PV181 Laboratory of security and applied cryptography

Symmetric cryptography

Marek Sýs, Zdeněk Říha

CROCS
Centre for Research on Cryptography and Security

Cryptography - brief overview

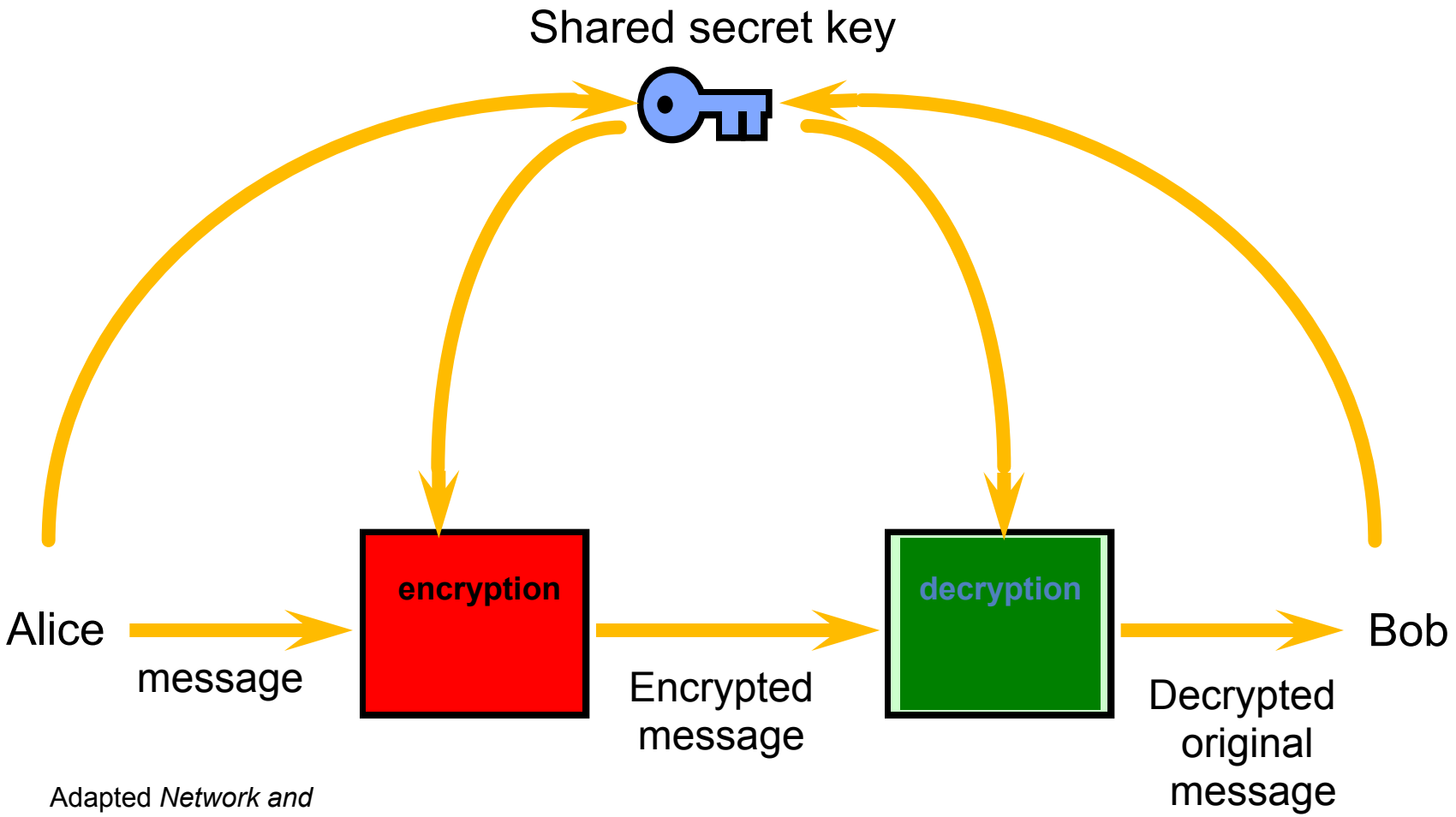
Goals of Cryptography

- 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

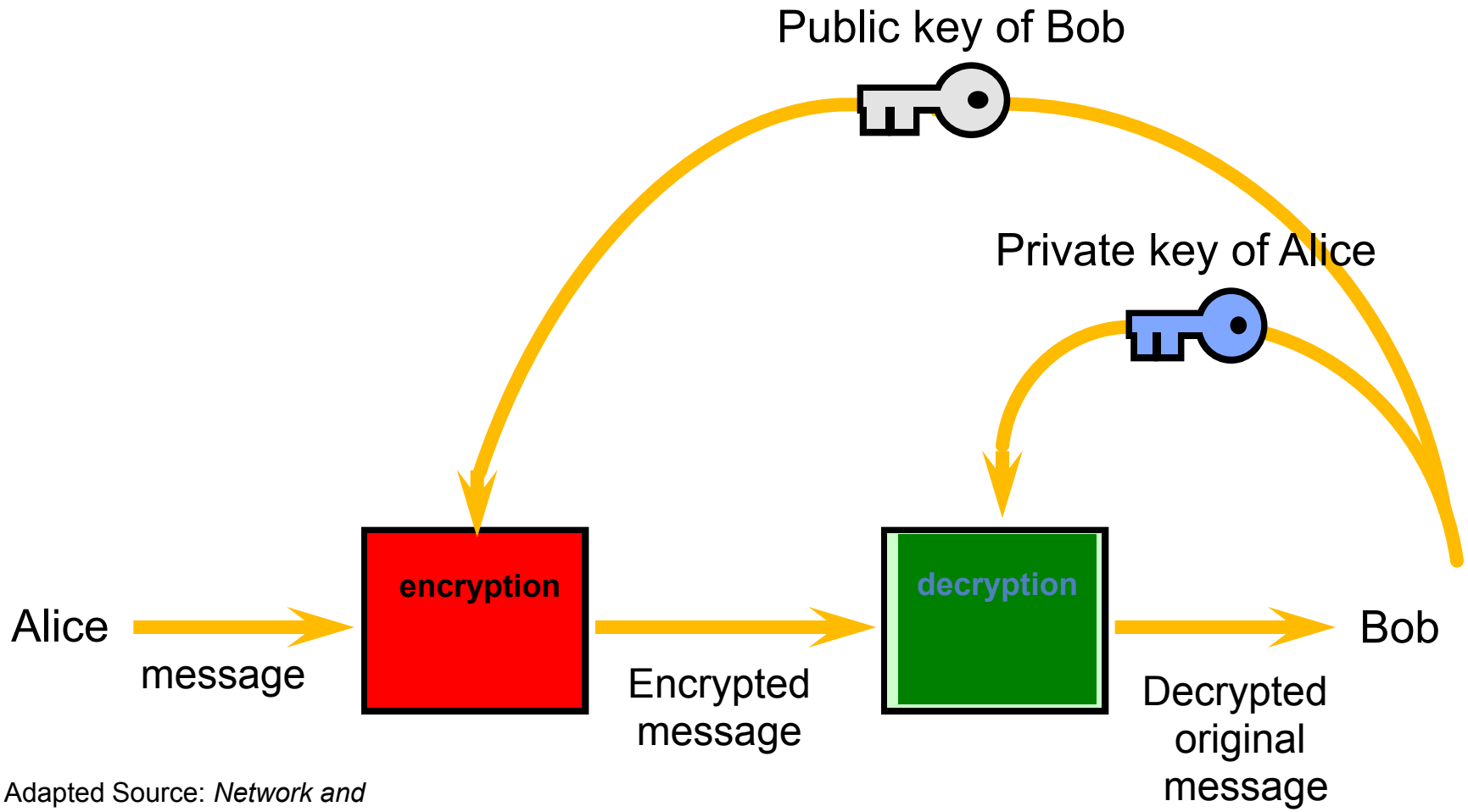
- **Ciphers** – encryption/decryption of data using **key**
  - Symmetric ciphers – **same** key for enc/dec
  - Asymmetric ciphers – **different** key for enc/dec
- **Random number generators** (RNGs)
  - Key generation
- **Hash functions** – “unique” fingerprint of data
- Based on previous: MAC, PBKDF, Digital signature

Symmetric cryptosystem



Adapted *Network and Internetwork Security* (Stallings)

Asymmetric cryptosystem



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

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 has is computed and then only the hash is protected (e.g. digitally signed).

Standards

Everything defined in standard:

- implementation, settings, usage, etc.
- **If you need something look into standard**

Different types:

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

Implementation testing - test vectors

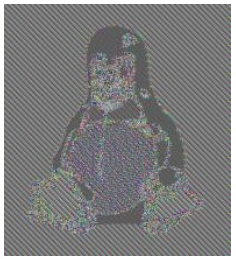
- Examples of input/output (and also intermediate) for the reference implementation
- MD5 defined in RFC 1321:
  - MD5("") = d41d8cd98f00b204e9800998ecf8427e
  - MD5("message digest") = f96b697d7cb7938d525…
- AES defined in FIPS197:
  - Plaintext: 00112233445566778899aabbccddeeff
  - Key 000102030405060708090a0b0c0d0e0f
  - Ciphertext 69c4e0d86a7b0430d8cdb78070b4c55a

Symmetric cryptography

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,…



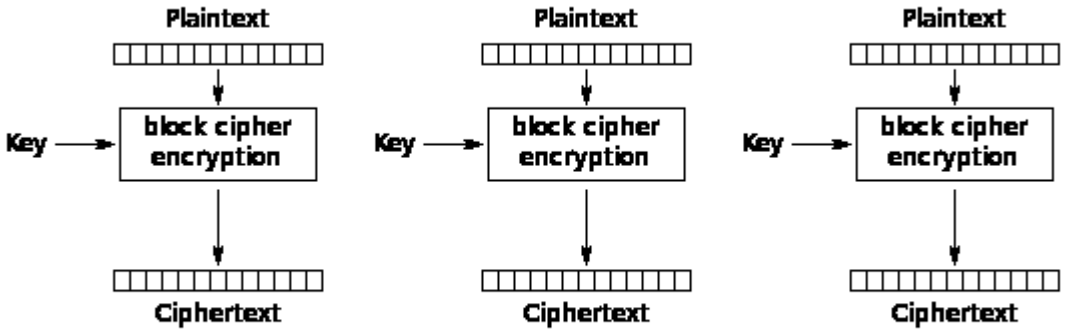
- CBC, OFB,CFB need initialization
  - Initialization vector (IV) – must be known

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

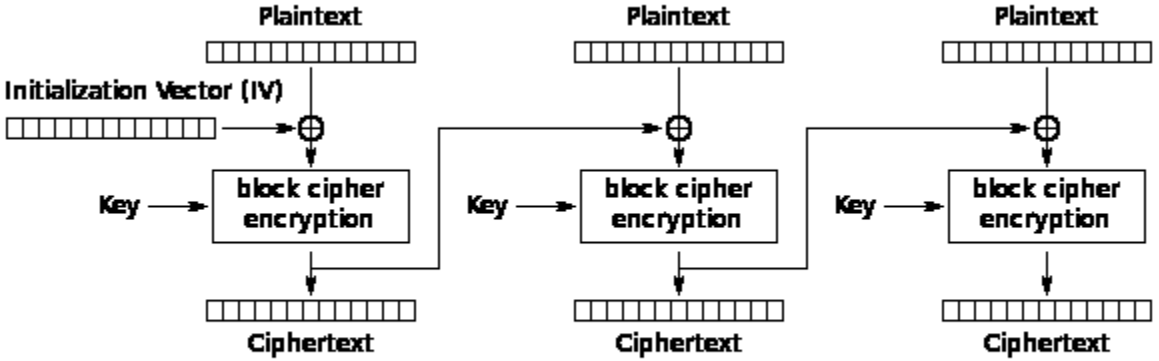
Block ciphers - padding

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

Hash functions - examples

- MD5
  - Input: „Autentizace“.
  - Output: 2445b187f4224583037888511d5411c7 .
  - 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

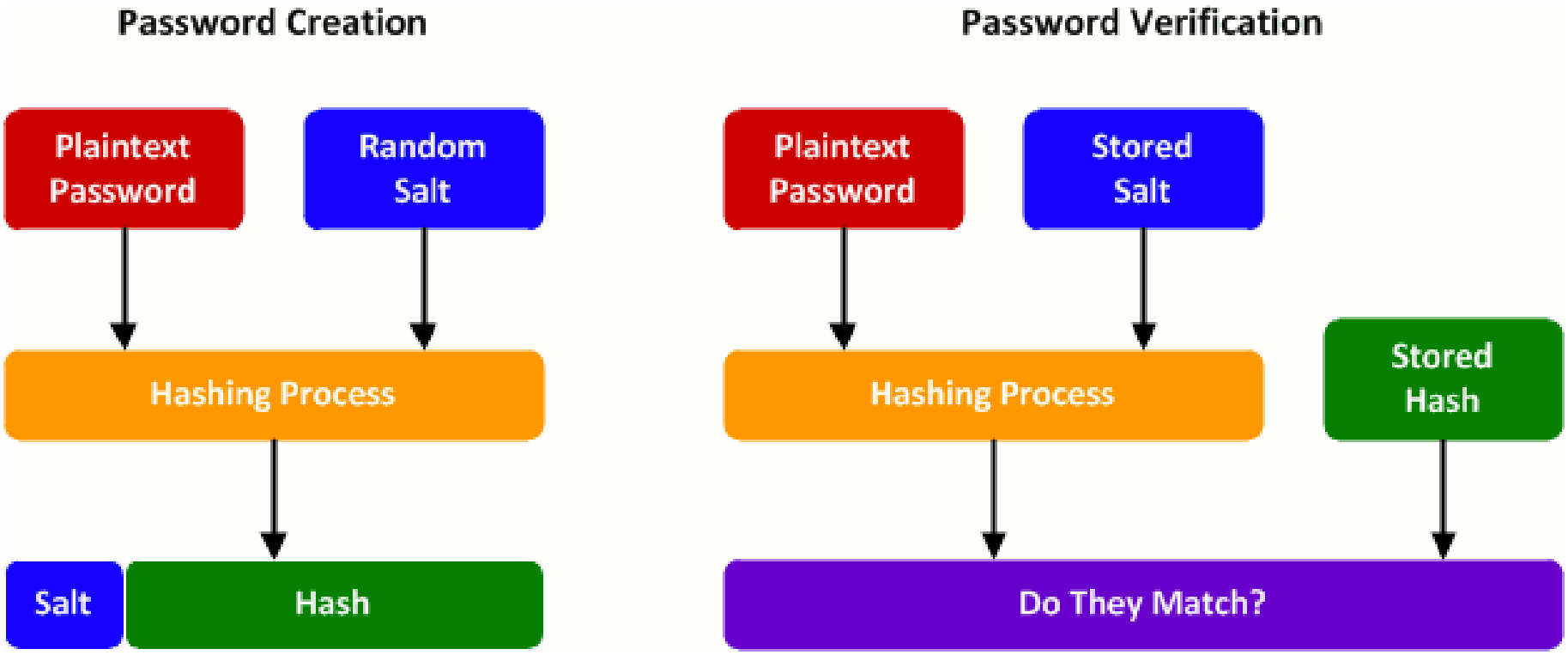
Secure Hash Algorithm (SHA)

- **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
  - Recomended

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. Slow down attack - increase password size:
   - random “**salt**” is added to password,

Password protection
password hashing & salting

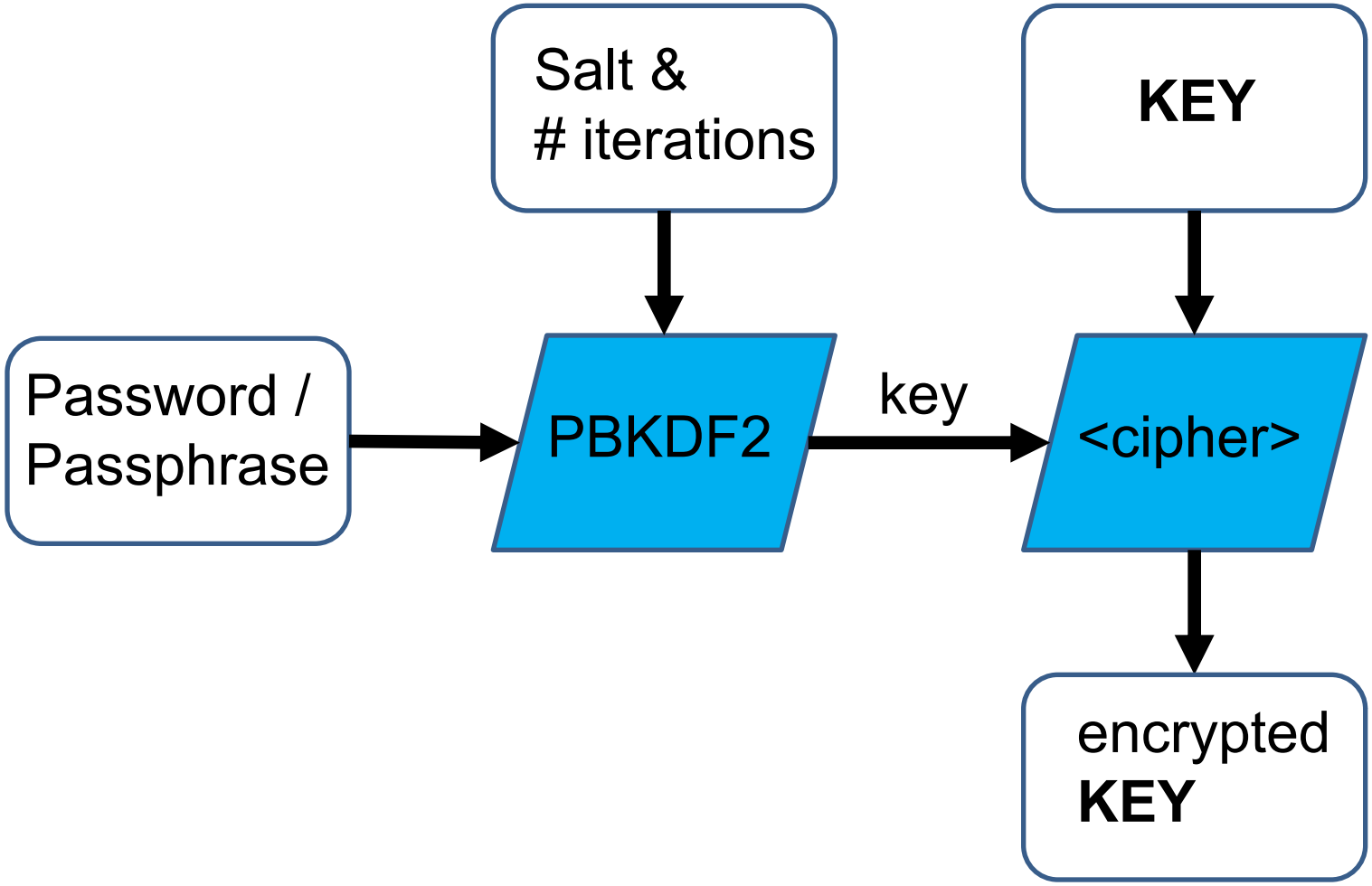


Source: http://blog.conviso.com.br/worst-and-best-practices-for-secure-password-storage/

Key/password protection

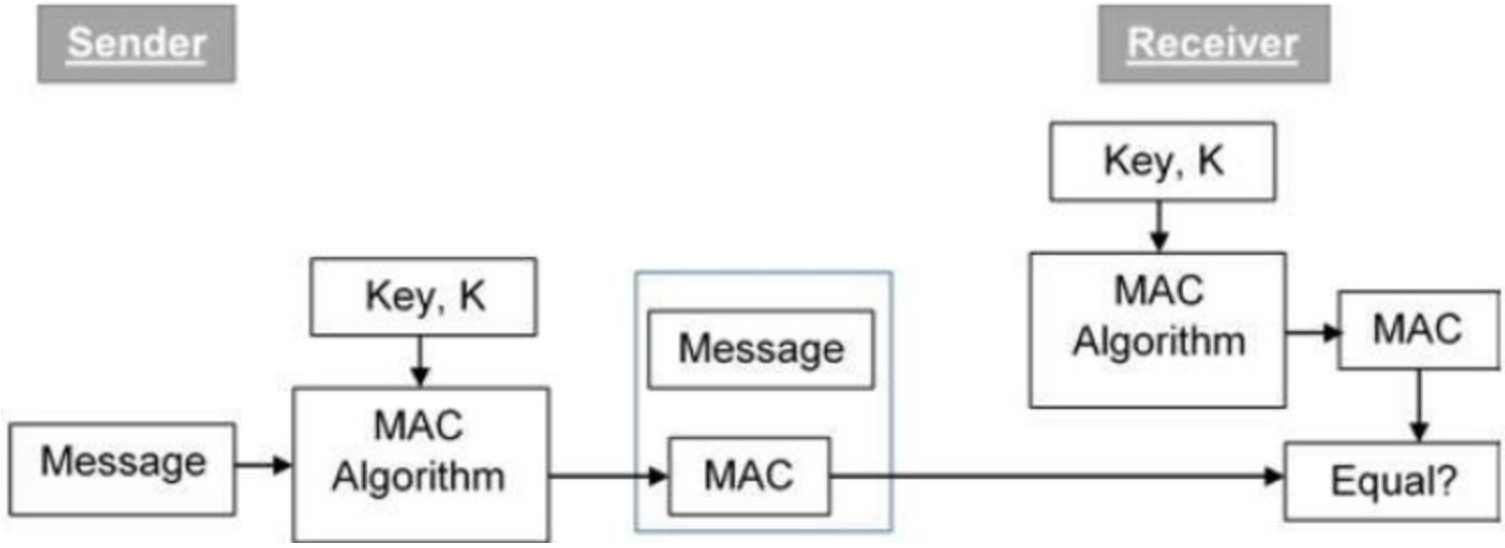
- Encryption (session) key – encrypted using other key – could be derived from password
- Insufficient entropy of passwords
  - Only 200 millions of guesses for ****** password
  - salt protects database of passwords not single password (more info here)
- Password Based Key Derivation Function(PBDKF):
  - 2 types PBDKF2 is newer (see PKCS#5)
  - Slow down hashing (hence attack)
  - Iterate (c times) hash function $K = H^c(pwd \mid salt)$

PBKDF2


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

Message authentication code (MAC)

- – Based on block cipher (MAC) or hash function (HMAC)
  – Key + message → algorithm → fixed size block MAC



Source: https://www.tutorialspoint.com/cryptography/message_authentication.htm

Instructions for the seminar

OpenSSL access

- Unix – direct (includes already OpenSSL)
- Windows:
  - Use **aisa** faculty server with Unix (recommended) – as it will be used also within other seminars
  - or install to your computer
- Connect to aisa via putty:
  1. Host aisa.fi.muni.cz → Open
  2. Prompt: Accept the key (only once)
  3. Login as: your UCO x… + Secondary password

OpenSSL materials

- Books:
  - free ebook OpenSSL cookbook

- Useful links:
  - Manual: https://www.openssl.org/docs/man1.0.2
  - https://wiki.openssl.org/index.php/Command_Line_Utilities
  - https://www.madboa.com/geek/openssl/

OpenSSL basics

- Usage:
  - With cmd history(prefered): openssl commands params
  - Without:
    - openssl (enter)
    - commands params (without openssl)
- Typical problems:
  - order of parameters
  - string: str or “str” (may represent string of 5 chars)
- Help = non-recognized command
  - e.g. blablabla or help

Symmetric crypto

OpenSSL commands:

- dgst – hashing, MAC, signature
- enc – encryption, base 64 encoding,…

- Hashed, encrypted, … data are not readable (not ASCII characters) hence we will use rather:
  - hex – byte encoded 2 chars from charset (0-9, A-F)
  - or base64 – 6 bits encoded by 1 char (A-Z, a-z, 0-9, +, /)
    - Padding is used (no padding, “=“, “==“)

Practical part

- See instructions in Tasks01.txt in IS