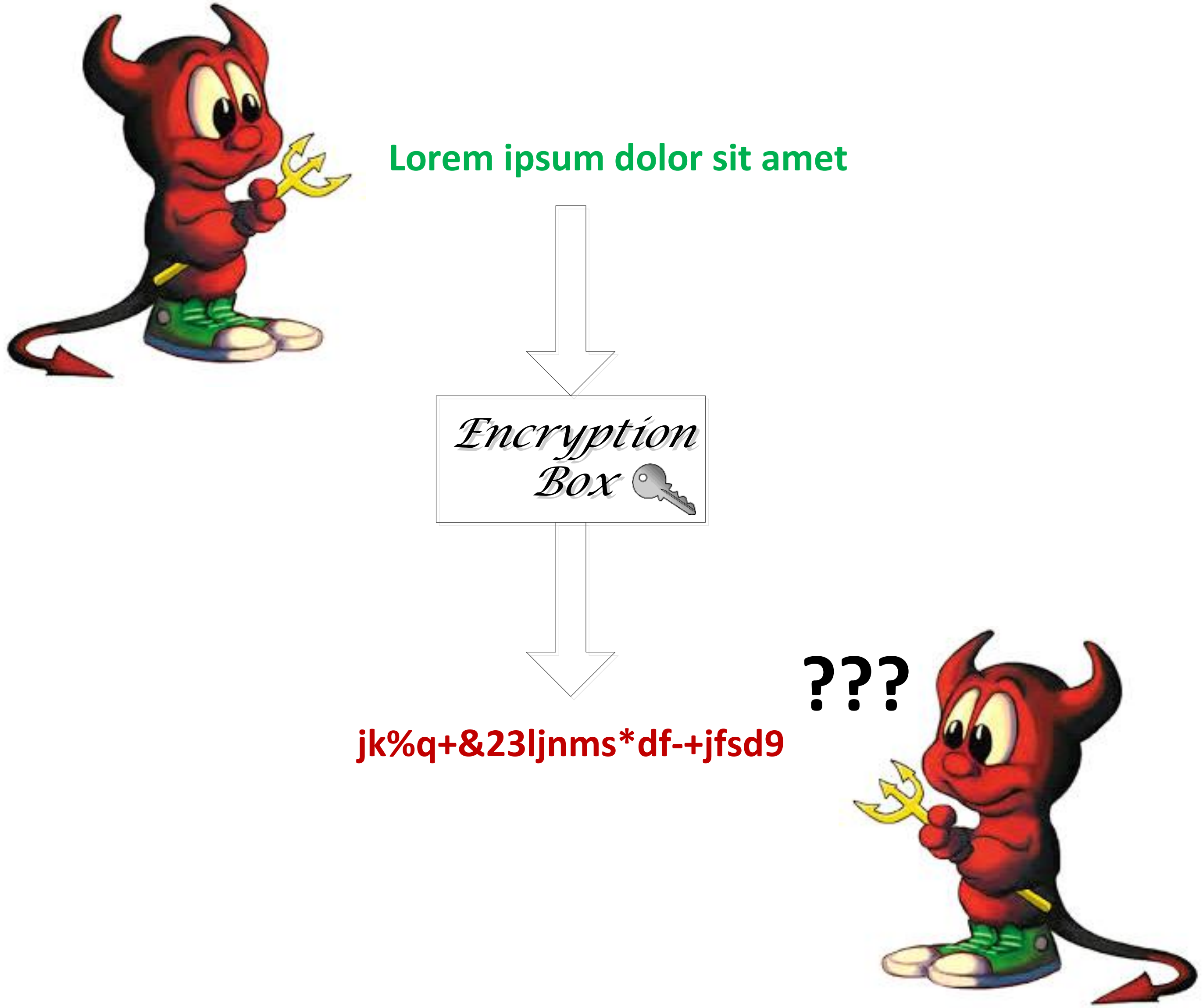
Symmetric Primitives

(block ciphers, stream ciphers, hash functions, keyed hash functions and (pseudo)random number generators)

An informal, yet instructive account of symmetric primitives …

Begin with an informal question

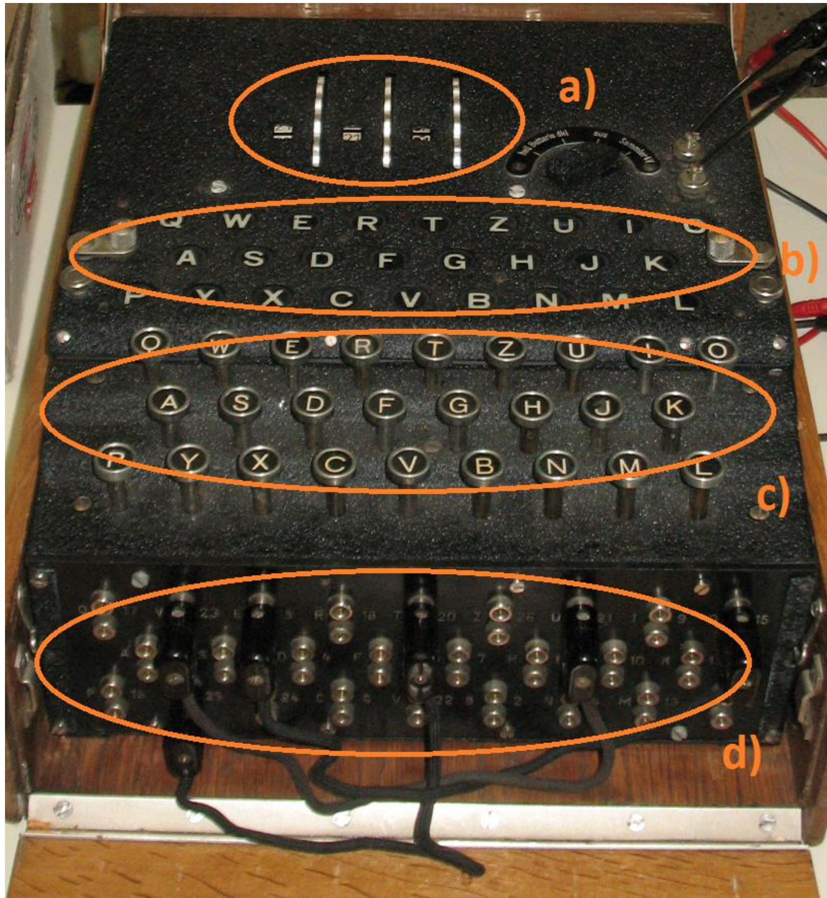
- Question: What do you expect from cryptography?
- (Potentially correct) Answer: Protect you stored data & ongoing communications (let's call this simply protect messages)
- Question: Assume you are given am encryption box (call it symmetric encryption) that encrypts your data with a key. Is your data now protected?
- (At least incomplete) Answer: Yes, as long as the adversary cannot find/guess the key … or maybe not



A practical example – the Enigma machine

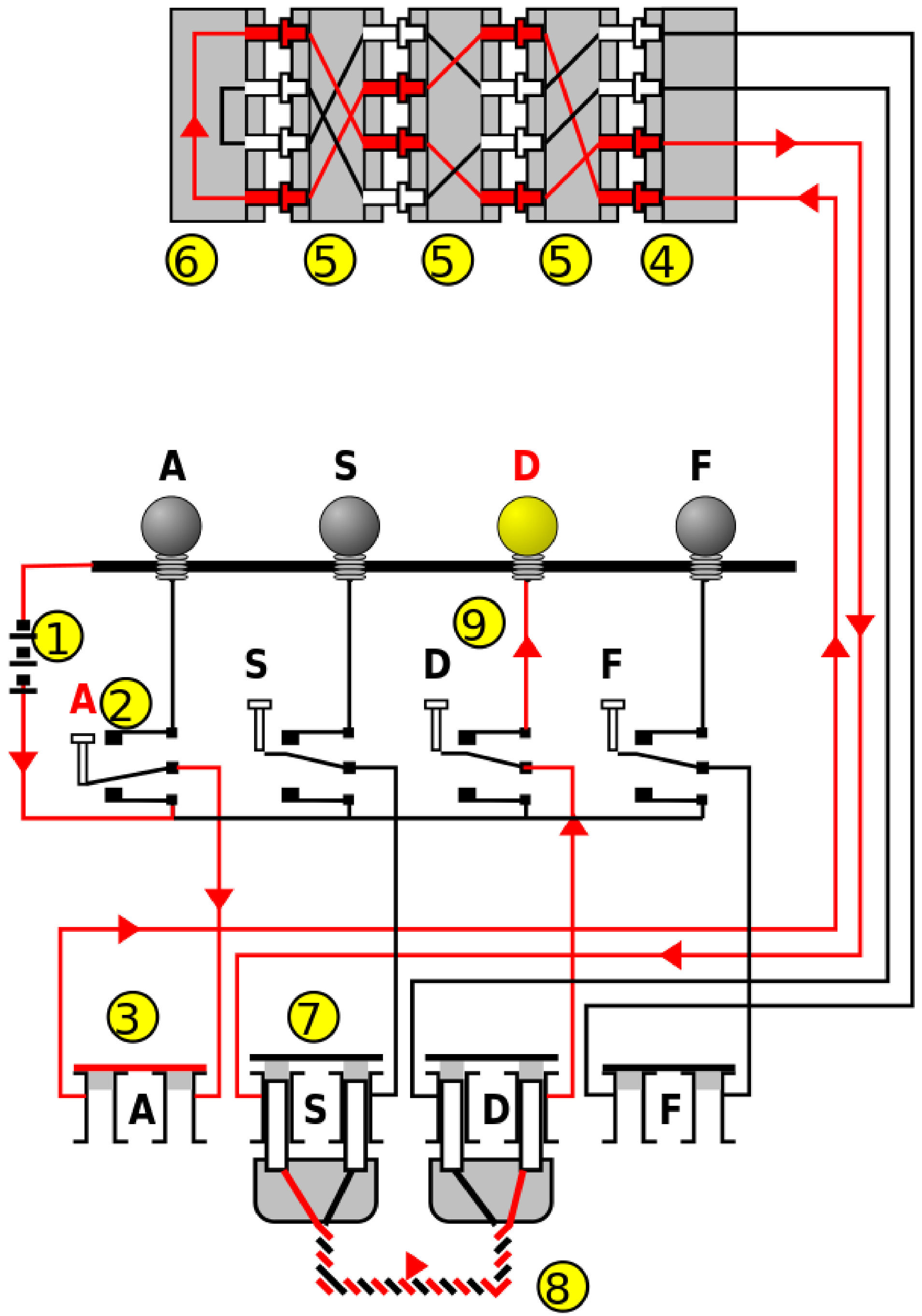
- A rotor cipher machine (several versions of it), elements:
  - 26 lamps (output, ciphertext) & keys (input, plaintext)
  - 3 or 5 (usually) rotors
  - at most 13 plugs that can connect each two letters on the plug-board (part of the key)

a) rotors (3)
b) lamps (26)
c) keys (26)
d) plugboard (2x13)



How Enigma works

- When one key is pressed (letter of the plaintext selected), circuit is closed under that key, current flows through the plugboard that follows the 3 rotors, returns from the reflector and lightens up the lamp (the letter of the ciphertext)
- Rotors move at each step, thus a character will not always get encrypted to the same character (i.e., a polyalphabetic substitution)



- A closer look to the Enigma secret key (depicted in the form of a codebook) may give you more insights on the security of this cryptosystem

- This is a print-screen from a nice tool by Dirk Rijmenants, see http://rijmenants.blogspot.ro/2005/11/enigma-codebook-tool.html

Enigma Codebook Tool

Codebook About Help

GEHEIM! 121 JANUAR 1900

Tag	Walzenlage			Ringstellung	Steckerverbindungen	Kenngruppen
31	I	V	IV	21 21 11	AF BD CY EX GN HI JL MS OR UZ	JVM JFQ JAO ZJG
30	III	V	I	07 06 01	AF BX DP GL HR IM KS NW QY TV	PEL VLB XIO BYG
29	V	I	III	06 13 12	AS BW CM DL ER FG HT IV JY XZ	UEW LZI LWP YJE
28	V	II	III	23 19 21	AP BQ CM DW EO FR GN IL KY VX	ZQW IIJ SVU GGW
27	I	II	IV	23 20 23	BZ CN EP FI GX KY LU MT QR SV	YAT ANE VGM JIB
26	IV	III	I	08 23 08	AQ BI CY DM FX JN KV LS OU RZ	MZX KFY PWL VRY
25	IV	II	I	06 12 26	AS BO CX EW HJ IT KU MZ NQ PR	VNC LMT VGK EIT
24	I	IV	II	03 01 16	BE CR DL FN GZ HX IY KT MU PS	AUT AAZ ZGW FFV
23	III	I	V	13 09 12	AR BY CQ DI EN HS JK LZ MT VW	ZZH GBC IHM FUP
22	V	III	IV	26 13 19	BT CZ DE FN HO IW JV LU PX QS	AQP KKW AIO CSG
21	I	V	II	15 17 25	AI BM DP EK GQ HY LW NZ OR TV	HPP ELN VNJ XHP
20	I	V	IV	23 22 19	BT CK EL FH GU MO NR PX QZ VW	AXU HFM AXQ QKI
19	V	I	III	05 25 23	BE CL FJ HQ IU OP RS TV WX YZ	LMQ SJW JGB DPN
18	IV	II	V	23 21 20	CI DK EV FQ GU HZ JX MW NS RT	CAN GBF VCE XWW
17	III	IV	I	06 22 12	AJ CQ DP EH GK MN OV RU TY XZ	TUC SJF RXC EOC
16	II	V	III	09 24 08	AX CL DY EQ FS HT JO KZ NR PW	MXT WXK AAB LTM
15	V	III	II	23 03 26	BP CV DN EQ GM HT JZ LU RY SW	JWG JJR JHU CKP
14	V	II	III	26 15 18	AL CO DJ FP HU IW KQ MR TY XZ	HAD TCA ATP IYR
13	V	III	IV	18 02 16	BK CR DP FX GY HZ IW JU LS OT	WTR FVS LFH LKZ
12	I	II	V	09 03 15	AW BS CU EF GL HZ JY MV OR PX	YYR TJF DSZ MLX
11	II	I	IV	07 17 08	AU BV DN ES GP HZ KL MW RT XY	EDQ LWV PXQ OUL
10	V	I	III	08 06 01	AI CV DF EO HU JP KS NW RT YZ	QRY JZO XYF GRF
09	I	II	V	05 07 16	AE BQ DN FV GY HM JR KZ OW UX	BUW WKP NDI YAA

How secure is Enigma

- Question: how hard is to break Enigma?
- Answer (not necessarily correct): as hard as to find the key
- Question: how big is Enigma's key?
- Answer: consider just (the way to place 3 rotors) x (the way to connect 13 plugs)

$$26^3 \times \frac{26!}{13! \times 2^{13}} = 138953282533065000$$

when compared to the number of DES keys $2^{56} = 72057594037927936$ will quickly lead to the conclusion that Enigma (deprecated by the end of WW2) is stronger than DES (deprecated only by the end of the '90s)

How secure is Enigma

- Question: imagine you have captured a ciphertext that begins with:

zeyt sadb dikf dsak sadk jnujj

Could you tell which is the corresponding plaintext from the following:

a) attackatdawnonthewestfront
b) attackatnightonthewestfront
c) attackatduskonthewestfront

- Answer: wrong design decision in Enigma, a letter cannot map to itself! Correct answer is c)

Partial conclusion

- For protecting data by symmetric primitives we need: **clear design principles** (how to build the ciphers) and a **formal treatment of security properties** (what is the exact security they should offer)

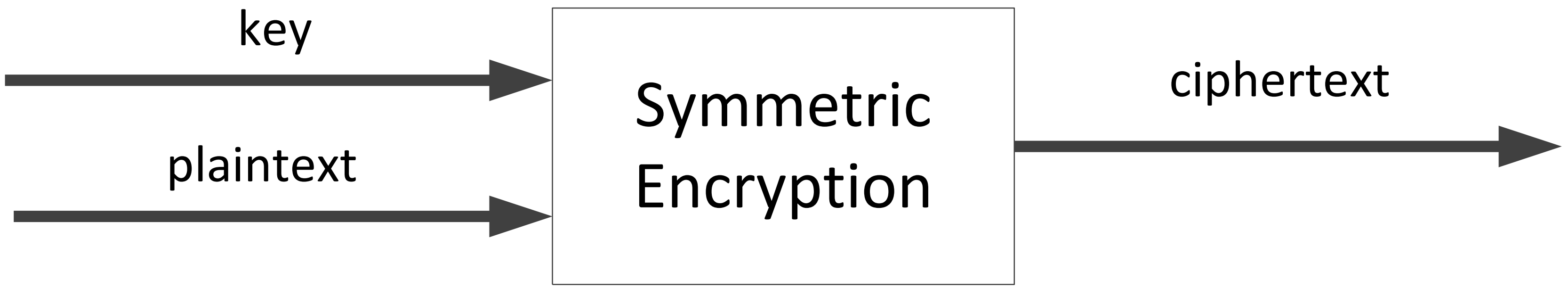
A more formal and constructive account of symmetric primitives …

you should learn:

i. where is the primitive used,
ii. what are the standards,
iii. how is it built,
iv. what are its properties

Type of functions (I) Symmetric encryption schemes

- Description (informal): an algorithm that takes as input a key k and message m called plaintext and returns the encrypted message c called ciphertext (similarly, algorithms for decryption and generating keys are needed)



i.e., $c = e_k(m)$

- Example of use: encrypted tunnels SSL/TLS, IPSEC; encrypted passwords (lmhash in Win XP); encrypted hard drives (TrueCrypt), etc.
- Standards:
    - Not to use: DES, RC4
    - To use: AES (128, 194, 256), 3DES (with 168 bit key, not recommended)

Symmetric encryption: formal definition

- A symmetric encryption scheme is a **triple of algorithms**:
  - ***Gen*** *is the key generation algorithm that takes random coins, a security parameter (l) and outputs the key* $k \leftarrow Gen(1^l)$
  - ***Enc*** *is the encryption algorithm that takes as input the key and some message, then outputs the ciphertext* $c \leftarrow Enc(k, m)$
  - ***Dec*** *is the decryption algorithm that takes as input the ciphertext and the key and outputs the message* $m \leftarrow Dec(k, c)$

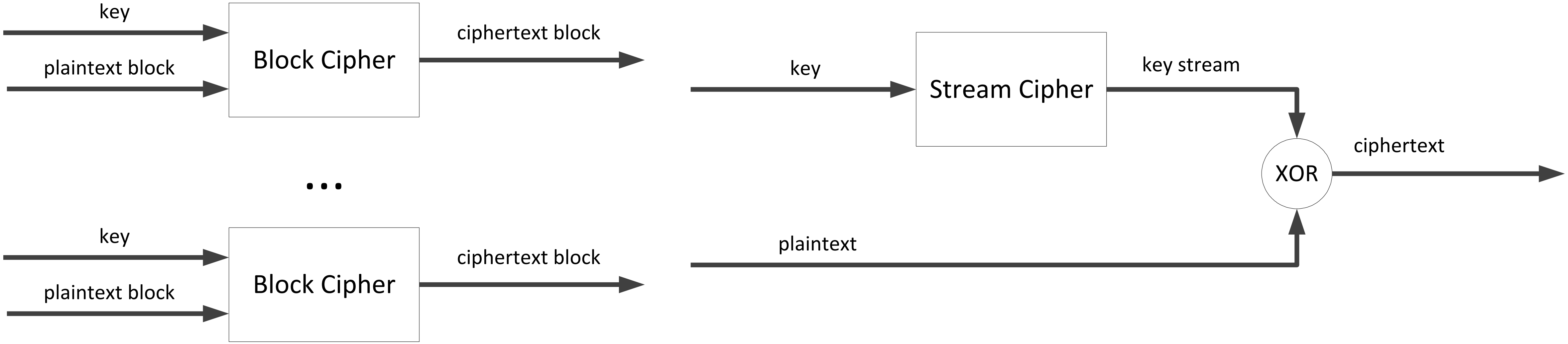
- A correctness condition enforces that $Dec(k, Enc(k, m)) = m$
- In some cases, the encryption and decryption algorithms are allowed to return \null on particular inputs (i.e., they refuse to encrypt/decrypt)

Design principle: product ciphers

- Substitutions and transpositions (suggested in the work of Shannon, also used before)
  - **Substitution (S-Box)** replaces a symbol (or group of symbols) by another symbol – creates confusion
  - **Permutations (P-Box)** also known as transpositions mixes the symbols inside a block – creates diffusion
- Ciphers that use both substitutions and permutations (S-Boxes and P-boxes) are also called **product ciphers** (sometimes product ciphers denote any cipher that uses more than one transformation, while product ciphers with only S&P are called **SP-networks**)
- Remarks:
  - DES and AES, the two well known standards are product ciphers
  - Feistel ciphers are also product ciphers

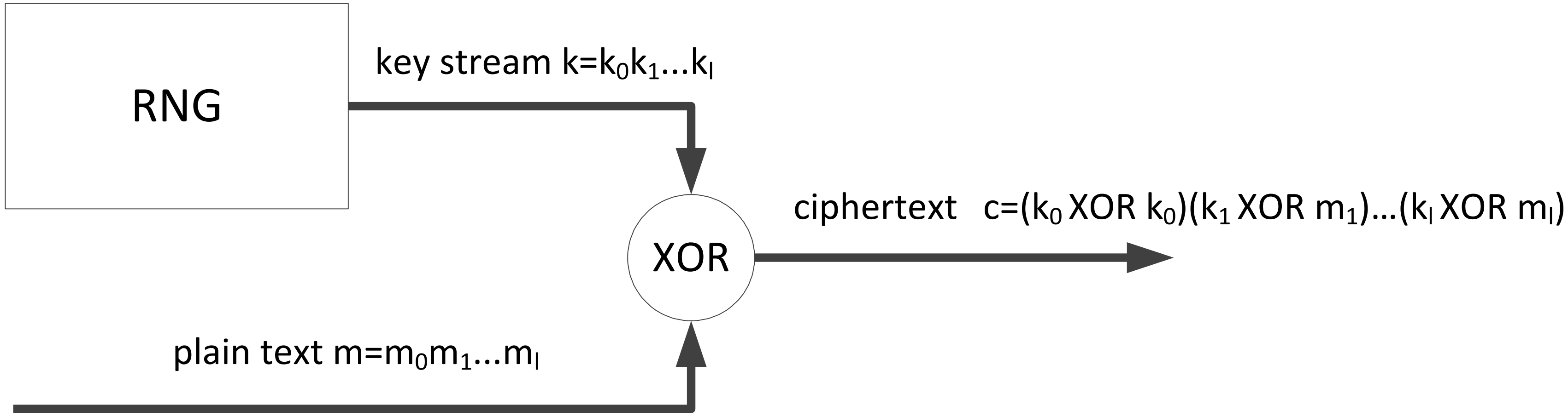
Classification: block ciphers vs. stream ciphers

- **Stream ciphers** – the message is combined via a simple transformation (e.g. XOR) with a keystream (which is a pseudorandom stream generated by a more complex mechanism), operation is done one character (bit) at a time. Examples include RC4 used in SSL/TLS or A5 used in GSM.
- **Block ciphers** – the message is transformed block by block (e.g., 128 bits) via a transformation that is depended on the key. Examples include DES, 3DES, AES.
- Remarks:
  - Block ciphers can be turned into stream ciphers in certain mode of operations, e.g., counter mode (this means that distinction between the two is not always clear)
  - Typically stream ciphers have low hardware complexity, are fast, but practical instantiations such as RC4 are not always secure



Example: the one-time pad (a stream cipher)

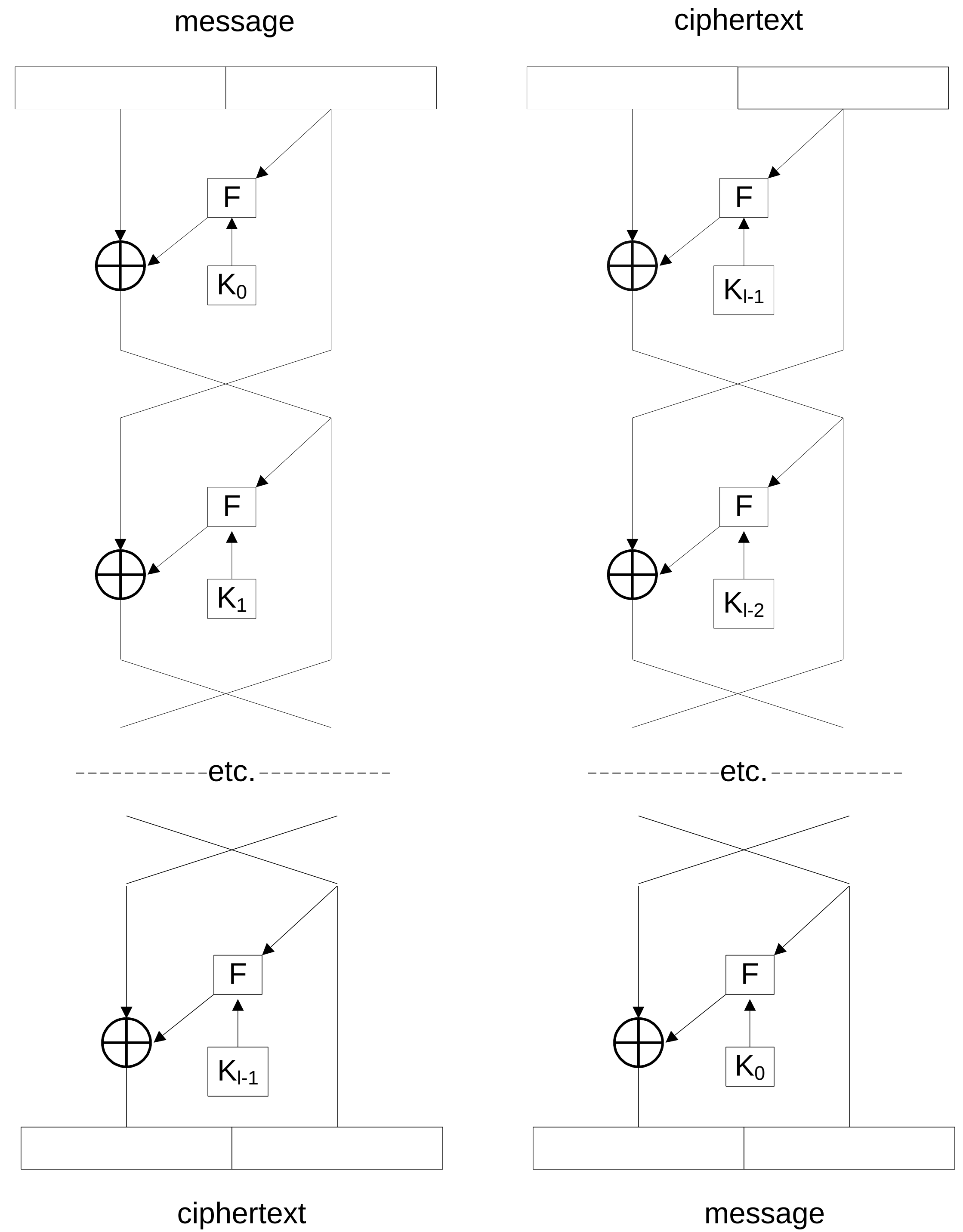
- Question: could you build a cipher that cannot be broken regardless of the computational power of the adversary?
- Answer: believe it or not, yes. The one-time pad is information-theoretically secure, i.e., cannot be broken regardless of computational power & ciphertext available.
- Description: generate a random key the same length as the plaintext, then simply XOR it with the plaintext



- Problems:
  - requires a random key stream the same length as the plaintext, but in practice you want a key as small as possible
  - Since it's symmetric the key needs to be exchanged a-priori on a secure channel, but then why not simply exchange the plaintext?
- Current status: there are still some practical applications where it's useful, e.g., quantum cryptography, otherwise it is not an efficient solution

Design: Feistel networks

- Designed by Horst Feistel in the ’70s at IBM
- SP-networks
- How they work:
  - ➢Variable number of rounds
  - ➢Each block is split into right and left part (if equal in size, then the network is called balanced)
  - ➢Right block is passed through a round function that depends on the round key
  - ➢Round key is derived from the master key (via the key scheduling algorithm)
  - ➢Security/performance trade-off: increasing the number of rounds and the size of the key results in increasing security level
  - ➢Decryption is performed by walking through the circuit in reverse order



Relevant property of the Feistel round

- Note that the Feistel round is invertible regardless of the properties of the round function, so inverting the network is straight forward as follows
  - By definition, deriving the output from the input:

$$L_i = R_{i-1}, R_i = L_{i-1} \oplus f_i(R_{i-1})$$

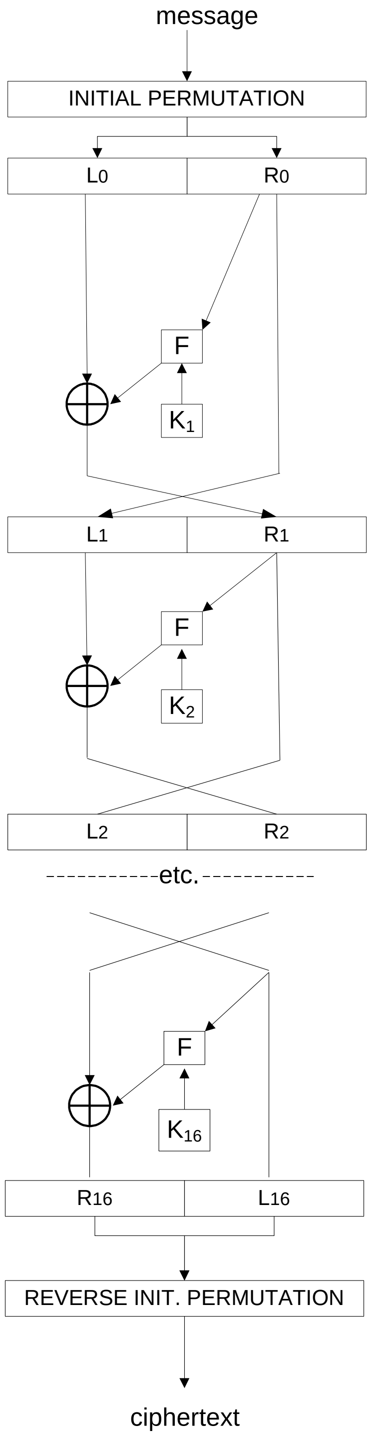
  - Which implies, deriving the input from the output

$$R_{i-1} = L_i, L_{i-1} = R_i \oplus f_i(L_i)$$

Design insights: DES

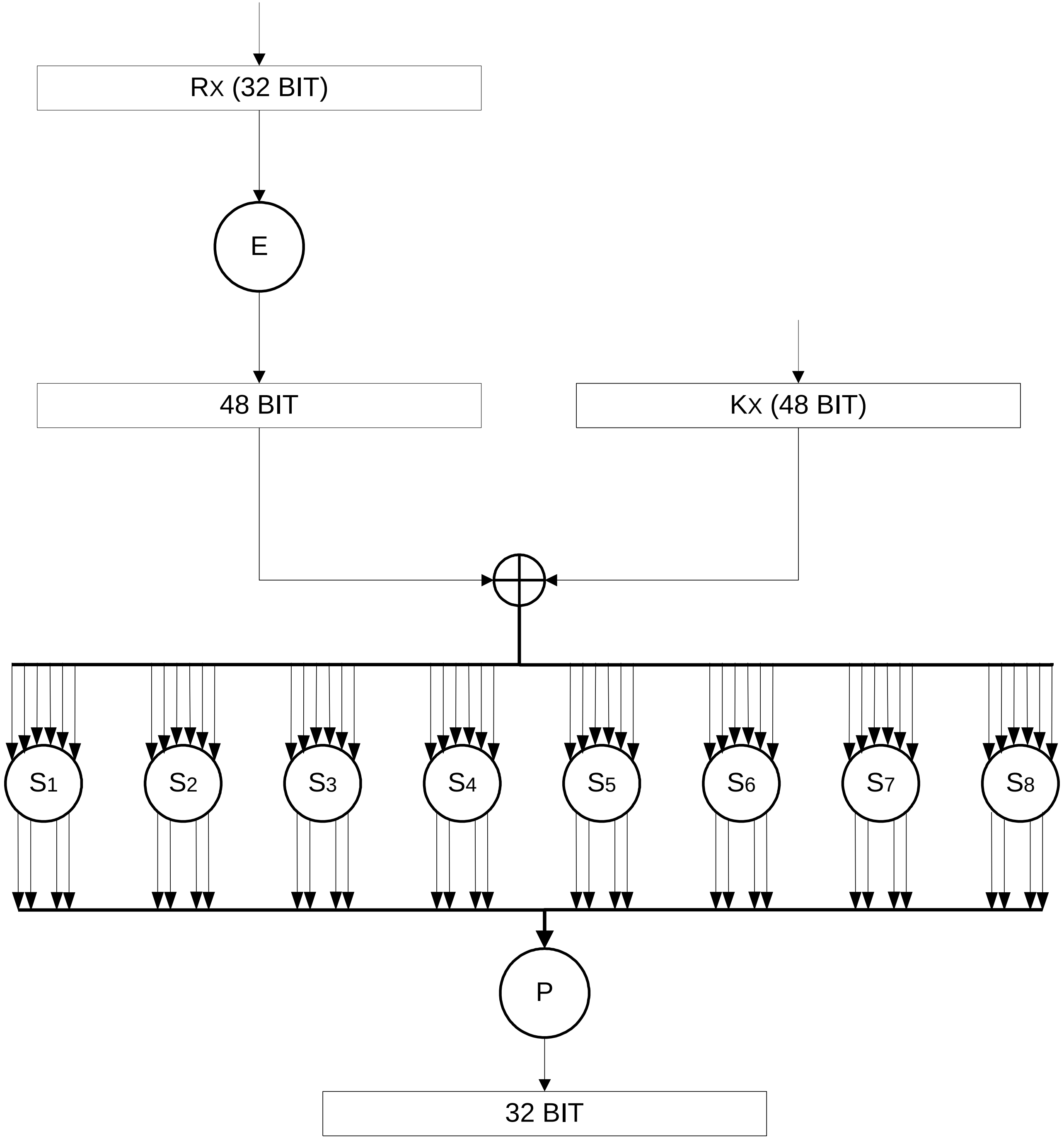
- Some DES facts:
  - ➢Developed in the 70s at IBM based on Feistel's design
  - ➢Standardized with the input from NSA
  - ➢Symmetric encryption standard between 1977-2001
  - ➢Considered insecure since the end of the 90s
  - ➢Replaced by AES (Rijndael) in 2001
  - ➢DES is a 16 round Feistel network
  - ➢DES operates on 64 bit blocks
  - ➢Surprisingly, DES key is only 56 bits

- Some DES oddities:
  - ➢DES has four weak keys: encryption and decryption have the same effect with these keys
  - ➢DES has six pairs of semi-weak keys: encryption with one key from the pair behaves as decryption with the other



DES round function

- How it works: the right half (32 bit) of the message block (64 bit) is expanded (48 bit) then XOR-ed with the round key (48 bit) and each 6 bits are provided as input to 8 x S-Boxes that output only 4 bits resulting in 32 bits that are passed through another permutation P
- This round transformation is applied 16 times, each time with a distinct round key



Examples: E, P and some S-boxes (from the standard)

$$E = \begin{pmatrix} 32 & 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 \\ 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 \\ 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 \\ 28 & 29 & 30 & 31 & 32 & 1 \end{pmatrix}$$

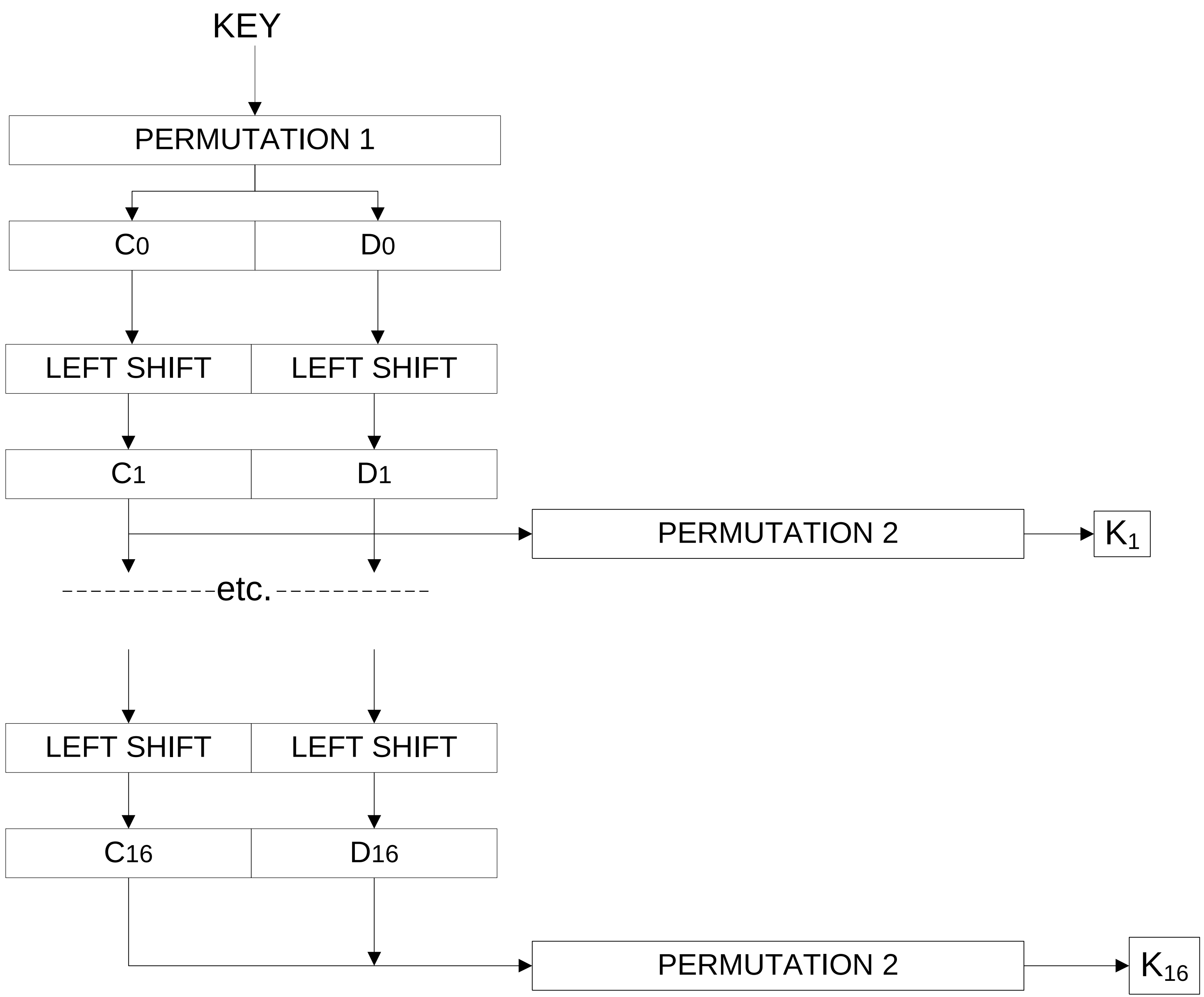
$$S_1 = \begin{pmatrix} 14 & 4 & 13 & 1 & 2 & 15 & 11 & 8 & 3 & 10 & 6 & 12 & 5 & 9 & 0 & 7 \\ 0 & 15 & 7 & 4 & 14 & 2 & 13 & 1 & 10 & 6 & 12 & 11 & 9 & 5 & 3 & 8 \\ 4 & 1 & 14 & 8 & 13 & 6 & 2 & 11 & 15 & 12 & 9 & 7 & 3 & 10 & 5 & 0 \\ 15 & 12 & 8 & 2 & 4 & 9 & 1 & 7 & 5 & 11 & 3 & 14 & 10 & 0 & 6 & 13 \end{pmatrix}$$

$$S_2 = \begin{pmatrix} 15 & 1 & 8 & 14 & 6 & 11 & 3 & 4 & 9 & 7 & 2 & 13 & 12 & 0 & 5 & 10 \\ 3 & 13 & 4 & 7 & 15 & 2 & 8 & 14 & 12 & 0 & 1 & 10 & 6 & 9 & 11 & 5 \\ 0 & 14 & 7 & 11 & 10 & 4 & 13 & 1 & 5 & 8 & 12 & 6 & 9 & 3 & 2 & 15 \\ 13 & 8 & 10 & 1 & 3 & 15 & 4 & 2 & 11 & 6 & 7 & 12 & 0 & 5 & 14 & 9 \end{pmatrix}$$

$$P = \begin{pmatrix} 16 & 7 & 20 & 21 \\ 29 & 12 & 28 & 17 \\ 1 & 15 & 23 & 26 \\ 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 \\ 32 & 27 & 3 & 9 \\ 19 & 13 & 30 & 6 \\ 22 & 11 & 4 & 25 \end{pmatrix}$$

DES key scheduling

- Derives each of the round keys from the master key



Designs: 3DES

- 3 DES keys K1, K2, K3 in the following transformation:

$$c = E_{K3}\left(D_{K2}\left(E_{K1}(m)\right)\right), \qquad m = D_{K1}\left(E_{K2}\left(D_{K3}(c)\right)\right)$$

- Considered to be secure so far (given that all three keys are random and independent) but it is slower than AES (thus no serious reasons for use in practice)
- Has 3 keying options: i. independent keys, ii. K1 and K2 independent but K3=K1, iii. all keys are equal K1=K2=K3 (this is DES)
- Main reason for practical persistence may be the electronic payment industry

Designs: AES

- AES facts:
  - Designed by Vincent Rijmen and Joan Daemen
  - Selected by public competition from the 5 finalists: MARS, RC6, **Rijndael**, Serpent, and Twofish
  - The new standard as of 2001
  - Not a Feistel network
  - Available with 3 key lengths: 128, 192, 256 bits
- How AES works
  - Operates on a 4x4 matrix of bytes (128 bit blocks) called state
  - Has 10, 12 or 14 rounds according to the key size
  - Each round has 4 transformations: SubBytes (a substitution) is non-linear substitution where each byte is replaced via a look-up table, ShiftRows (a permutation) the last three rows are shifted, MixColumns the four bytes of each column are combined via a linear transformation, AddRoundKey each byte of the state is combined with the round key via a XOR operation

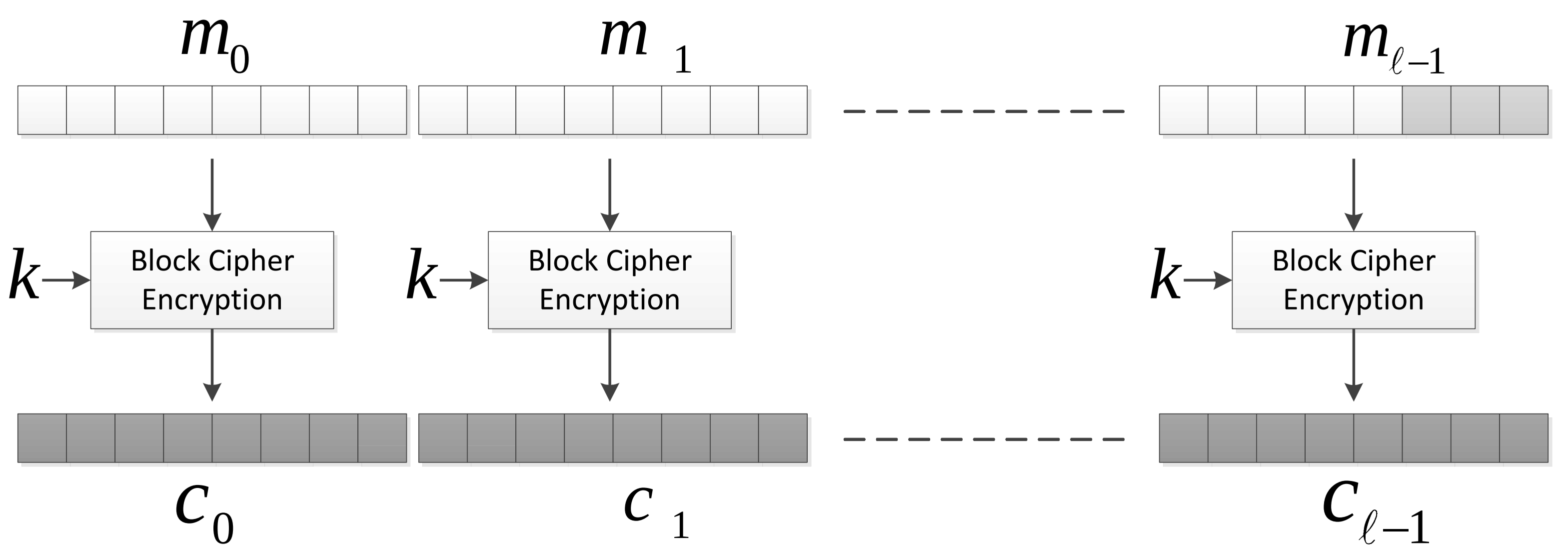
AES_Encrypt_Round(State, Key)	AES_Decrypt_Round(State, Key)
{	{
SubBytes(State) ;	AddRoundKey^{-1}(State, Key);
ShiftRows(State);	MixColumns^{-1} (State);
MixColumns(State);	ShiftRows^{-1} (State);
AddRoundKey(State, Key);	SubBytes^{-1} (State) ;
}	}

Block Ciphers use in practice

- Question: block ciphers work on single blocks of message, how do you extend them to multiple blocks?

Electronic Code Book (ECB)

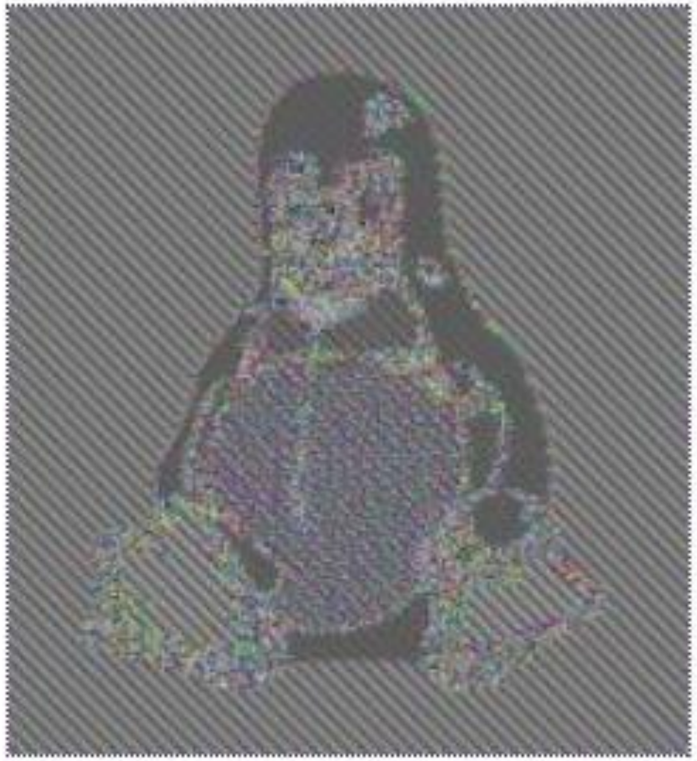
- The message is parsed into blocks and each block is encrypted with the secret key
- Decryption is done by reversing this operation



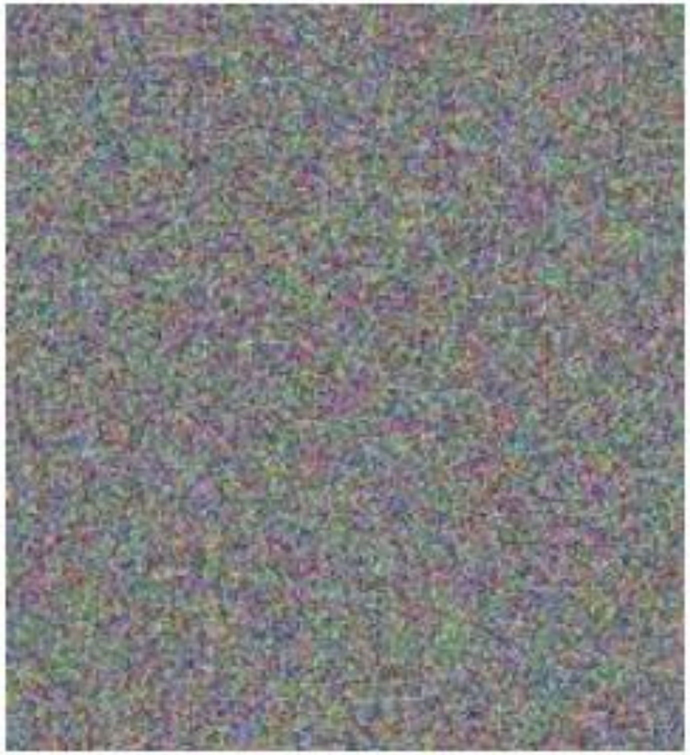
- Question: assuming that the block cipher is secure, is this construction secure?

- Answer: No. Do not use ECB.

Original



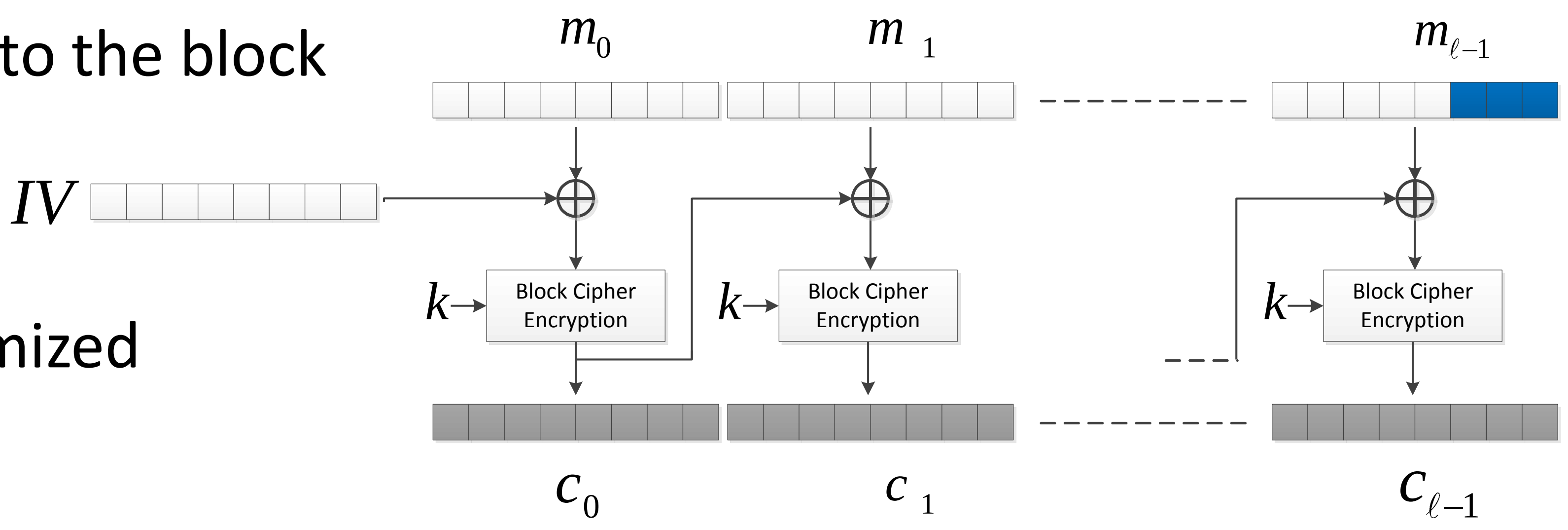
Encrypted using ECB mode



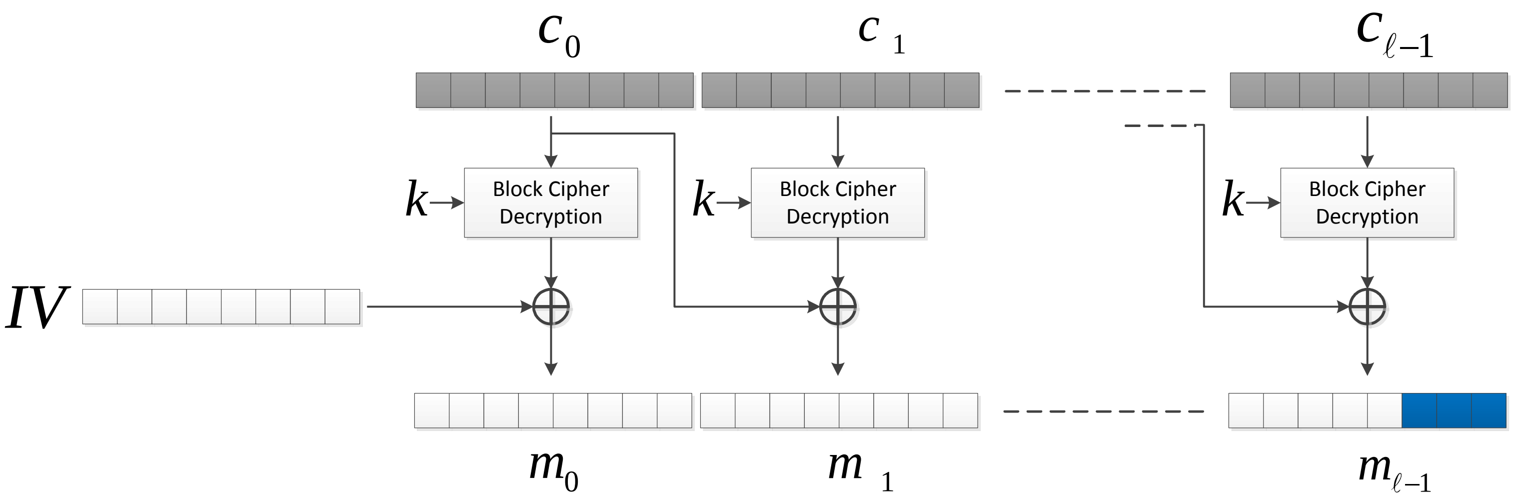
Other modes than ECB results in pseudo-randomness

Cipher Block Chaining (CBC)

- Initialization Vector (IV) is a non-secret random value used for randomization of the first output block
- Last message chunk is padded to the block length



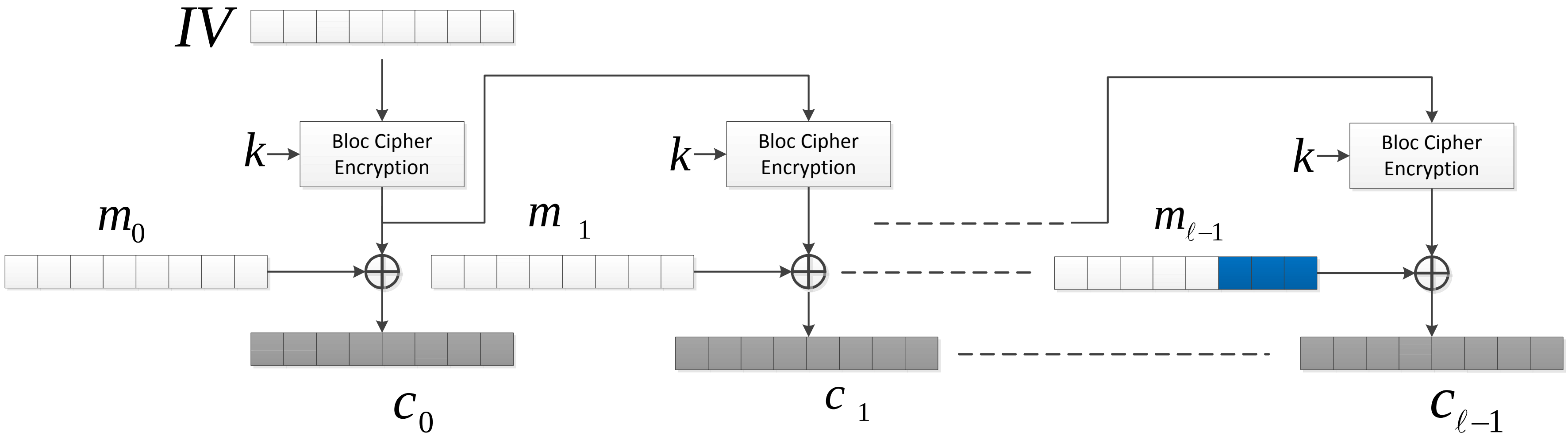
- Pros: encryption is fully randomized and secure

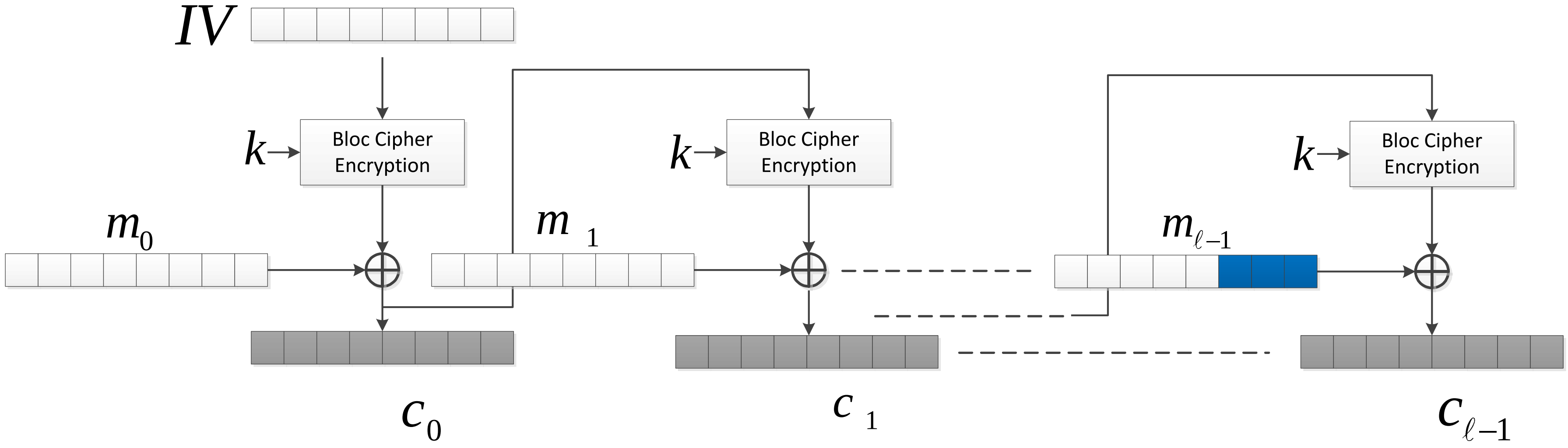


- Cons: if one of the blocks is lost, decryption cannot be performed

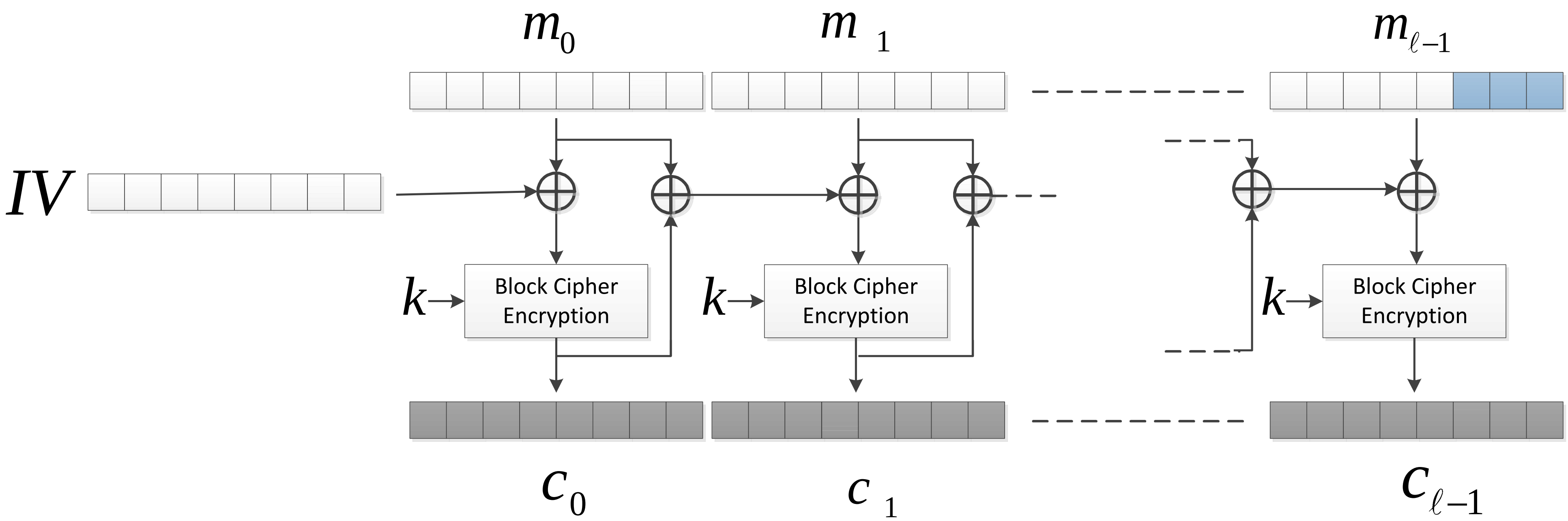
Some variations: Output-feedback (OFB) and Cipher Feedback (CFB)

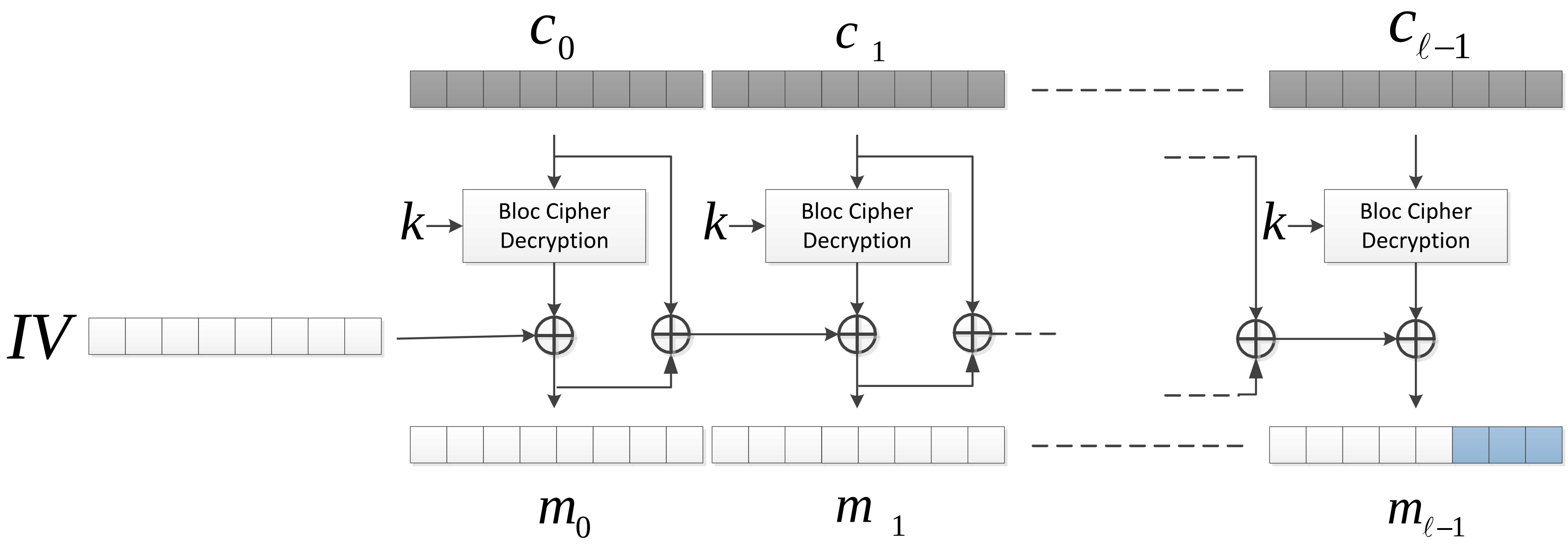
- Pros: OFB allows decryption even when message blocks are lost, it also allows pre-computation of the key stream





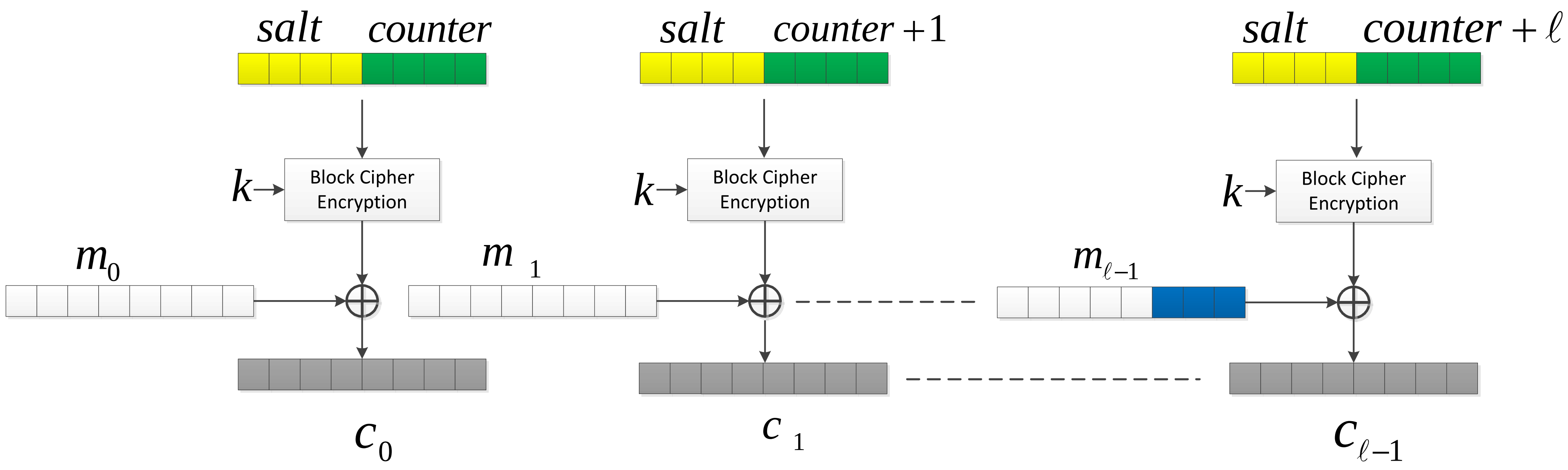
Another variation: Propagating Cipher Block Chaining (PCBC)





Counter Mode

- A counter is incremented and encrypted for each block, then XORed with the message
- Pros: decryption can still be performed if bocks are lost, key-stream can be pre-computed
- This mainly converts the block cipher into a stream cipher



Adversary capabilities (informal) – what the adversary can do?

- **CPA** – chosen plaintext adversary, an adversary that has access to a black-box that encrypts plaintexts at the adversary choice
- **CCA** – chosen ciphertext adversary, an adversary that has access to a black-box that decrypts cyphertexts at the adversary choice
- **Adaptive vs. non-adaptive** – is an additional flavour that can be added to both CPA and CCA meaning that the adversary can continue (adaptive) or not (non-adaptive) to query the encryption/decryption box after he received the target ciphertext that he is required to break (obviously the adversary is not allowed to query the target ciphertext to the decryption box)

Security notions (informal)

- semantic security (SS) (Goldwasser & Micali 1982)

  Any information that can be efficiently computed with the ciphertext, can be also computed without the ciphertext

- indistinguishability of ciphertexts (IND)

  Given two messages selected by the adversary and the encryption of one of them chosen at random (without adversary's knowledge) the adversary cannot decide which is the encrypted message

- real or random indistinguishability (RoR)

  Given a message selected by the adversary and the encryption of either this message or some complete random message (not known to the adversary) the adversary cannot decide if the ciphertext corresponds or not to its chosen plaintext

- Question: which of the previous properties is the strongest?
- Answer: under proper formalization they are all equivalent, see Goldreich – Foundations of Cryptography, vol II, p.383
- Question: which is easier to prove?
- Answer: generally IND or RoR are easier to prove and are the standard tool in proving security

How to prove equivalences?

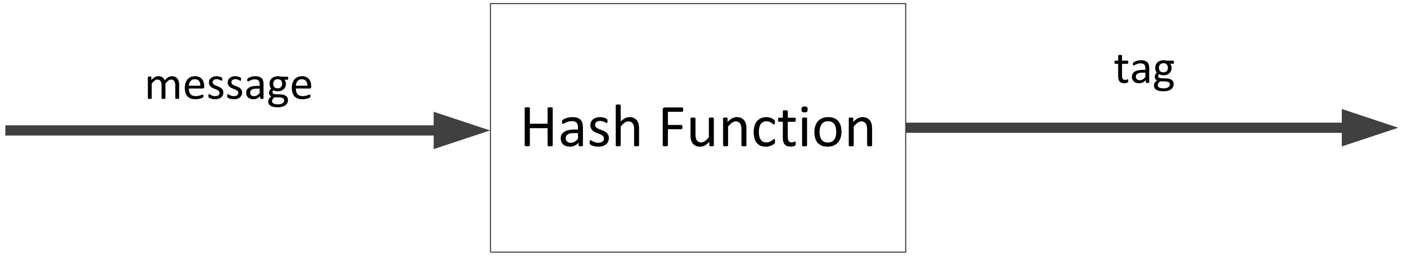
- Security reductions, proving that a cryptosystem that has one property has the other (or the reverse, if it doesn’t have one property it doesn’t have the other)

Example, security reductions: IND → RoR & IND ←RoR

- Proof to be done as exercise during laboratory hours

Type of functions (II) Hash functions

- Description: an algorithm that takes as input a message of any length and turns it into a constant size output (usually referred as tag or simply hash)



i.e., $tag = H(m)$

- Example of use: assure integrity of software downloads/updates, protect stored passwords, etc.

e.g., downloading images from ubuntu.com

14.10

(Utopic Unicorn): October 2014 (Supported until July 2015)

md5 Hash	Version
08494b448aa5b1de963731c21344f803	ubuntu-14.10-desktop-amd64.iso
4a3c4b8421af51c29c84fb6f4b3fe109	ubuntu-14.10-desktop-i386.iso
91bd1cfba65417bfa04567e4f64b5c55	ubuntu-14.10-server-amd64.iso

- Standards:
  - ➢Not to use MD5, SHA1 (not resistant to collisions)
  - ➢To use SHA2 (mostly 256, 384 and 512 are somewhat slow)
  - ➢Future use: SHA3 (Keccak the winner of the competition)
  - ➢Alternatives: BLAKE is a lightweight design, one of the SHA3 finalists

Security properties for hash functions

- The following properties are mandatory for hash functions:
  - **Pre-image resistance** – given the hash of some message it is infeasible to find the message

    i.e., $h(m) \xrightarrow{?} m$

  - **Secondary pre-image resistance** – given the hash of a message and the message it is infeasible to find a second message that has the same hash value

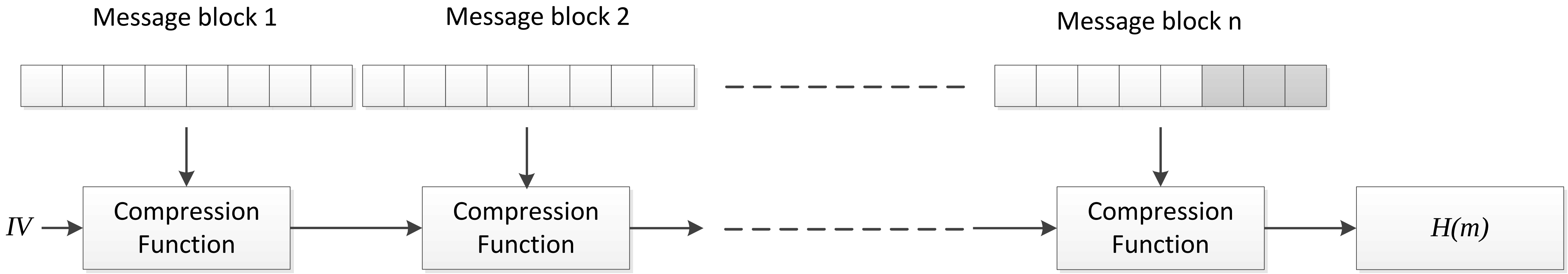
    i.e., $m_1, h(m_1) \xrightarrow{?} m_2 \; s.t. \; h(m_1) = h(m_2)$

  - **Collision resistance** – it is infeasible to find two messages that have the same hash

    i.e., $\xrightarrow{?} m_1, m_2 \; s.t. \; h(m_1) = h(m_2)$

Design principle

- The Merkle-Damgard construction provides a method for turning a collision-resistant one-way functions into a collision-resistant hash functions
- This design stands behind MD5, SHA1 and SHA2
- The IV is fixed (not random like in block ciphers modes of operation)



Design insights: MD5

- 4 IV’s defined as follows

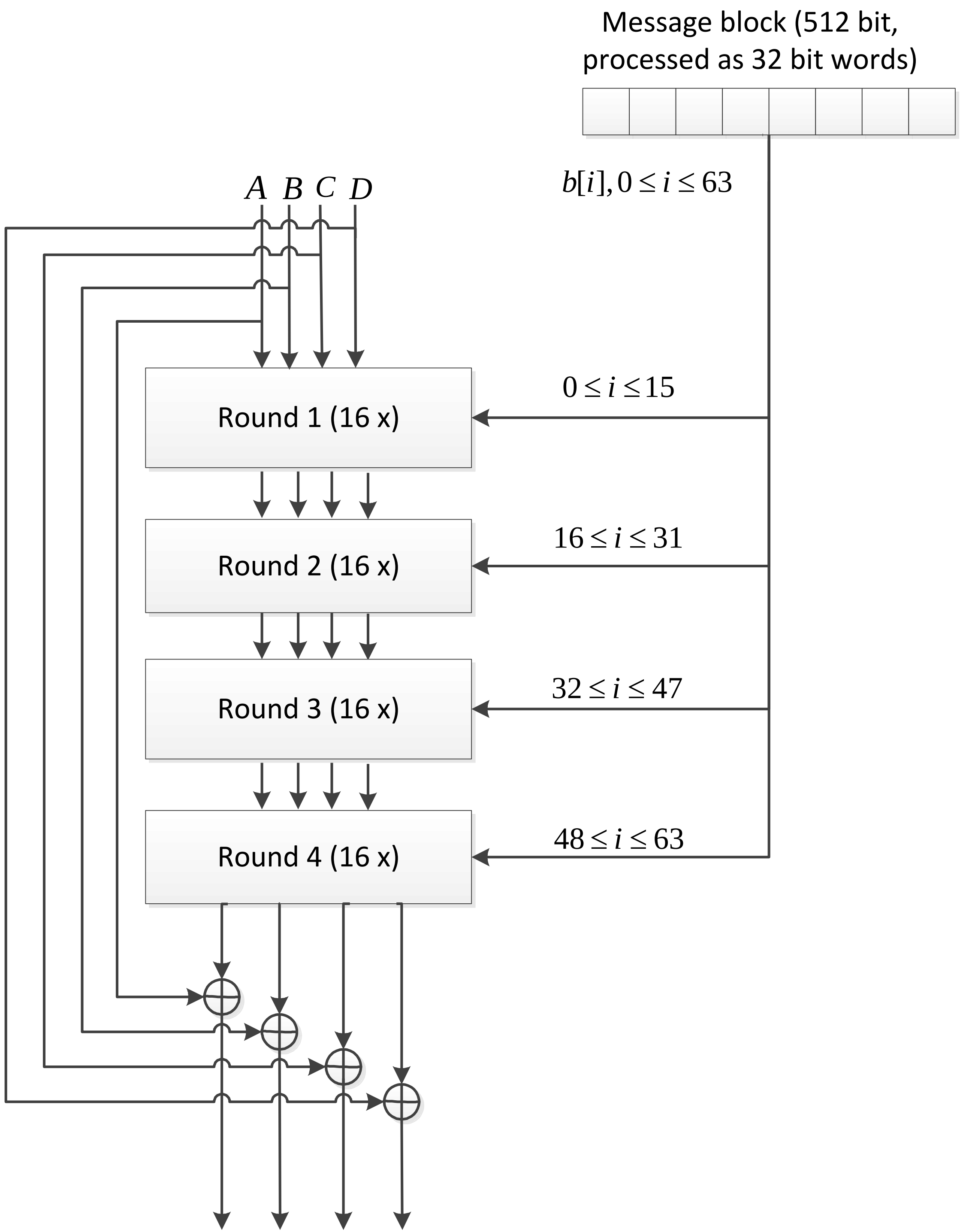
A= 0x67452301,

B= 0xefcdab89,

C= 0x98badcfe,

D= 0x10325476.

- Message is processed in blocks of 512 bits that are further split in 128 bit chuncks and propagated as IVs for the next block to be hashed (i.e., Merkle-Damgard construction)



MD5 round function

- Each round proceeds with the following transformation (A, B, C, and D are the IV's, K and S are fixed constants and M is the message):

$$D \leftarrow C,$$
$$C \leftarrow B,$$
$$B \leftarrow B + ((A + FR(B, C, D) + M + K) <<< S),$$
$$A \leftarrow D.$$

- Round function is distinct for each round (still, all round functions consist in simple logic operations AND, OR, XOR and NOT):

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z),$$
$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z),$$
$$H(X,Y,Z) = X \oplus Y \oplus Z,$$
$$I(X,Y,Z) = Y \oplus (X \vee \neg Z).$$

Test vectors as per RFC 1321

- Examples of what you get after you hash

```
MD5 ("")           = d41d8cd98f00b204e9800998ecf8427e
MD5 ("a")          = 0cc175b9c0f1b6a831c399e269772661
MD5 ("abc")        = 900150983cd24fb0d6963f7d28e17f72
MD5 ("message digest") = f96b697d7cb7938d525a2f31aaf161d0
MD5 ("abcdefghijklmnopqrstuvwxyz")=c3fcd3d76192e4007dfb496cca67e13b
MD5 ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789")
     = d174ab98d277d9f5a5611c2c9f419d9f
MD5("12345678901234567890123456789012345678901234567890123456...2345678
90") = 57edf4a22be3c955ac49da2e2107b67a
```

Type of functions (I) Keyed Hash Functions (or MACs)

- Description (informal): an algorithm that takes a message of arbitrary length and and a key then outputs a tag

i.e., $tag = MAC_k(m)$

- Example of use: assuring message authentication, i.e., binding a message with the identity of a principal that knows a key
- Standards:
  - Not to use: simple concatenation of key to a message is in general insecure
  - To use: HMAC or NMAC with one of the previous hash functions
  - Future use: N/A

Message Authentication Codes formal definition

- A message authentication code is a **triple of algorithms**:
  - ***Gen*** *is the key generation algorithm that takes random coins, a security parameter l and outputs the key* $k \leftarrow Gen(1^l)$
  - ***Mac*** *is the tag-generation algorithm that takes as input the key and some message, then outputs the tag* $tag \leftarrow MAC(k, m)$
  - ***Ver*** *is the verification algorithm that takes as input the key, the tag and the message and outputs 1 if the tag is valid or 0 otherwise* $m \leftarrow Ver(k, tag, m)$

- A correctness condition enforces that $Ver(k, MAC(k, m), m)$ =1

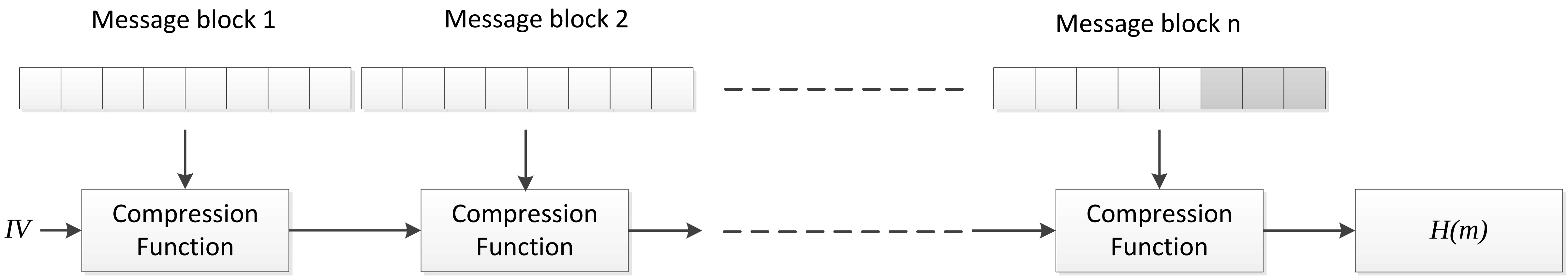
Desired Properties for MACs

- Fortunately, there is only one strong definition of security (of course, this can be refined in several ways)
- MACs must have **(existential) unforgeability under chosen message attacks**, that is, an adversary that receives any number of valid message-tag pairs (i.e., pairs that are computed with the MAC algorithm) is unable to output a new message-tag pair that will successfully pass through the verification algorithm

What not to use

- Question: based on the previous security definition for MAC code, is the simple concatenation of message to key, i.e., H(k||m), secure?

- Answer: No. Concatenation attacks are possible due to the construction of some hash functions (revisit MD5 and the Merkle-Damgard construction)



HMAC

- Simple and secure
- The application of a hash function twice with an inner-padding (ipad) and outer-padding (opad)
- ipad is B blocks of 0x36 and opad is B blocks of 0x5C, where B is the byte size of the block to be processed (e.g., B=64 in case of MD5 that uses blocks of 512bits)

$$HMAC(K,m) = H((K \oplus \text{opad}) \,||\, H((K \oplus \text{ipad}) \,||\, m))$$

- Can be paired with any hash function, e.g., HMAC-MD5, HMAC-SHA256, etc.
- NMAC (Nested MAC) is as simple as HMAC, however it requires changing the IV which is less handy when implementing

Various paradigms of combining MACs with encryptions

- A frequent application of MAC functions is in authenticated encryption, i.e., assuring that an encrypted ciphertext indeed originates from the source (note that block ciphers are not designed for this)
- There are three paradigms employed in practice:
  - **Encrypt-and-MAC**, i.e., $E_k(m)||MAC_k(m)$, used in SSH
  - **MAC-then-encrypt** , i.e., $E_k\big(m||MAC_k(m)\big)$, used in SSL/TLS
  - **Encrypt-then-MAC**, i.e., $E_k(m)||MAC_k\big(E_k(m)\big)$, used in IPSec
- **Encrypt-then-MAC** has better security than the previous two and should be the desired alternative in practice
- For details, see Bellare & Namprempre, "Authenticated Encryption: Relations among notions and analysis of the generic composition paradigm", 2000

Type of functions (IV) RNGs and PRNGs

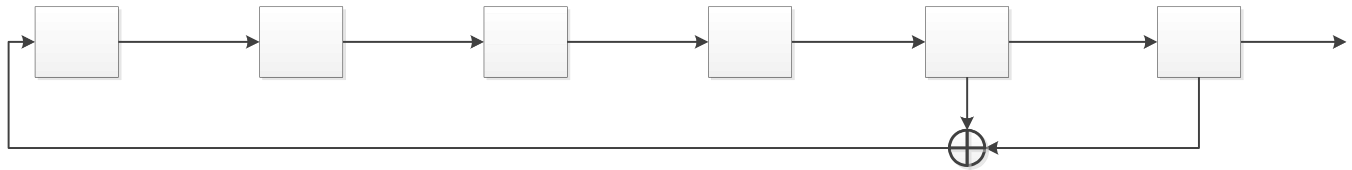
- Random numbers stay at the core of any cryptosystem since you need randomness for the secret keys
- Description (informal):
  - **TRNG** – True random-number generators output random sequences based on physical processes that are hard/infeasible to model, i.e., white noise from a Zenner diode, oscillator drift, SRAM state at power-up, etc.
  - **PRNGs** – deterministic algorithms that generate a random sequence based on a value called seed (they all have cycles but this does not mean they are insecure, computationally secure PRNGs exist)
- Example of use: used in any handshake SSL/TLS, IPSec, etc. that needs to generate a fresh session key

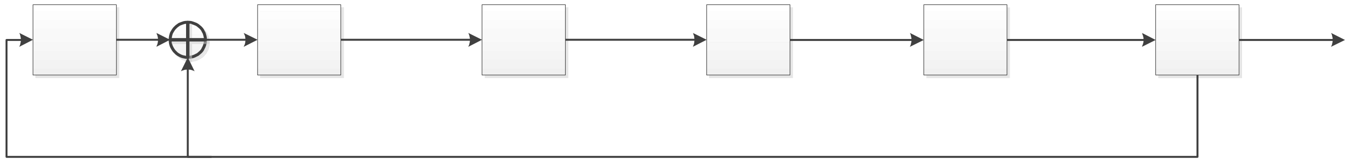
PRNG examples

- The linear congruential generator, an insecure and yet common solution

$$X_{i+1} = aX_i + c \bmod n \quad (X_0 \text{ is the seed})$$

- Galois or Fibonacci LFSR (Linear Feedback Shift Register) are another common, insecure alternative





- Bloom-Bloom-Shub is cryptographically secure but requires a large modulus n and is computationally expensive, thus almost absent in practice (X_0 is the seed)

$$X_i = X_{i-1}^2 \bmod n \quad (X_0 \text{ is the seed})$$

- Block ciphers in counter mode or stream ciphers provide secure instantation of PRNGs (as long as the cipher is secure)

How to test RNG & PRNGs

- Various statistical tests are usually employed, none is perfect but may provide some degree of confidence
- Dieharder is a battery of tests used by many enthusiasts or professionals http://www.phy.duke.edu/~rgb/General/rand_rate.php

Questions?