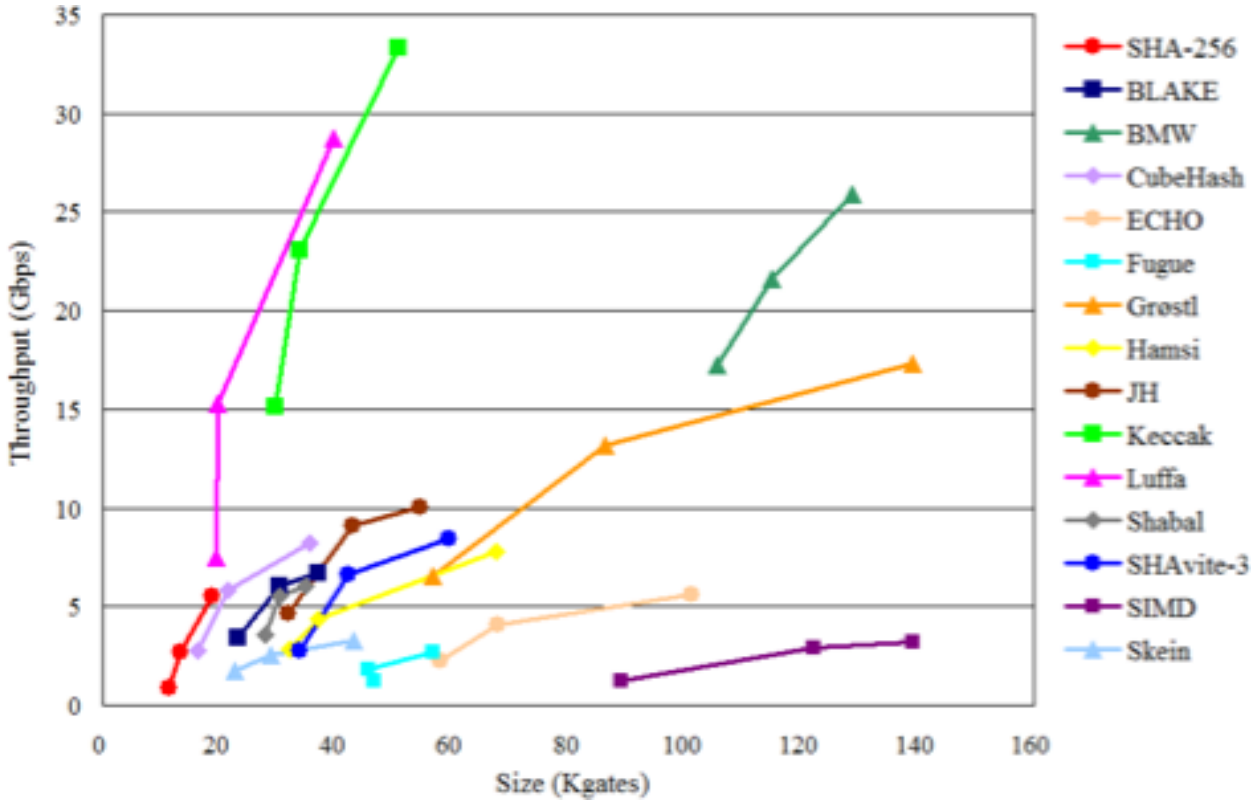
Keccak ι

Exclusive-or a round constant into one word of the state.

- In round n, for $0 \leq m \leq l$, $a[0][0][2m-1]$ is exclusive-ORed with bit $m+7n$ of a degree-8 LFSR (Linear Feedback Shift Register) sequence.

This breaks the symmetry that is preserved by the other sub-rounds.

Why Keccak



Kangouroo 12, a fast version of Keccak

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try $1 : 1 - 0$

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$
- $\vdots$
- Try q : $1 - (q-1)/2^k$

Birthday Paradox : Hash Function

Let an Hash function $H : D \rightarrow 2^k$

Naïve Collision

With $2^k + 1$ try there is a collision

$$P(at\ least\ 1\ collision) = 1 - P(no\ collision)$$

Probability of no collision

- Try 1 : $1 - 0$
- Try 2 : $1 - 1/2^k$
- Try 3 : $1 - 2/2^k$
- $\vdots$
- Try q : $1 - (q-1)/2^k$

$$P(no\ collision) = \prod_{i=1}^{i=q}(1 - i/2^k)$$

Birthday Paradox : Hash Function

$P(\textit{at least 1 collision}) = 1 - P(\textit{no collision})$
Using $1 - x \approx e^{-x}$ we have

$$1 - e^{-\sum_{i=1}^{i=q}(1-i/2^k)} = 1 - e^{-q(q-1)/2^{k+1}}$$

If you want a probability of ϵ to have a collision

Need ot solve

$$\epsilon = 1 - e^{-q(q-1)/2^{k+1}}$$

$$q(q-1) = 2^{k+1} ln(1/(1-\epsilon)$$

$$k \approx \sqrt{2^{k+1} ln(1/(1-\epsilon)}$$

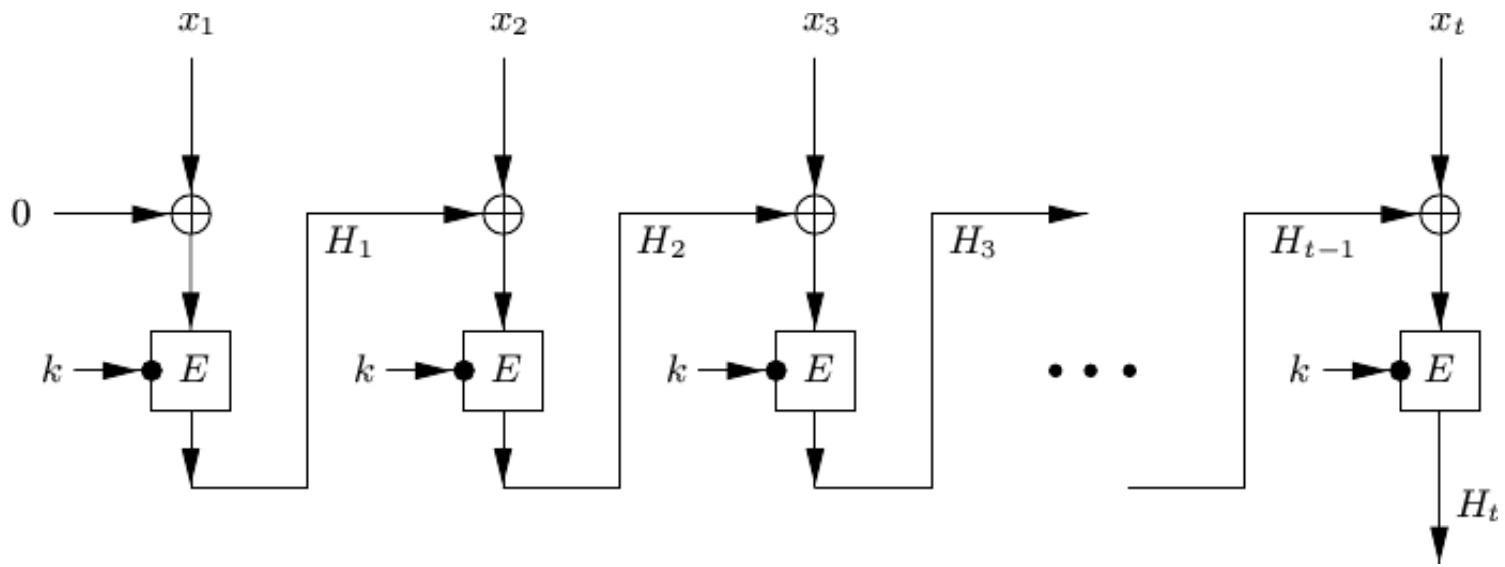
Examples

- $\epsilon = \frac{1}{2} \Rightarrow k \approx 1.177\sqrt{2^{k+1}}$
- $\epsilon = \frac{3}{4} \Rightarrow k \approx 1.665\sqrt{2^{k+1}}$
- $\epsilon = 0.9 \Rightarrow k \approx 2.146\sqrt{2^{k+1}}$

Remark: if 2^{k+1} is 365 among $1.77\sqrt{365} \approx 23$
So should be at least > 64 or even 80. > 128 or 160 to resist birthday attack.

Outline

MAC based on block ciphers



DMAC (CBC-MAC variant)

Example

$c_1 := m_1;$
for $i = 2$ to n do:
$z_i := c_{i-1} \oplus m_i$
$c_i := \mathsf{E}(z_i);$
$tag := \mathsf{E}'(c_n);$

HMAC

Example

$z_1 := k \| m_1;$
$c_1 := \mathcal{H}(z_1);$
for $i = 2$ to n do:;
$\quad z_i := c_{i-1} \| m_i$
$\quad c_i := \mathcal{H}(z_i)$
$z' := k' || c_n;$
$tag := \mathcal{H}(z');$

Outline

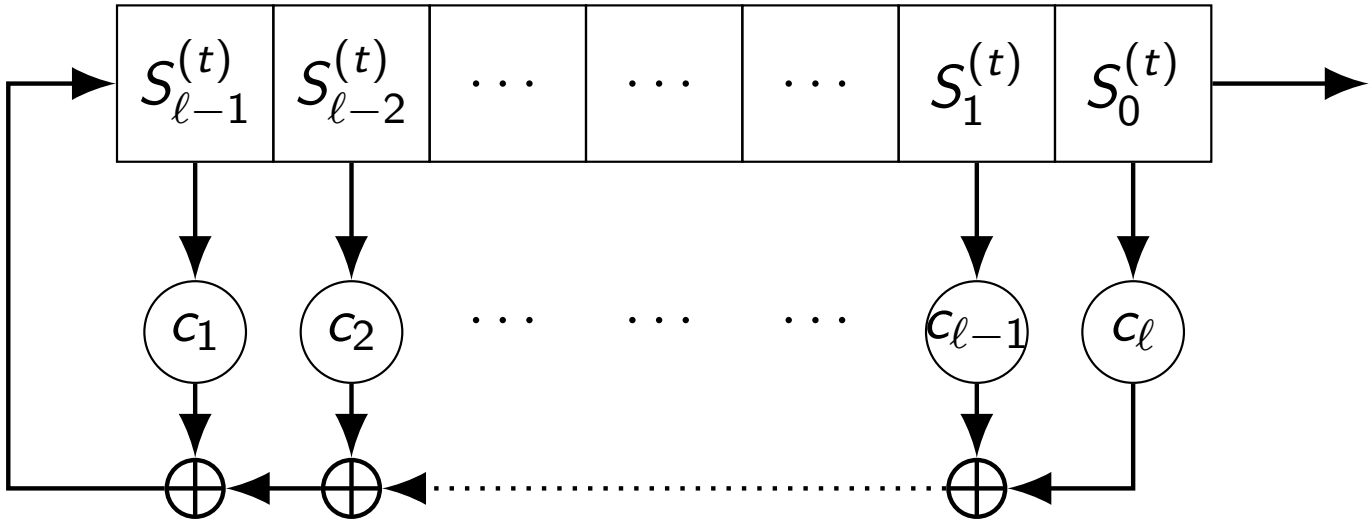
Symetric Encryption

Two kinds of symetric encryption:

- block cipher (fixed plaintext size) DES AES
- stream cipher (unlimited plaintext size) RC4, E0, Crypto-1

To encrypt and to decrypt the same secrete key K is used !

Linear Feedback Shift Register



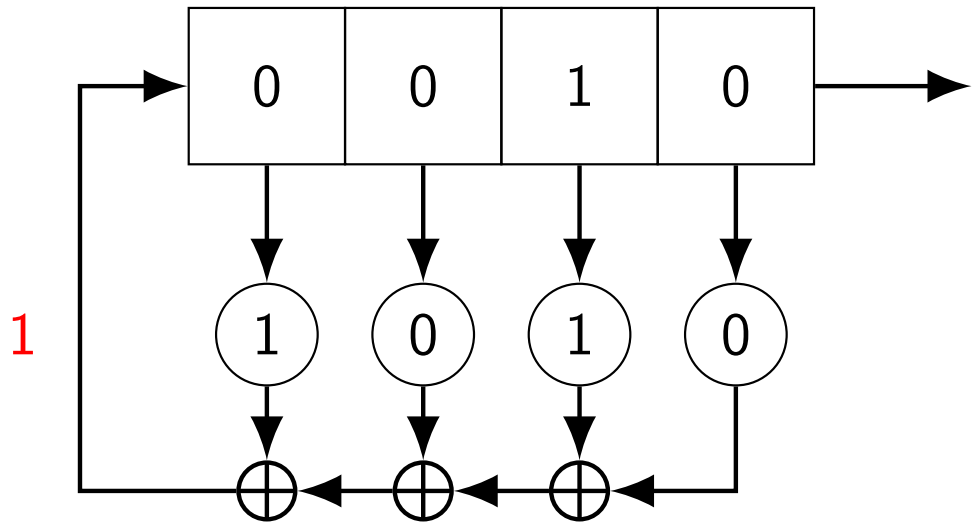
- Length of the register is ℓ, $s^{(0)}$ is the seed
- $\forall c_i \in \{0, 1\}$

$$\forall t \geq 0, s_{\ell-1}^{(t+1)} = \sum_{i=1}^{\ell} c_i s_{\ell-i}^{(t)}$$

$$Shift : s_i^{(t+1)} = s_{i+1}^{(t)}, \forall i, 0 \leq i \leq \ell - 2$$

Example

Seed $s^{(0)} = 0010$ and $c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$

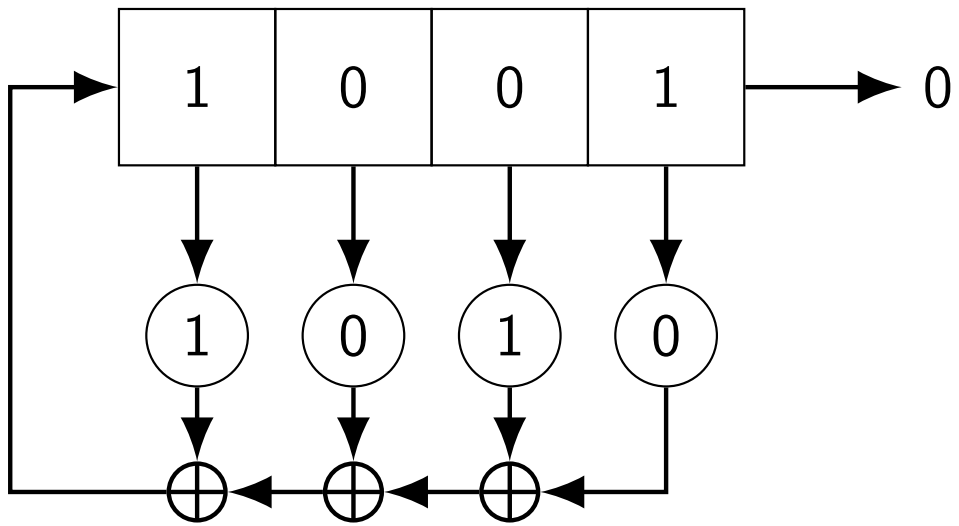


$$
\begin{aligned}
s_3^{(1)} &= (s_3^{(0)} \cdot c_1) \oplus (s_2^{(0)} \cdot c_2) \oplus (s_1^{(0)} \cdot c_3) \oplus (s_0^{(0)} \cdot c_4) \\
&= (0 \cdot 1) \oplus (0 \cdot 0) \oplus (1 \cdot 1) \oplus (0 \cdot 0) \\
&= 1
\end{aligned}
$$

Example first output bit

$c_1 = 1$ $c_2 = 0$ $c_3 = 1$ and $c_4 = 0$

$$s_2^{(1)} = s_3^{(0)}, s_1^{(1)} = s_2^{(0)}, \text{ and } s_0^{(1)} = s_1^{(0)}$$



Definitions

Period

A serie $(s_n)_{n\in\mathbb{N}}$ is periodic of perido p if $s_{n+p} = s + n, \forall n$.

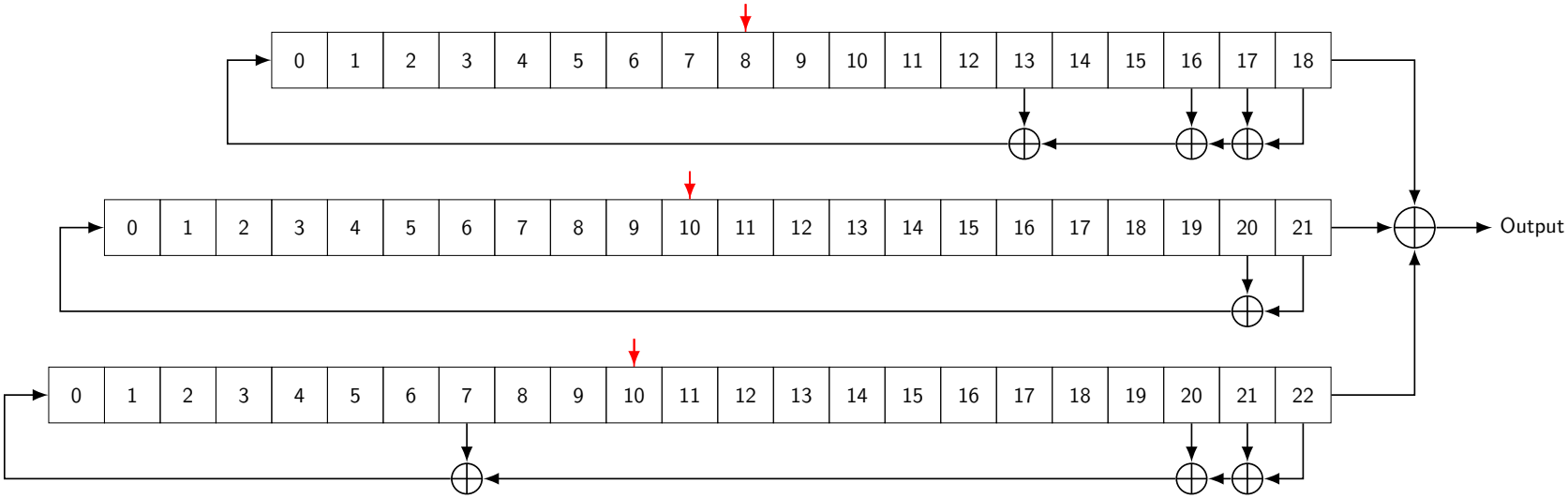
Retroaction polynomial

$p(X) \in \mathbb{F}_2[X]$:

$$p(X) = 1 + \sum_{i=1}^{\ell} c_i X^i$$

A5/1 used for GSM in Europe 1994

Red bits are used to determine the majority amont 3 values.
Winner registers are shift.



$$x^{19} + x^{18} + x^{17} + x^{14} + 1$$

$$x^{22} + x^{21} + 1$$

$$x^{23} + x^{22} + x^{21} + x^{8} + 1$$

Attack on A5/1

- 1997, Golic attack in $2^{40.16}$
- 2000, Alex Biryukov, Adi Shamir and David Wagner : few minutes with 2 minutes of plain communication (using in total 300 Go data, in 2^{48} steps).
- 2000 Eli Biham et Orr Dunkelman attack in $2^{39.91}$ with $2^{20.8}$ bits fo data.
- Improvement by Maximov et al for one minute of computation and few clear secands of plain communication.
  Maximov, Alexander; Thomas Johansson; Steve Babbage (2004). "An Improved Correlation Attack on A5/1". Selected Areas in Cryptography 2004: 1–18.
  Barkan, Elad; Eli Biham (2005). "Conditional Estimators: An Effective Attack on A5/1". Selected Areas in Cryptography 2005: 1–19.
- 13 December 2013, with Snowden affirmations, NSA can listen GSM communications

RC4 by Ron Rivest in 1987

"Rivest Cipher 4" or "Ron's Code" is a stream cipher used in TLS (Transport Layer Security) and WEP (Wired Equivalent Privacy).

- The key-scheduling algorithm (KSA)
- The pseudo-random generation algorithm (PRGA)

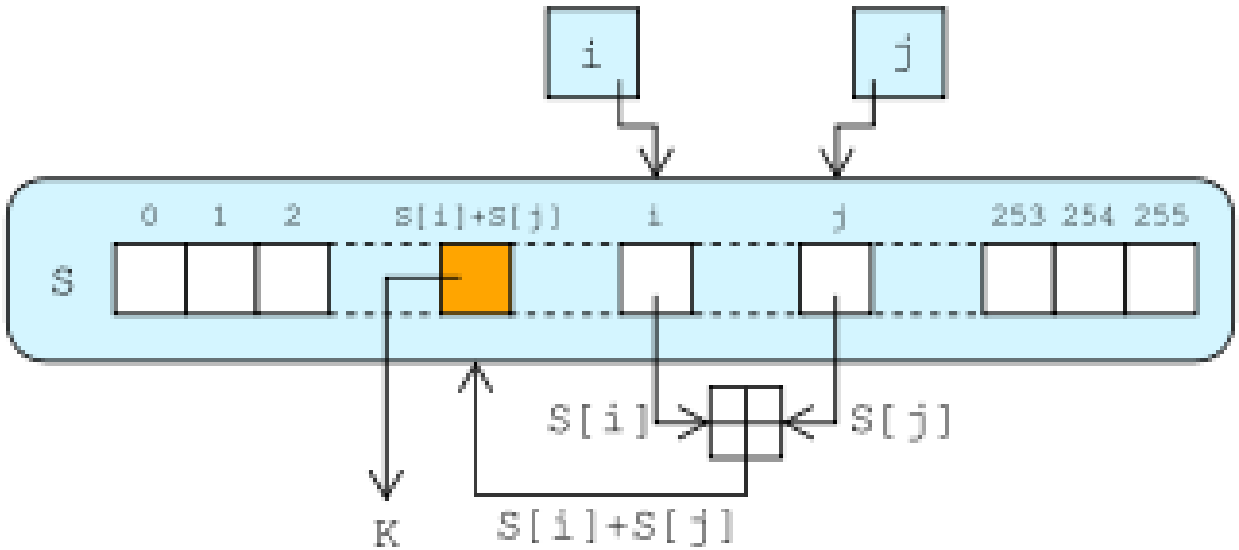
KSA use a key of length between 40 – 128 bits

- Array "S" is initialized to the identity permutation.
- 256 iterations with mixes of bytes of the key at the same time.

```
j := 0
for i from 0 to 255
    j := (j + S[i] + key[i mod keylength]) mod 256
    swap values of S[i] and S[j]
endfor
```

Pseudo-Random Generation Algorithm (PRGA)

```
i := 0; j := 0;
while GeneratingOutput:
    i := (i + 1) mod 256
    j := (j + S[i]) mod 256
    swap values of S[i] and S[j]
    K := S[(S[i] + S[j]) mod 256]
    output K
```



Recent attacks on RC4

- Fluhrer, Mantin and Shamir attack 2001
- Klein's attack 2005
- John Leyden (2013-09-06). "That earth-shattering NSA crypto-cracking: Have spooks smashed RC4?"
- "Fresh revelations from whistleblower Edward Snowden suggest that the NSA can crack TLS/SSL connections, the widespread technology securing HTTPS websites and virtual private networks (VPNs)."
- " Attack relies on statistical flaws in the keystream generated by the RC4 algorithm. It relies on getting a victim to open a web page containing malicious JavaScript code that repeatedly tries to log into Google's Gmail, for example. This allows an attacker to get hold of a bulk of traffic needed to perform cryptanalysis."

  Nadhem AlFardan, Dan Bernstein, Kenny Paterson, Bertram Poettering and Jacob Schuldt. "On the Security of RC4 in TLS". Royal Holloway University of London. Retrieved March 13, 2013.

RC4 bad

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         S[i] ^= S[j];
         S[j] ^= S[i];
         S[i] ^= S[j];
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          S[i] ^= S[j];
          S[j] ^= S[i];
          S[i] ^= S[j];
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

RC4 Good

```
int main (int argc , char * argv []) {
    unsigned char S [256] , c;
    unsigned char key [] = KEY;
    int klen = strlen ( key );
    int i,j,k;

    /* Init S[] */
    for (i =0; i <256; i++)
        S[i] = i;

    /* Scramble S[] with the key */
    j = 0;
    for (i =0; i <256; i++) {
         j = (j+S[i]+ key [i% klen ]) % 256;
         k = S[i];
         S[i] = S[j];
         S[j] = k;
     }
     /* Generate the keystream and cipher the input stream */
     i = j = 0;
     while ( read (0, &c, 1) > 0) {
          i = (i +1) % 256;
          j = (j+S[i]) % 256;
          k = S[i];
          S[i] = S[j];
          S[j] = k;
          c ^= S[(S[i]+S[j]) % 256];
          write (1, &c, 1);
     }}
```

Swap

Classical way (using temporary variable)

```
tmp = a
a = b
b = tmp
```

Without but with + or XOR

```
a = a+b
b = a-b
a = a-b

a = a^b
b = a^b
a = a^b
```

Swap

The buggy adaptation

```
S[i] = S[i]^S[j]
S[j] = S[i]^S[j]
S[i] = S[i]^S[j]
```

because when $i = j$, we have

```
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
S[i] = S[i]^S[i]
```

- instead of exchanging a value with itself, we set it to 0
- the RC4 state fills up with 0
- the bitstream quickly degrades to a sequence of 0
- encryption does not happen anymore

Outline

Linear Cryptanalysis

Eurocrypt '93, Mitsuru Matsui, Linear Cryptonalysis Method for DES Cipher
S-Box are non linear by construction

$$X_i \rightarrow Y_i$$

Goal find such relations in order to approximate S-Boxes:

$$X_1 \oplus X_2 \oplus X_3 \oplus \ldots \oplus X_N = Y_1 \oplus Y_2 \oplus Y_3 \oplus \ldots \oplus Y_N$$

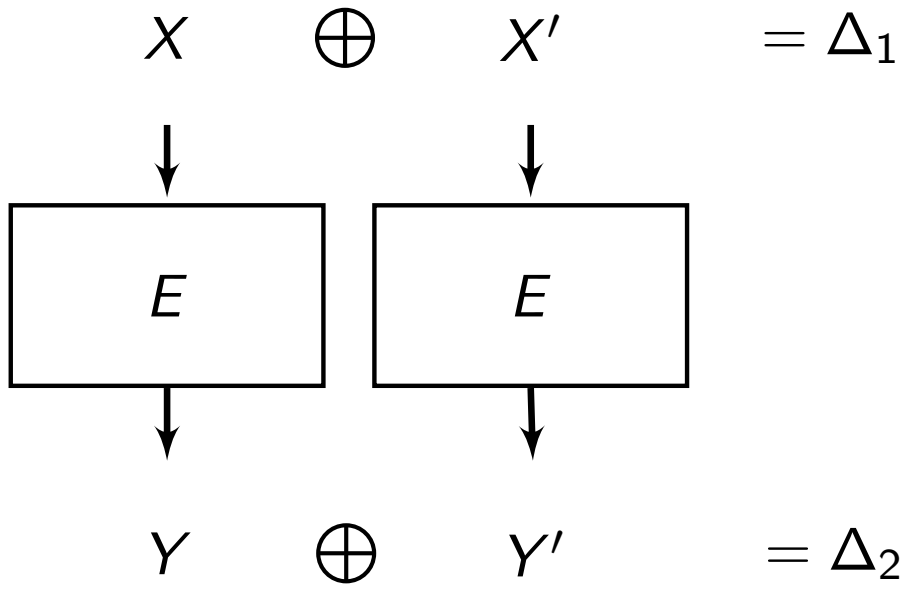
with a high probability to happen.

Piling-up Lemma (Matsui)

For n independent, random binary variables $X_1, \ldots, X_n$ of probability ϵ_i.

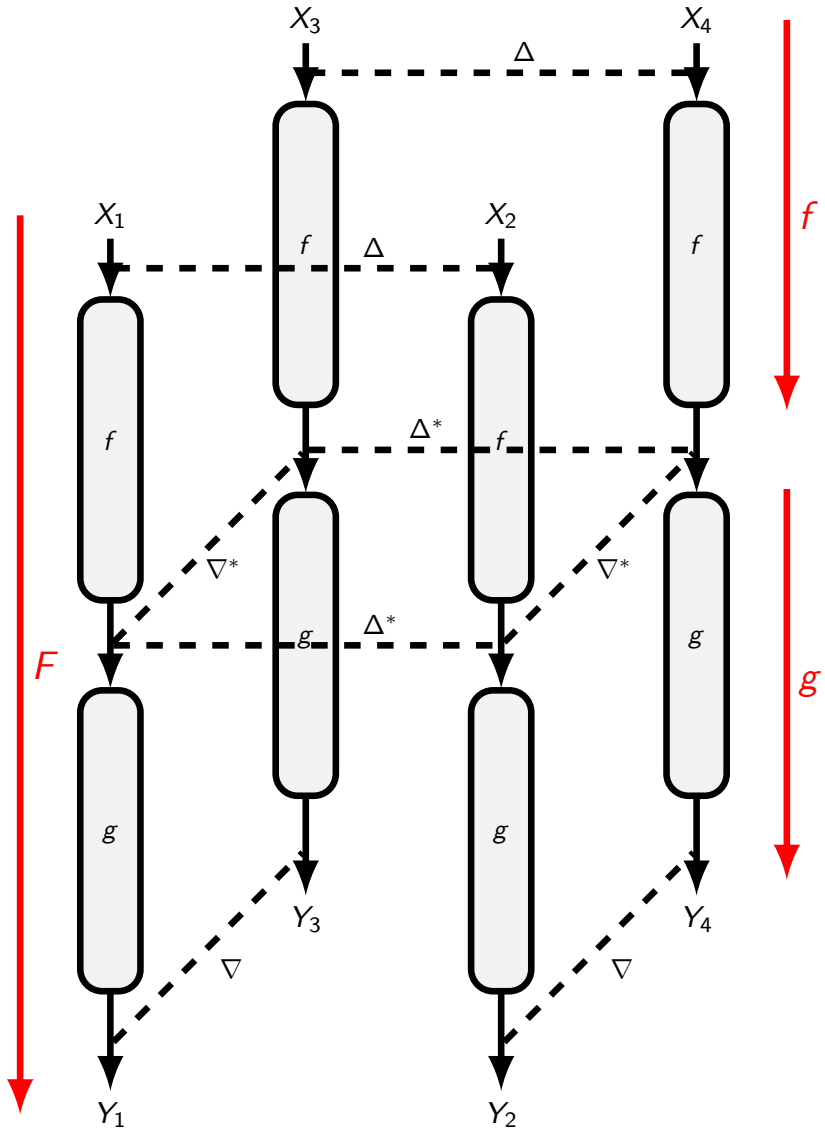
$$Pr[X_1 \oplus \ldots \oplus X_n = 0] = \frac{1}{2} + 2^{n-1} \prod_{i=1}^{n} \epsilon_i$$

Differential Cryptanalysis



1. Look for X and X' such that $X \oplus X' = \Delta_1$
2. Compute $Y = E(X)$ and $Y' = E(X')$
3. Guess the last piece of the key and for all values such that $Y \oplus Y' = \Delta_2$ have an high probability then increase a counter.

Boomerang

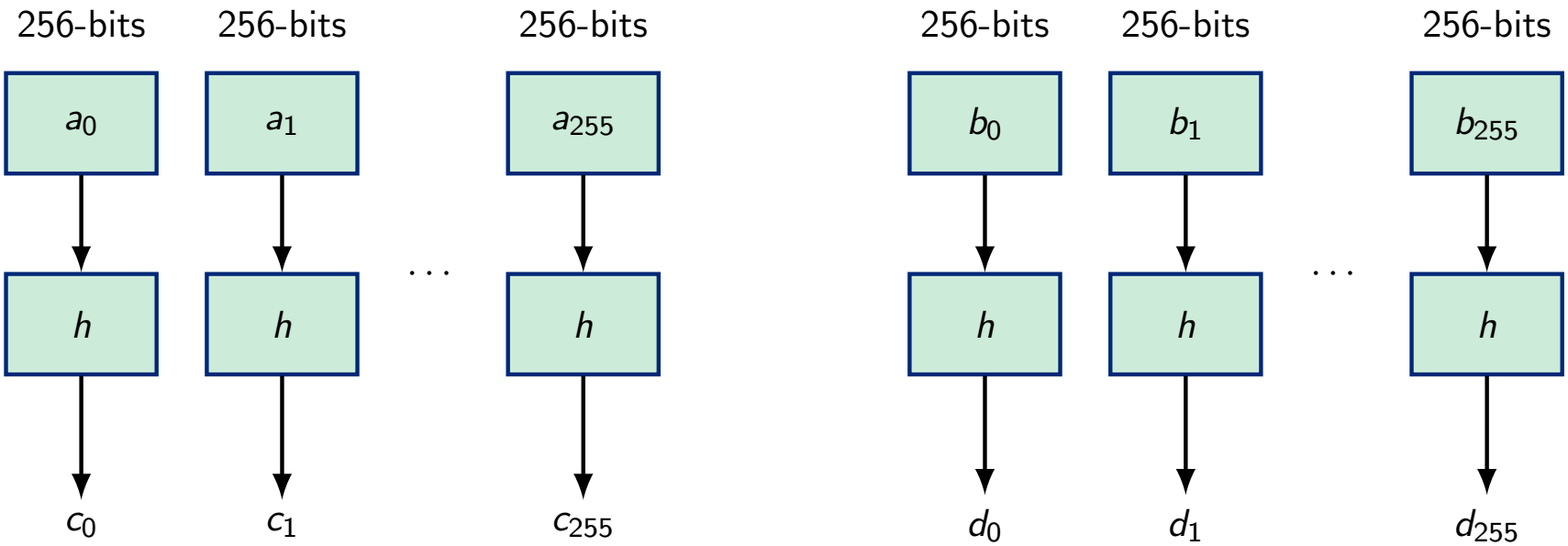

X_3
X_4
Δ
X_1
f
Δ
X_2
f
f
f
Δ^*
f
∇^*
∇^*
F
g
Δ^*
g
g
g
g
Y_3
Y_4
∇
∇
Y_1
Y_2

Outline

Lamport one-time-signature

- A digital signature scheme consists of three algorithms (Gen, Sig, Ver)
- h is a secure hash function.

Gen: private key, $sk = ((a_0, \cdots, a_{255}), (b_0, \cdots, b_{255}))$ for $n = 256$.

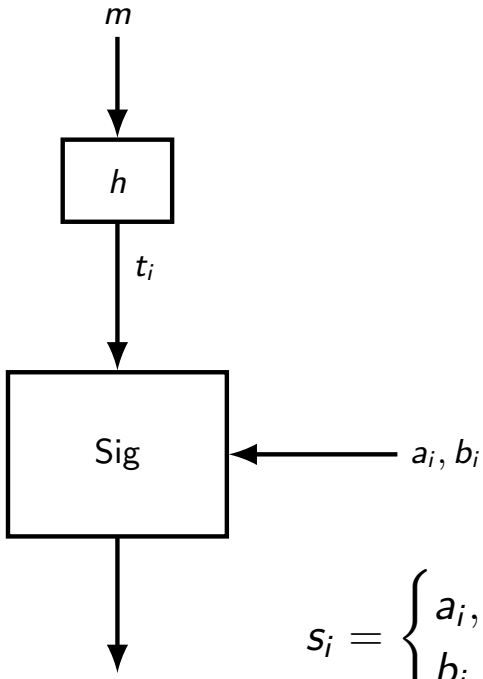


public-key: $pk = ((c_0, \cdots, c_{255}), (d_0, \cdots, d_{255}))$
sk size = pk size $= 2n^2 = 2 \times 256^2$ bits

Lamport one-time signature (cont.)

Sig: signing algorithm

The process to sign message m:



$$s_i = \begin{cases} a_i, & \text{if } t_i = 0 \\ b_i, & \text{otherwise} \end{cases}$$

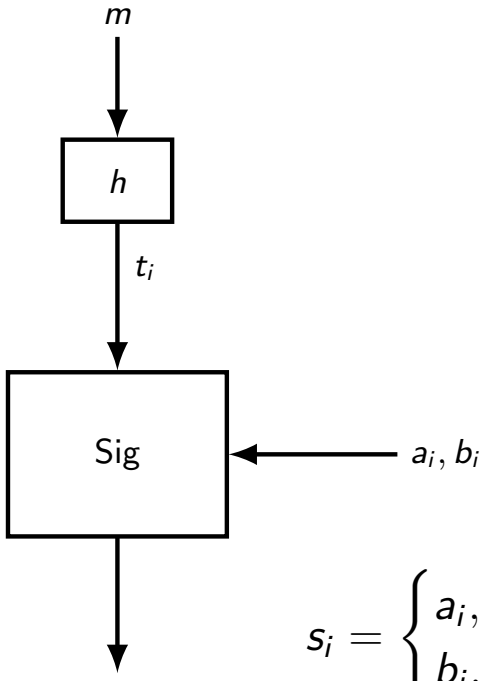
$$sig_{sk}(m) = (s_0, s_1, \cdots, s_{n-1})$$

- Computation: n comparisons, efficient
- The signature size $= n^2 = 256^2$, very large

Lamport one-time signature (cont.)

Sig: signing algorithm

The process to sign message m:

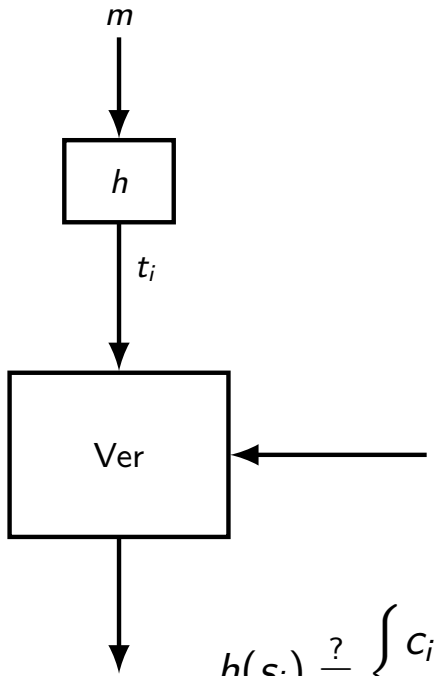


$$s_i = \begin{cases} a_i, & \text{if } t_i = 0 \\ b_i, & \text{otherwise} \end{cases}$$

$$sig_{sk}(m) = (s_0, s_1, \cdots, s_{n-1})$$

- Computation: n comparisons, efficient
- The signature size $= n^2 = 256^2$, very large

Ver: verifying algorithm



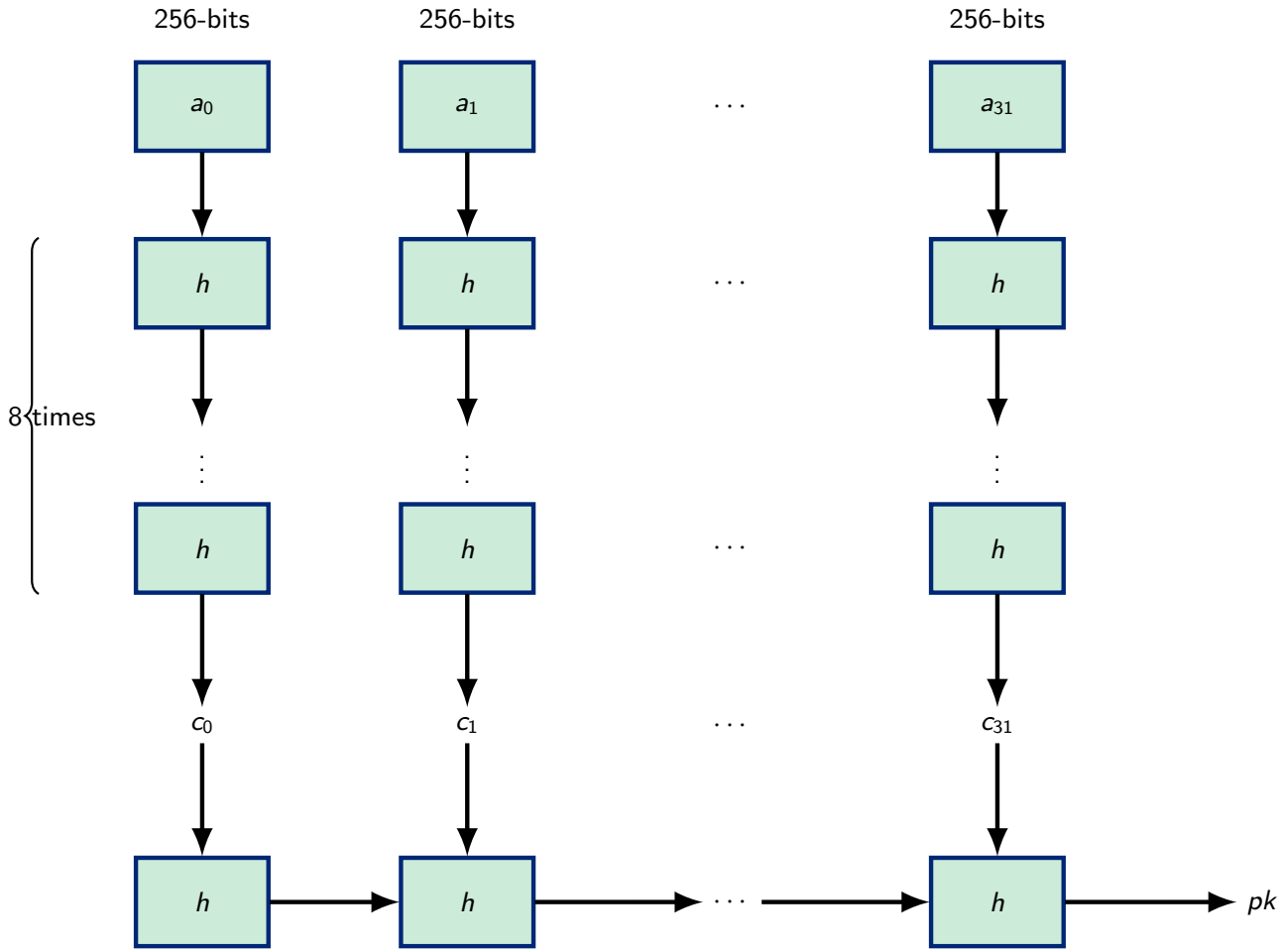
$$h(s_i) \stackrel{?}{=} \begin{cases} c_i, & \text{if } t_i = 0 \\ d_i, & \text{otherwise} \end{cases}$$

Computing complexity: n hash evaluations

Winternitz (variants) one-time signature (OTS)

Goal: Reducing both key size and signature size. Let w be a factor of n, e.g., $n = 256$, $w = 8$.

Gen: (sk, pk)

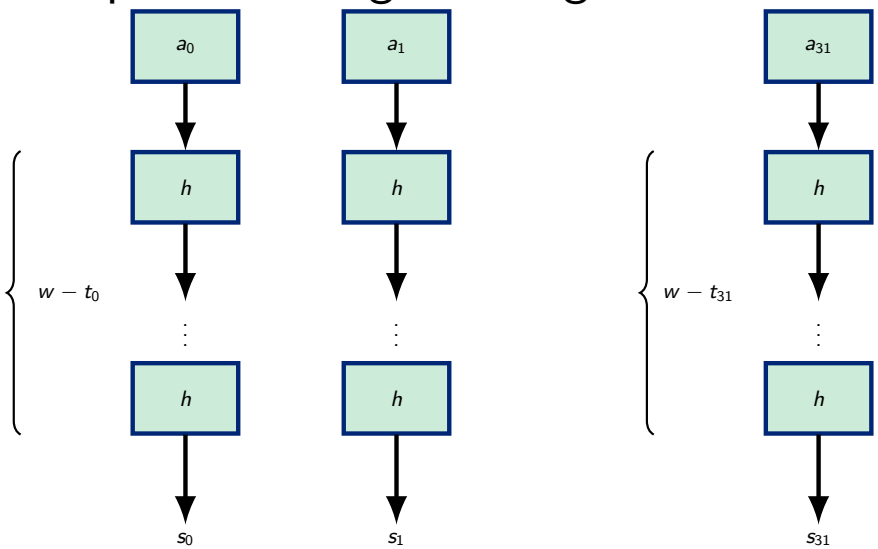


- Computation cost: the same as Lamport OTS, sk size $= n/w \times n = 256/8 \times 256$.
- pk size $= n = 256$ bits, through a hashing chain, shown above, or a

Winternitz (variants) OTS (cont.)

Sig: signing algorithm

The process to sign message m:

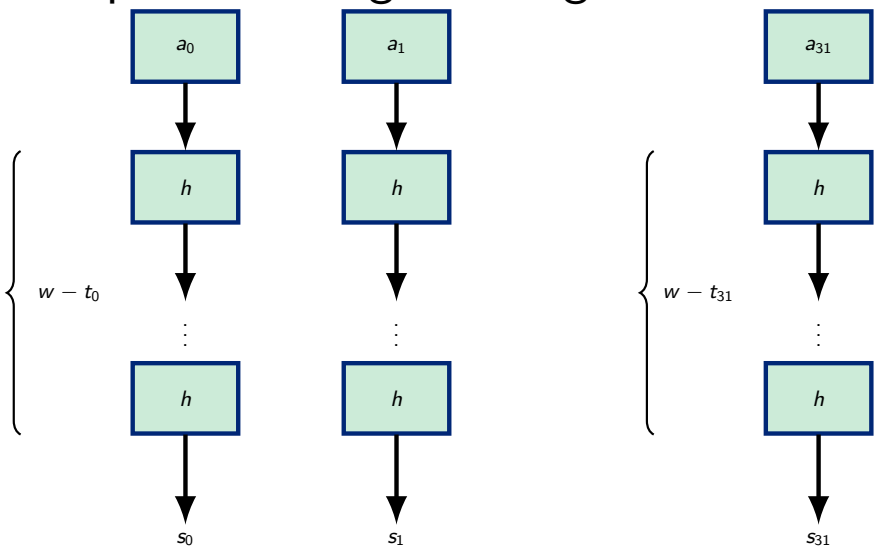


- Computation: ave. n hash eval., much more cost than Lamport OTS
- The signature size $= \frac{n}{w} n = 32 \times 256$, still large

Winternitz (variants) OTS (cont.)

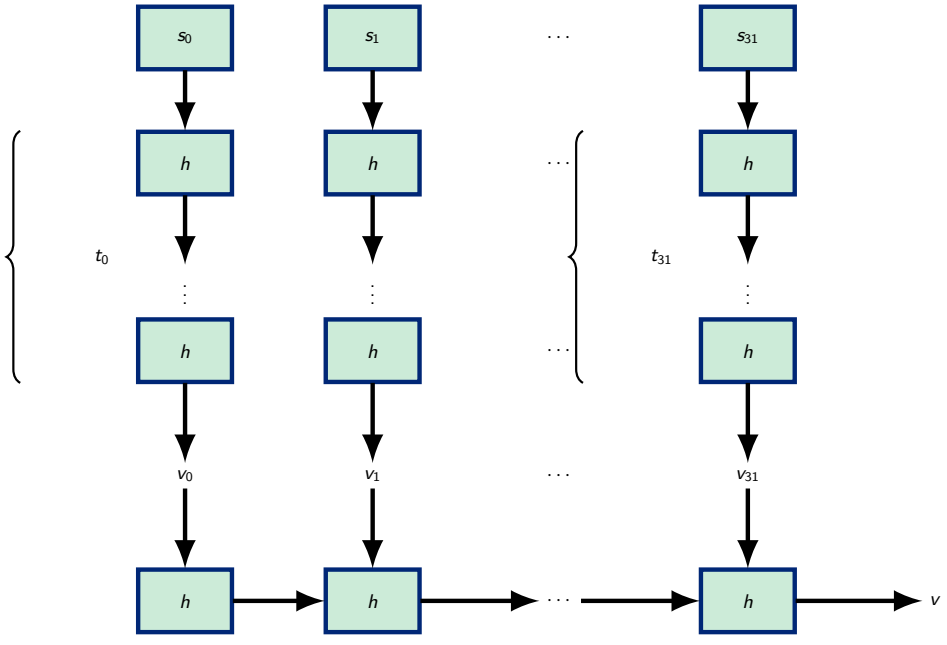
Sig: signing algorithm

The process to sign message m:



- Computation: ave. n hash eval., much more cost than Lamport OTS
- The signature size $= \frac{n}{w} n = 32 \times 256$, still large

Ver: verifying algorithm



- Computing complexity: ave. $n/2 + n/w$ (hash chain) hash evaluations or $n/2 + \log(n/w)$ (Merkle tree) hash evaluations.

U-EPS (Gen, Sig, Ver) Algorithms

Underline cryptographic algorithms

- $\mathcal{H}$: a family of collision resistant hash functions from $\{0,1\}^*$ to $\mathbb{Z}_2^n$.
- $\mathcal{F}$: a family of symmetric encryption algorithms $c = Enc_k(m)$ and $m = Dec_k(c)$, which is secure under chosen plaintext attack (CPA).
- $\mathcal{G}$: a family of sequential memory-hard symmetric encryption algorithms $g(k, m)$, denoted as $c = MemEnc_k(m)$, and $m = MemDec_k(c)$.
- Sequential memory-hard means that parallel algorithms cannot asymptotically achieve efficiency advantage than non-parallel ones.
- This method is introduced for design of the password-based key derivation function scrypt in order to thwart parallel brute-force attacks using GPUs, FPGAs or ASIC chips on passwords.
- The design has been widely used by cryptocurrencies, e.g., Litecoin, Dogecoin and Mastercoin.

U-EPS(Gen, Sig, Ver)

Gen: generate private and public key pair (sk, pk)

- Uniformly generate a, b, each as an n-bit stream, and set the private key $sk = (a, b)$.
- Compute $pk = h(a||b)$, n bits.

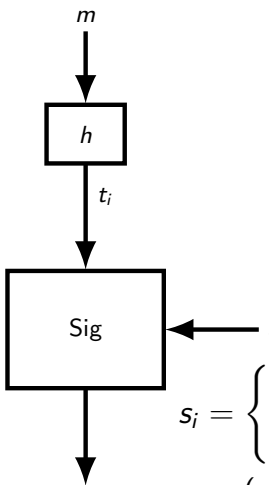
U-EPS(Gen, Sig, Ver)

Gen: generate private and public key pair (sk, pk)

- Uniformly generate a, b, each as an n-bit stream, and set the private key $sk = (a, b)$.
- Compute $pk = h(a||b)$, n bits.

Sig: signing algorithm

Our Sig is a similar Lamport Sig together with an encryption of the sum of the two components in the private key.



$$s_i = \begin{cases} a_i, & \text{if } t_i = 0 \\ b_i, & \text{otherwise} \end{cases}$$

$$s = (s_0, s_1, \cdots, s_{n-1})$$

$$c = Enc_r(a \oplus b), r = h(t||u), u \text{ rand}$$

$$Sig_{sk}(m) = (s, c)$$

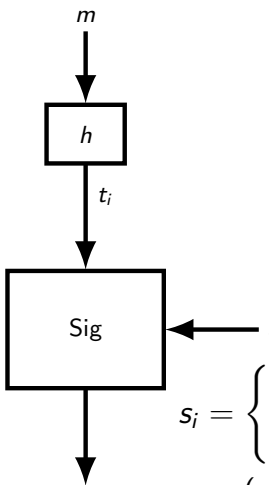
U-EPS(Gen, Sig, Ver)

Gen: generate private and public key pair (sk, pk)

- Uniformly generate a, b, each as an n-bit stream, and set the private key $sk = (a, b)$.
- Compute $pk = h(a||b)$, n bits.

Sig: signing algorithm

Our Sig is a similar Lamport Sig together with an encryption of the sum of the two components in the private key.



$$s_i = \begin{cases} a_i, & \text{if } t_i = 0 \\ b_i, & \text{otherwise} \end{cases}$$

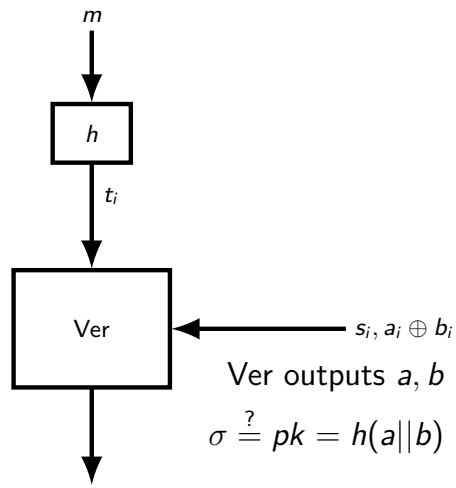
$$s = (s_0, s_1, \cdots, s_{n-1})$$

$$c = Enc_r(a \oplus b), r = h(t||u), u \text{ rand}$$

$$Sig_{sk}(m) = (s, c)$$

Ver: signature verification algorithm.

- Using the delayed release u and t, the hash value of message, to decrypt c to get $a \oplus b$.



$$\sigma \stackrel{?}{=} pk = h(a||b)$$

Outline

Today

1. Symetric
2. Modes
3. Hash
4. Macs
5. LFSR
6. Cryptanalyse

Thank you for your attention.

Questions ?