PV181 Laboratory of security and applied cryptography

Symmetric cryptography

Marek Sýs, Zdeněk Říha

CROCS
Centre for Research on Cryptography and Security

Goals of Cryptography

- Confidentiality (privacy) - preventing open access
  - **ciphers**
- Authentication:
  1. Entity – identity verification – various (password, MAC, …)
  2. Data origin – identity of message originator – **MAC**
- Integrity - preventing unauthorized modification
  - **hash functions**
- Non-repundation - preventing denial of actions
  - **digital signature**

Crypto primitives

- **Ciphers** – encryption/decryption of data using **key**
  - Symmetric ciphers – **same** key for enc/dec
  - Asymmetric ciphers – **different** key for enc/dec
- **Random number generators** (RNGs)
  - generation of Keys, IVs, Nonces, …
- **Hash functions** – “unique” fingerprint of data

- Based on previous: MAC, PBKDF, Digital signature

Standards

Primitives are defined in various types of standards:

- FIPS PUB 197 – AES block cipher
- RFC1321 – md5 hash function
- NIST SP,…

Test vectors: defined output to test implementation

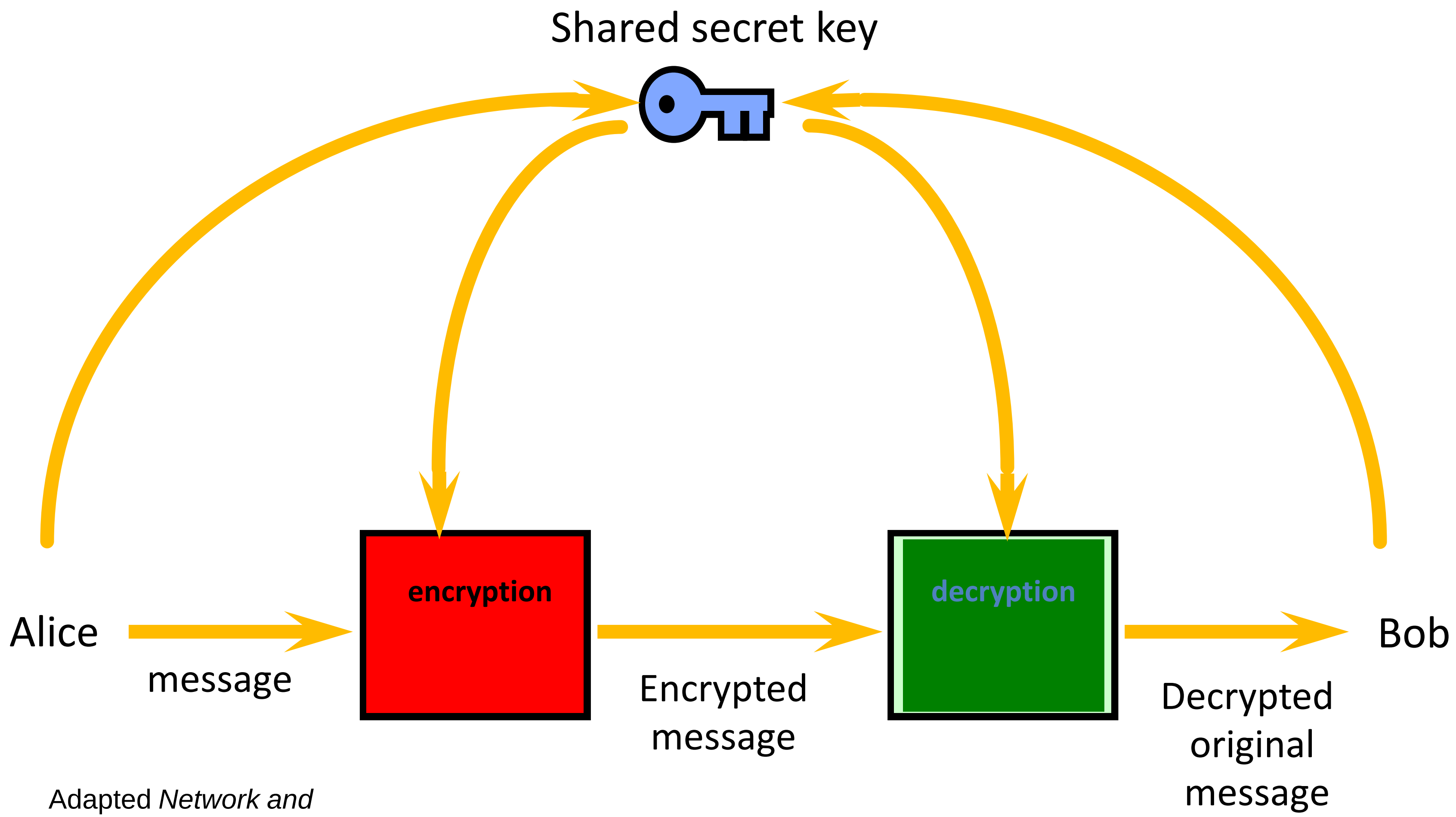
- MD5 ("") = **d41d8cd98f00b204e9800998ecf8427e**

Ciphers: Kerckhoffs' principle

- A cryptosystem should be secure even if everything about the system, except the **key**, is public knowledge.

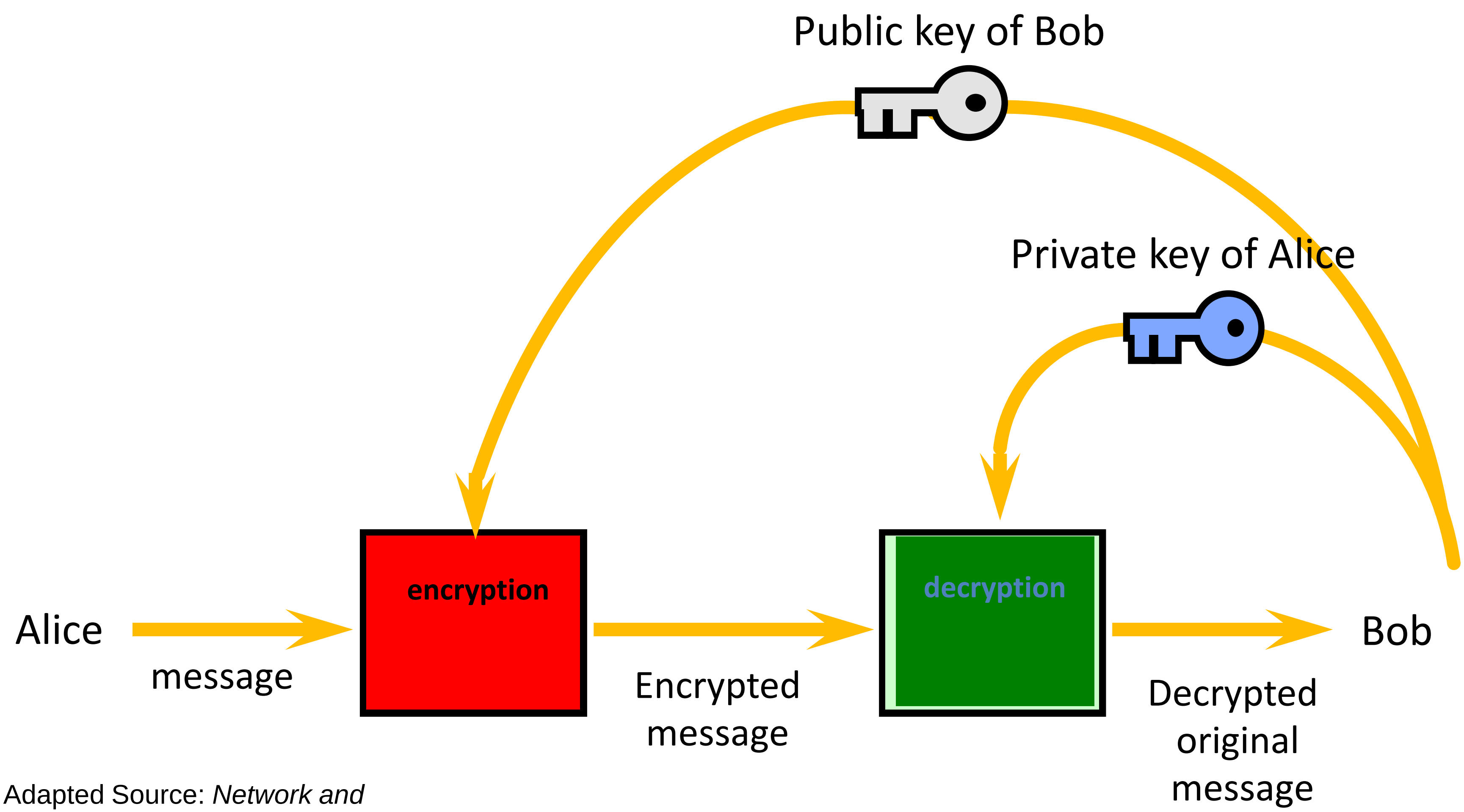
- I.e. only the **key** should be kept secret, not the algorithm.

Symmetric cryptosystem



Adapted *Network and Internetwork Security* (Stallings)

Asymmetric cryptosystem

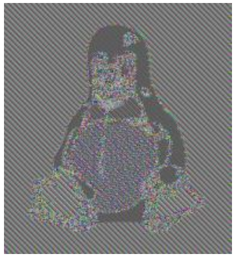


Adapted Source: *Network and Internetwork Security* (Stallings)

Block cipher

- Input divided into blocks of fixed size (e.g 256 bits)
  – Padding - message is padded to complete last block

- Different modes of operation:
  – Insecure basic ECB mode – leaks info
  – Secure modes: CBC, OFB,CFB,CTR,…



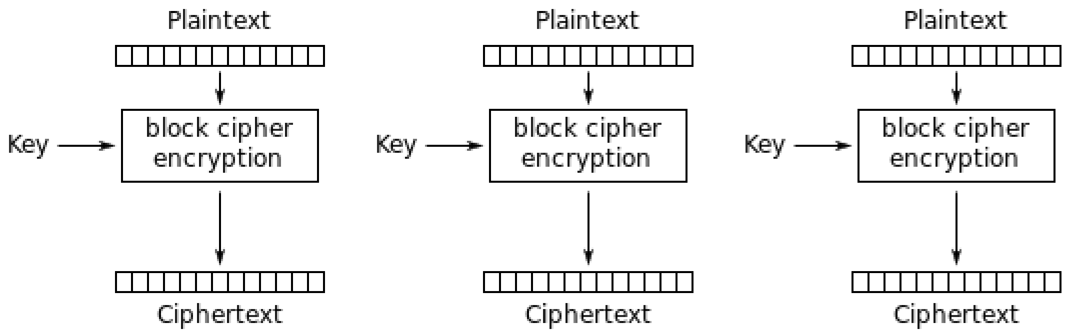
- CBC, OFB,CFB need initialization
  – Initialization vector (IV) – must be known

Source: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

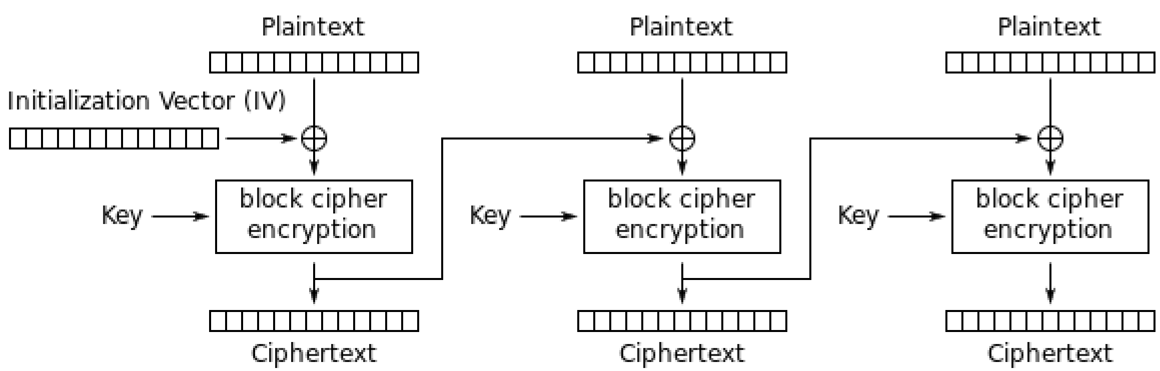
Block ciphers - padding

Standard	*method*
ANSI X.923	`... | DD DD DD DD DD DD DD DD | DD DD DD DD` **`00 00 00 04`** `|`
ISO 10126	`... | DD DD DD DD DD DD DD DD | DD DD DD DD` **`81 A6 23 04`** `|`
PKCS7	`... | DD DD DD DD DD DD DD DD | DD DD DD DD` **`04 04 04 04`** `|`
ISO/IEC 7816-4	`... | DD DD DD DD DD DD DD DD | DD DD DD DD` **`80 00 00 00`** `|`
Zero padding	`... | DD DD DD DD DD DD DD DD | DD DD DD DD` **`00 00 00 00`** `|`

Block ciphers: ECB vs CBC mode



Electronic Codebook (ECB) mode encryption



Cipher Block Chaining (CBC) mode encryption

Source: https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

Random number generators

- Used to generate: keys, IV, …

1. Truly RNG - physical process
   - aperiodic, slow
2. Pseudo RNG (PRNG) – software function
   - deterministic, periodic, fast
   - initialized by **seed** – fully determines random data

- Combination often used:
  - truly RNG used to generate **seed** for PRNG
  - dev/urandom, dev/random in Linux, **Fortuna** scheme

Hash function

- **Cryptographic** hash function
- Input of arbitrary size
- Output of fixed size: n bits (e.g. 256 bits).
- Function is not injective (there are “collisions”).
- Hash is a compact representative of input (also called imprint, (digital) fingerprint or message digest).
- Hash functions often used to protect integrity. First the hash is computed and then only the hash is protected (e.g. digitally signed).

Hash function properties

- One-way property
  - It is easy to calculate **h(x)** for arbitraty **x.**
  - In a reasonable time it is not possible for the fixed **y** to find **x**, such that **h(x) = y**.
- Collision resistance
  - (**weak**): In a reasonable time it is not possible for a given **x** to find **x’** (**x ≠ x’**) such that **h(x) = h(x’)**.
  - (**strong**): In a reasonable time it is not possible to find any **x, x’** such that **h(x) = h(x’)**.

Cryptographic hash functions

- MD5: output 128 bits
  - Still used although not considered secure at all
  - Broken: efficient algorithm for finding collisions available
  - 128-bit output not considered secure enough
- RIPEMD
  - Output : 128, 160, 256 or 320 bits
  - Less frequently used
- Whirlpool
  - Output: 512 bits
  - Based on AES
  - Recommended by NESSIE project
  - Standardized by ISO

Secure Hash Algorithm (SHA)

- **SHA-1**
  - NIST standard, collision found in 2016, 160 bits hash
- **SHA-2**
  - function family: SHA-256, SHA-384, SHA-512, SHA-224
  - defined in FIPS 180-2
  - Recommended
- **SHA-3**
  - New standard 2015
  - Keccak sponge function family: SHAKE-128, SHA3-224, …
  - defined in FIPS 202, used in FIPS-202, SP 800-185
  - Recommended

Hash functions - examples

- MD5
  - Input: „Autentizace“.
  - Output: 2445b187f422458303788851 1d5411c7 .
  - Output 128 bits, written in hexadecimal notation.
  - Input: „Cutentizace“.
  - Output: cd99abbba3306584e90270bf015b36a7.
  - A single bit changed in input → big change in output, so called “Avalanche effect”
- SHA-1
  - Input: „Autentizace“.
  - Output: 647315cd2a6c953cf5c29d36e0ad14e395ed1776
- SHA-256
  - Input: „Autentizace“.
  - Output: a2eb4bc98a5f71a4db02ed4aed7f12c4ead1e7c98323fda8ecbb69282e4df584

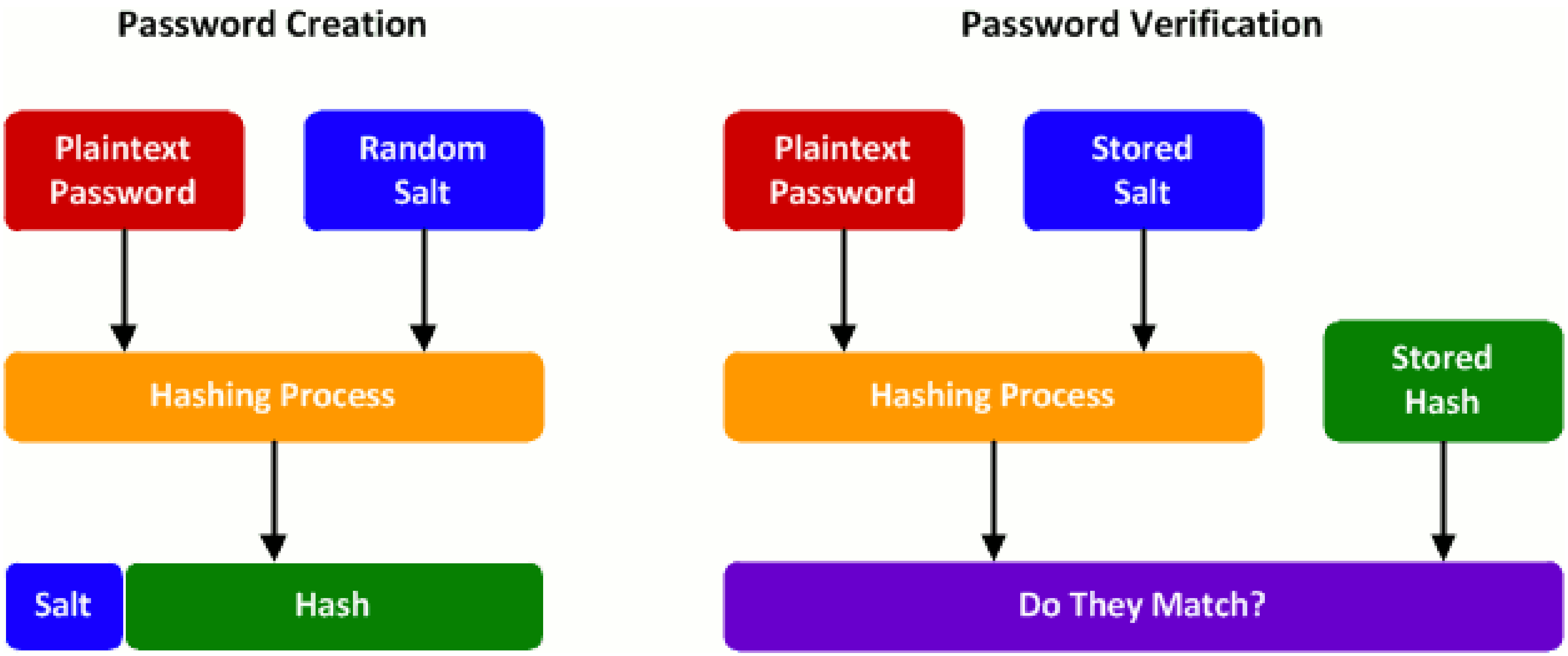
Password protection
password hashing & salting

1. Clear password could be stolen:
   - store hash of password

     hash = H(password)
   - Checking: password is correct if **hash** matches
2. Attack (brute force or dictionary)
   - trying possible passwords “aaa”, “aab”…“zzz” – N tests
   - N test for single but also for 2,3,… passwords **!!!**
3. Salt - random string (salt) added to password

   hash = H(salt | password)
   - protects many passwords not one (salt also stored)

Password protection
password hashing & salting



Source: http://blog.conviso.com.br/worst-and-best-practices-for-secure-password-storage/

Key protection

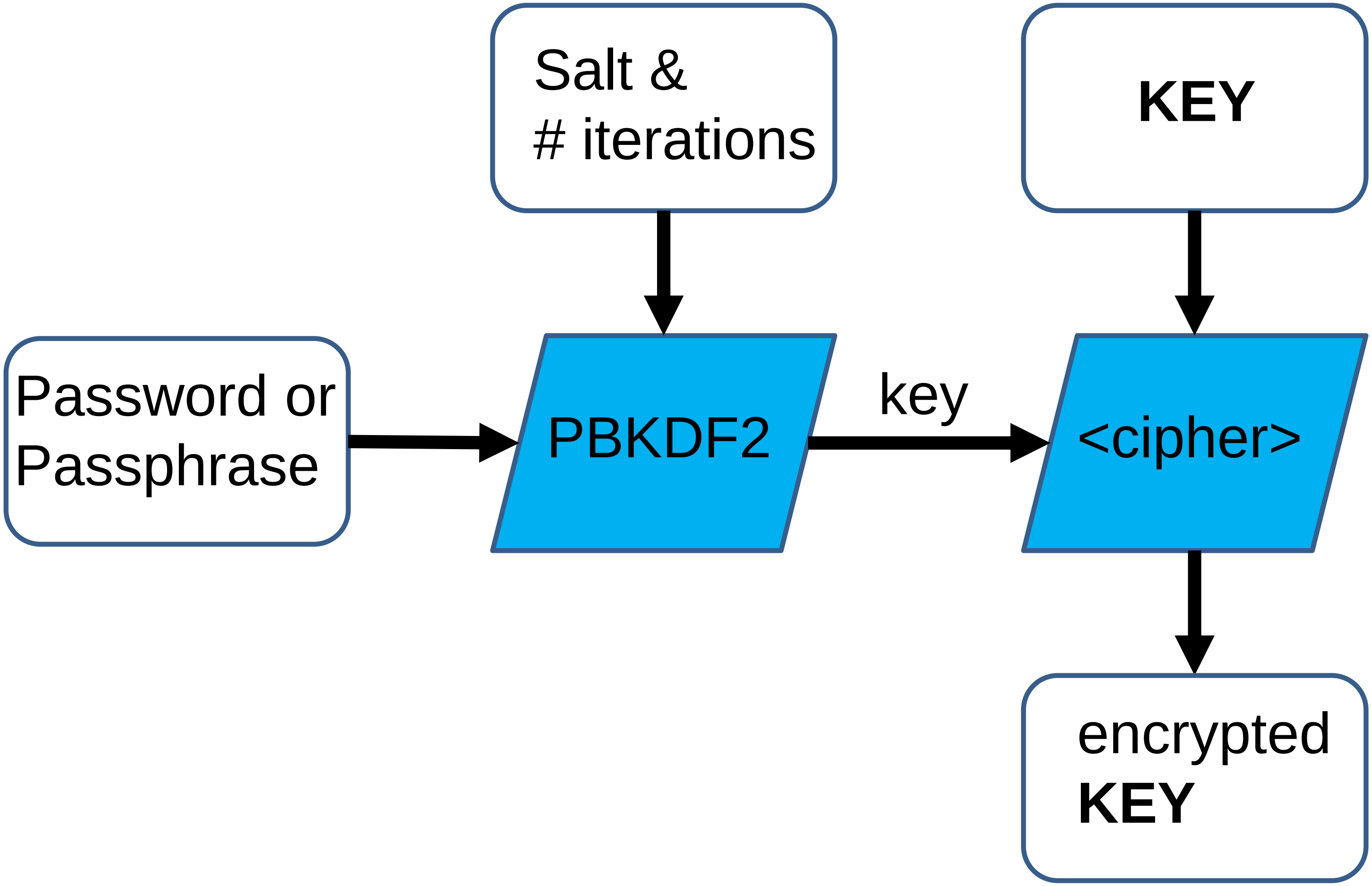
- Encrypt **key** (using cipher and other key $\mathbf{k}$)
  - Key $\mathbf{k}$ typically derived from password

4. Password based key derivation function (PBKDF):
   - 2 types - PBKDF and newer PBKDF2 (PKCS#5)
   - slow down hashing of passwords – hash of hash of hash…

$$\mathbf{k} = \boldsymbol{H^c(salt \mid pwd)}$$

   - Attacker is c times slower / need c times more resources

PBKDF2


Salt &
iterations
KEY
Password or
Passphrase
PBKDF2
key
<cipher>
encrypted
KEY

MAC

- based on block cipher or hash func. (HMAC)
- MAC = **authenticity** + integrity
  - message authenticity – came from stated sender
  - message integrity - was not altered
  - Key + message → algorithm → MAC