COMPUTER SECURITY

Cryptography: more advanced topics

Long[1] texts' encipherment: operation modes

Base method

- divide *P*, plaintext, in parts of equal size (blocks) $P = P_1 P_2$ ...
  - pieces could be of 1 b, 1 B, 8 B (typical), 16 B (typical)...
- encipher each part separately by one of the methods:
  - stream[2]
    - each part uses a different key: $C = E_{K1}(P_1)\ E_{K2}(P_2) \ldots$
      - or, for simplicity: $C = K_1(P_1)\ K_2(P_2) \ldots$
    - in practice, encipher function usually is (bitwise) plain XOR, $\oplus$!
  - block
    - each part uses same key: $C = K(P_1)\ K(P_2)$ ...
  - “mix” of previous
    - same key for each part acts as successive different key

1 See that, in practice, almost any text is "long"! At least regarding symmetric cryptography.
2 PT: contínuo, sequencial

Rationale for "operation modes"

- stream
  - Pro: most secure (even, provable secure with *One-time pad*)
  - Con: very long, one-time usable (random) key
- block
  - Pro: single (random) key
  - Con: same plaintext, same ciphertext
    - if $P_1 = P_2$, then $C_1 = C_2$ [FIG]
- mixed
  - Pro: single (random) key
  - Con: added complexity; possible vulnerability to undetectable modifications of ciphertext!

a) 

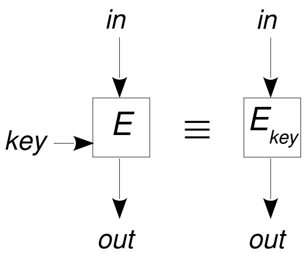
b) 

Fig. a) original picture;
b) enciphered with AES 256b,
ECB mode

Some operation modes

Stream method

- Most common: $E = D =$ plain XOR, $\oplus$
  - e.g. $C_1 = P_1 \oplus K_1$; $P_1 = C_1 \oplus K_1$
- Observation:
  - K_i should be random and not be reused
- Notation:



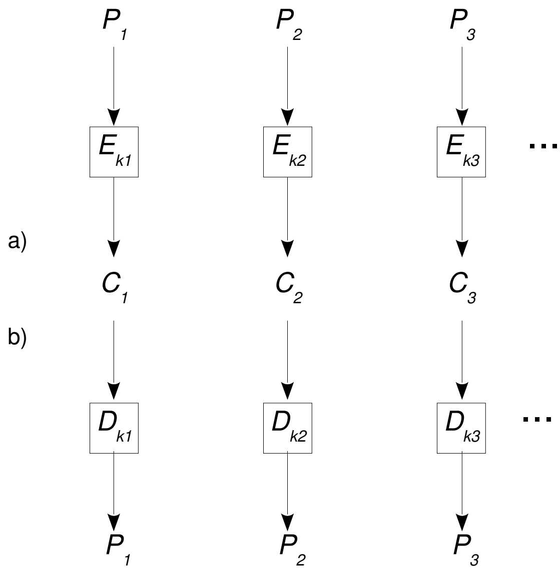


Fig. Use of plain stream method: a) enciphering; b) deciphering.

Block method

- *ECB, Electronic Code Book*
- Some properties:
  - padding of last block
  - parallelizable en/deciphering
- Formulas:
  - $C_i = E_k(P_i)$, $i>0$
  - Write the decipherment formula. :-)

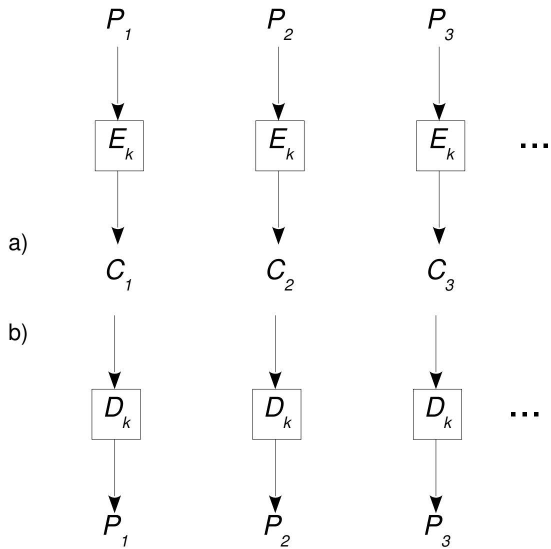


Fig. Use of (plain) block method: a) enciphering; b) deciphering.

"Mix" method: CTR

- *CTR, Counter Mode*
- Some properties:
  - *IV*[1], initialization vector (random+counter)
  - no padding
  - parallelizable en/deciphering
- Formulas:
  - Write the formulas for the encipherment (*C* as function of *P*) and vice versa.

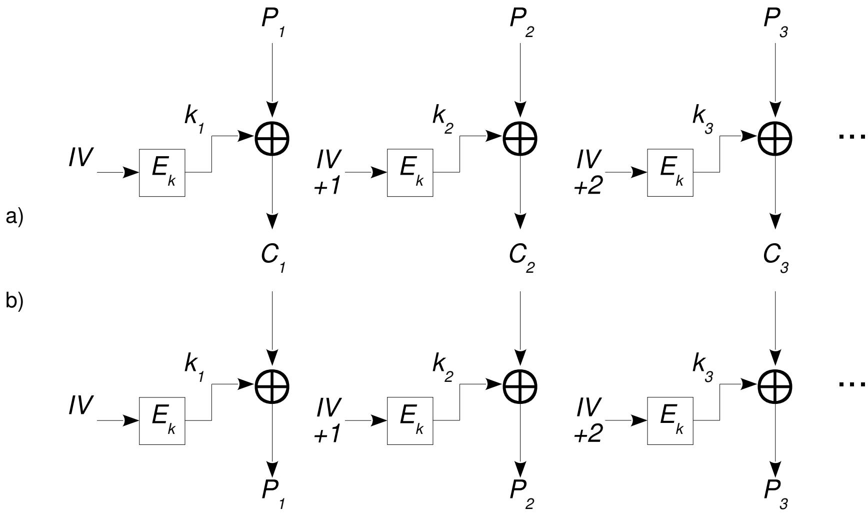


Fig. Use of "mixed" method CTR: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

1 public value that, as a rule, should be random

“Mix” method: CFB

- *CFB, Cipher Feedback*
- Some properties:
  - *IV*, initialization vector
  - no padding
  - not parallelizable enciphering; parallelizable deciphering
- Formulas:
  - $C_0 = IV$;
    $C_i = P_i \oplus E_k(C_{i-1})$, $i>0$
  - Write the decipherment formula.

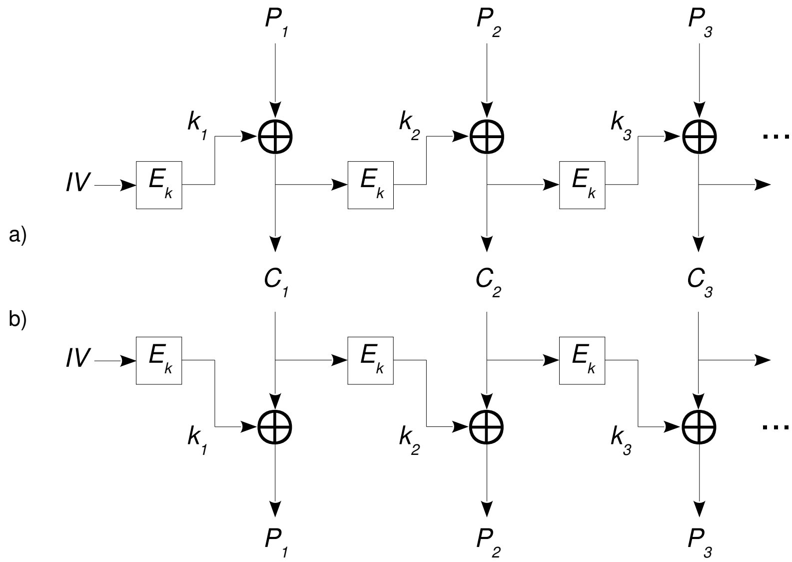


Fig. Use of “mixed” method CFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

“Mix” method: OFB

- *OFB, Output Feedback*
- Some properties:
  - *IV*, initialization vector
  - no padding
  - not parallelizable en/deciphering, but successive $E_k^i(IV)$ can be done in advance
- Formulas:
  - $C_i = P_i \oplus E_k^i(IV)$, $i \geq 0$
  - Write the decipherment formula.

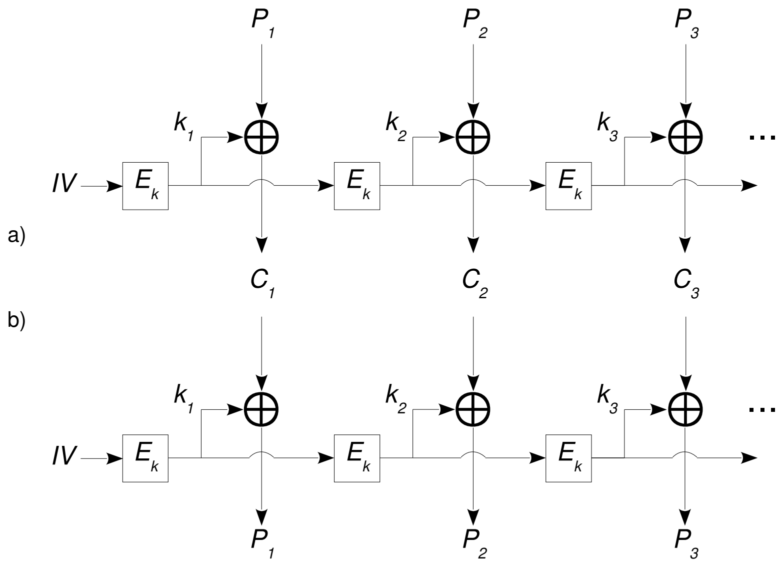


Fig. Use of “mixed” method OFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

“Mix” method: CBC

- *CBC, Cipher Block Chaining*
- Some properties:
  - *IV*, initialization vector or explicit initialization by (phony) 1st block!
  - padding
  - not parallelizable enciphering; parallelizable deciphering
- Formulas:
  - $C_0 = IV$; $C_i = E_k(P_i \oplus C_{i-1})$ $i>0$
  - Write the decipherment formula.

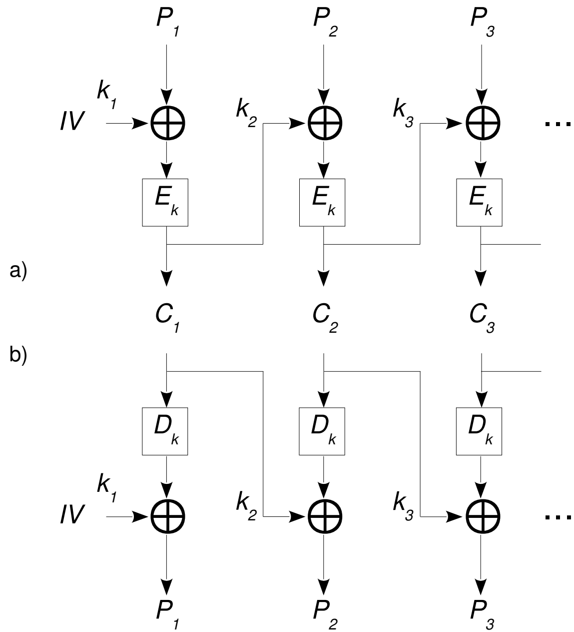


Fig. Use of “mixed” method CBC: a) enciphering; b) deciphering (Notice the virtual keys k_i.)

Shortcomings of "mixed" block methods:

- some "mixed" methods are vulnerable to modifications of ciphertext
- so, some type of integrity protection must be added to the confidentiality protection: authenticated encipherment modes (*see later*)

Padding

Need

- size of plaintext varies (just hardly ever is multiple of block size)
  - so, final block might need[1] padding!
  - but, "casual" padding might open an attack path (*see ahead*)!
- harden message deciphering and traffic analysis[2]
  - by obscuring the size (and content) of ciphertext
    - e.g. avoiding short messages' attack on RSA[3]
    - e.g. avoiding deterministic ciphering's attack[4]

1 some "modes of operation" do not need padding... why?
2 interception and examination of (ciphered or not) communications to deduce information (e.g. from patterns)
3 asecuritysite.com/encryption/crackrsa2
4 en.wikipedia.org/wiki/Deterministic_encryption

..."Long" texts' encipherment: Padding...

Padding schemes

- several schemes (bit padding or, more usually, byte padding)
  - shared-key cryptography
    - e.g. PKCS[1] #5[2], #7[3] (enciphering) [FIG]
  - one-way cryptography
    - e.g. RFC 6234 (SHA-1, SHA-256) [FIG]
    - e.g. SHA3 (sponge) [FIG]
  - public-key cryptography
    - e.g. PKCS #1 v2 (RFC 8017)
      - RSA's PKCS1-v1_5 [FIG]
      - RSA's OAEP, Optimal Asymmetric Encryption Padding [FIG]
        - Exercise (after analyzing picture): what about deciphering?... does receiver need *seed* and *L*?...

1 Public Key Cryptography Standards, devised and published by RSA Security LLC since the 1990s
2 PKCS #5: Password-Based Cryptography - from a password, generate a (symmetric) key for a following symmetric encipherment.
3 #7 padding just extends 8B block #5 padding to 16B (128b) blocks

..."Long" texts' encipherment: Padding examples (figs)...

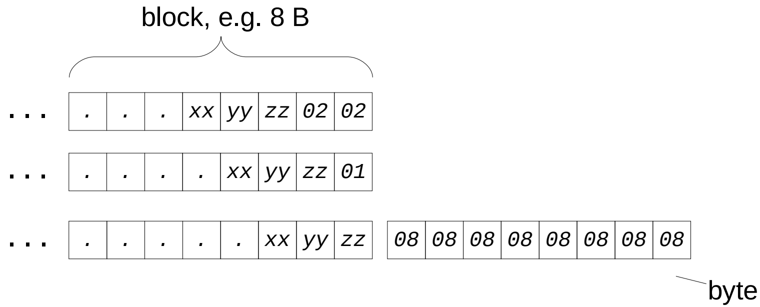


Fig: Shared-key cryptography padding: examples for PKCS #5 (8B blocks); #7 will be similar, but appropriate to 16B blocks.

...”Long” texts' encipherment: Padding examples (figs)...

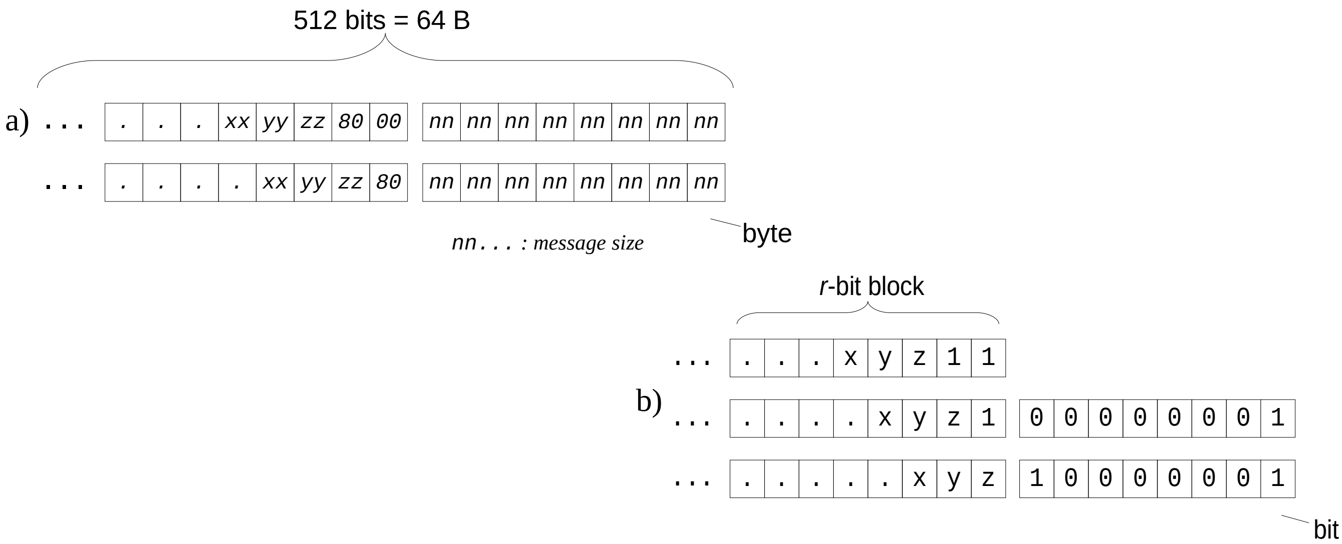
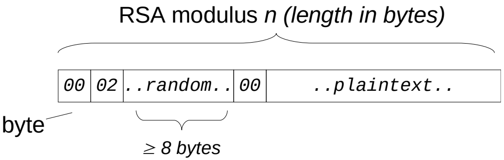


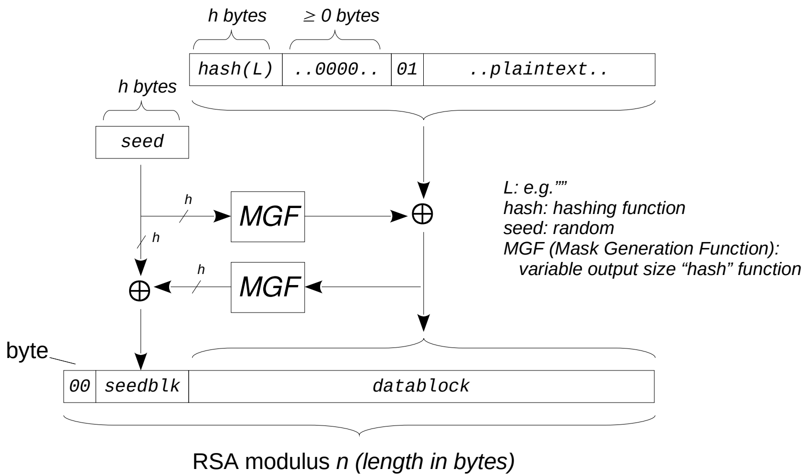
Fig: Instances of one-way cryptography padding:
a) RFC 6234 padding: (SHA1, SHA256...) - sequence of *nn*s is message size;
b) Sponge multirate padding: `10*1` (*r* is the number of bits of input block.

...”Long” texts' encipherment: Padding examples (figs)...



a)

Fig: RSA padding:
a) PKCS1-v1_5 ;
b) OAEP, Optimal Asymmetric Encryption Padding
(*L*, Label, can be empty string; *hash*: hashing function; *seed* must be random; *MGF*, Mask Generation Function, produces pseudorandom variable size strings).
After padding, RSA enciphering proceeds with final data being treated as of *n*-byte hex number.



b)

..."Long" texts' encipherment: Padding...

Attack examples

- length extension: one-way cryptography, MAC (if = $h(K||P)$)
  - if $hash(P1) = hash(IV, P1) = hash(hash(IV), P1)$
    $hash(P1||P2) = hash(P1, P2) = hash(hash(P1), P2)$
  - SEED Lab!
- padding oracle: two-way cryptography, CBC mode
  - if attacker can keep testing decipherment with crafted ciphertext
  - if deciphering error code says explicitly "*invalid padding*" instead of a general "*decryption failed*"
  - CBC: $P_i = D_k(C_i) \oplus C_{i-1} \quad i>0$
    - a byte/bit change in C_{i-1} affects corresponding byte/bit in P_i
    - starting from last C_i block (where padding is), keep changing last byte until padding is valid; then repeat for previous bytes
    - see [FIG] (PKCS #5, #7 padding)

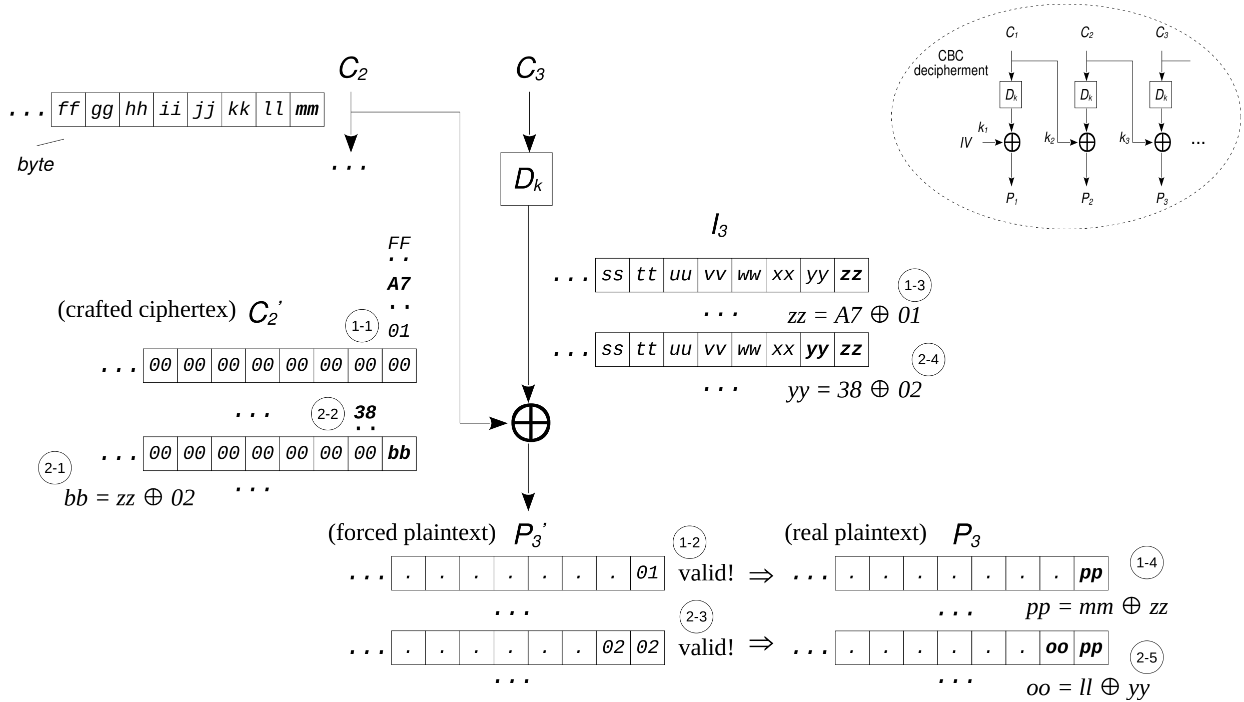
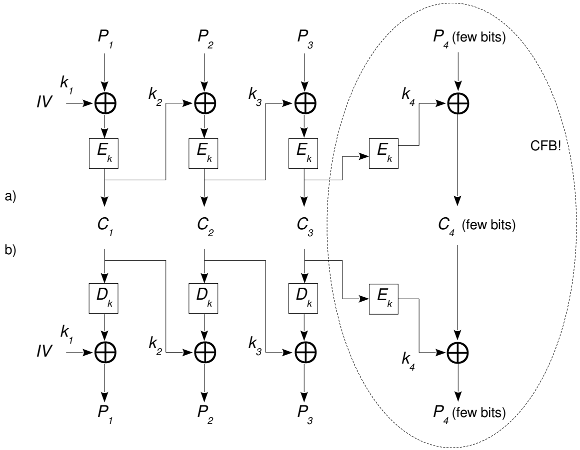


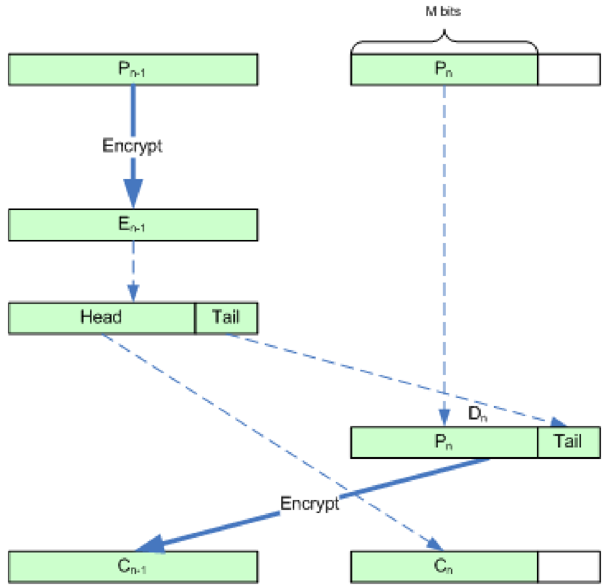
Fig. Padding oracle attack procedure for PKCS #5, #7 padding (CBC mode). C_3 is last cipher block.

..."Long" texts' encipherment: Padding...

Real need for padding?

- avoidance:
  - ciphertext stealing [FIG in Wikipedia]
  - residual block termination [FIG]
- will it be worth the trouble?...





One-way cryptography

Motivation

- «*Hash functions are everywhere in cryptography — everywhere!*»[1]

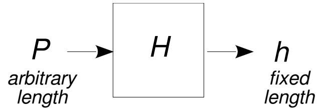
Applications of one-way functions

- data integrity protection
  - *P* public: *F* = *h*(*P*) is characteristic of *P* [2]
- confirmation of knowledge
  - *P* secret: publish *F* = *h*(*P*); later, when P is turned public, F proves previous knowledge of *P*
- key derivation
  - known *k1*, *k2* = *h*(*k1*) is new key that does not compromise *k1*!
- pseudo-random number generation
  - *seed* secret: h^n(*seed*) is apparently random for any successive *n*
- ...

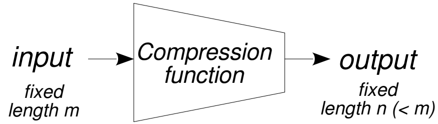
1 *Real-World Cryptography*, D. Wong, Manning, 2021
2 hash functions (unkeyed!) are also called MDC (Modification Detection Code) functions

Definitions[1]

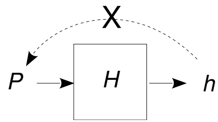
- (minimum) **hash** function *H* [2]



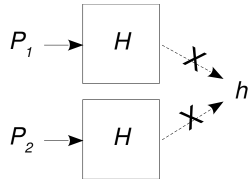
  - compression: maps input *P* of arbitrary finite bit-length, to output *h* of fixed bit-length
  - ease of computation: for any *P*
- **compression** function



  - hash function with fixed-size inputs
- **one-way** hash function



  - impractical[3] to invert function
- **collision-resistant** hash function



  - impractical to find two inputs with same output

1 *Handbook of Applied Cryptography*, A.J. Menezes et. al., 5th Printing, CRC Press, 2001
2 can use (secret) keys or not...
3 impractical = currently, computationally infeasible

Simple examples ($P = P_1 P_2 P_3 ... = P_1 \| P_2 \| P_3 ...$)

- (minimum) **hash** function (*in*, len(P) ; *out*, len(h))
  - $h = P_1 \oplus P_2 \oplus P_3 \oplus ...$, length (P_i) = length (h)
- **compression** function (*in*: m bits ; *out*: n bits)
  - *out* = (*in*'s first n bits) $\oplus$ (*in*'s last (m-n) bits || (2n-m) 0 bits)
- **one-way** hash function (*in*: m bits ; *out*: n bits)
  - $h = P \bmod h$
- **collision-resistant** hash function
  - ?...

Possible constructions of hash functions

- iterated hash functions (e.g. Merkle–Damgård construction)
  - block cipher based hash functions (e.g. Davies-Meyer construction)
    - using existing secure cipher functions
  - customized (e.g. SHA-1)
    - specifically designed “from scratch” for optimized performance
  - modular arithmetic based[1] (e.g. MASH-1)
    - quite few implementations as research interest is low:
      - sluggish relative to customized hash functions, «*embarrassing history of insecure proposals*» (Menezes et al.)
  - sponge constructions (e.g. SHA-3)
    - new paradigm, allowing easy adjustment of output length

1 ISO/IEC 10118-4:1998, Hash-functions using modular arithmetic

...One-way cryptography (cont.): Iterated hash functions - Merkle–Damgård construction

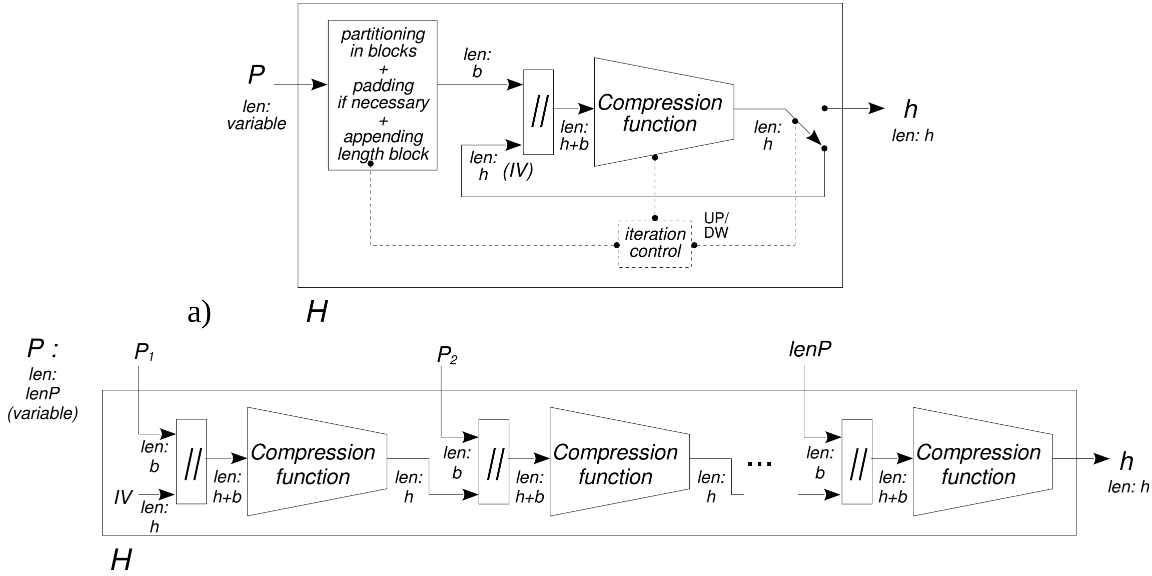


Fig. Two views of the Merkle–Damgård construction: a) software-view ; b) time-view.

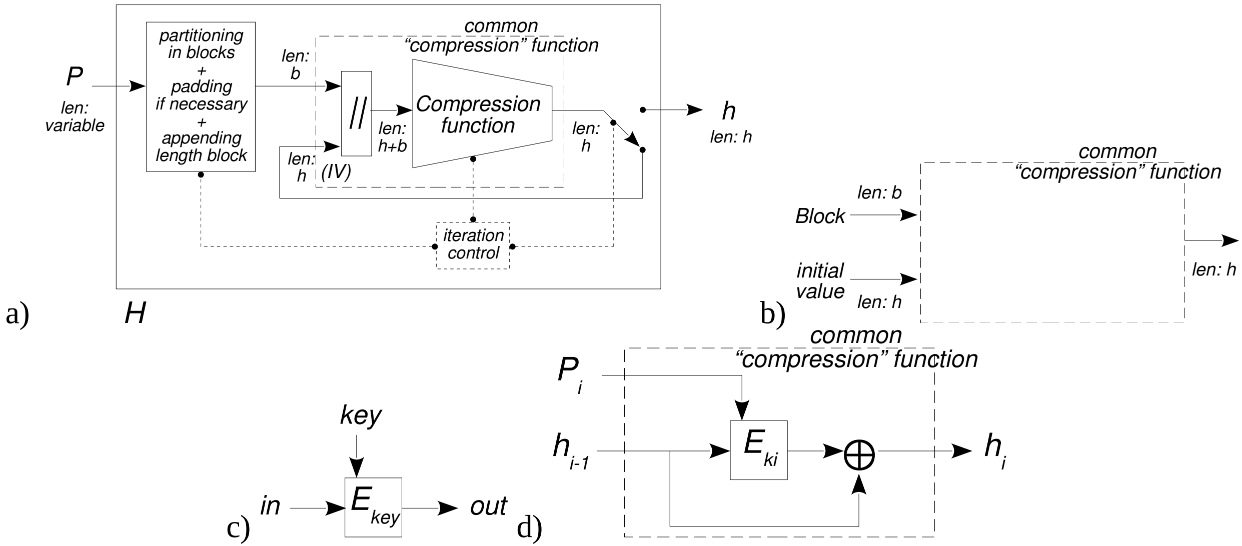
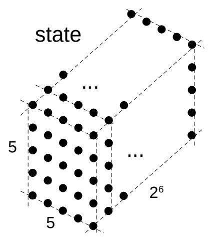
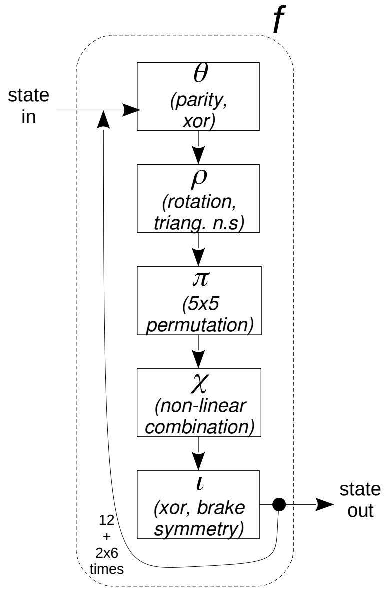


Fig. Davies-Meyer construction for <u>commonly seen</u> "compression functions": a) relation of "common compression" to here presented compression function; b) "compressed" function as block; c) general enciphering snippet - Note that if *in* is fixed, E_{key} is one-way for mapping *key* -> *out*!; d) Davies-Meyer construction.

Case study (simplified): SHA-3 (sponge construction)

- besides normal input *M*, another input parameter can specify length *l* of output *Z* [next page FIG]
- padding rule + function Keccak-*f*[1600][1], permutation of $b = r+c$ bits
  - state: $b = 5 \times 5 \times 2^6$ bits = 1600 bits
  - *r*: bits affected by input ; *c*: always internal bits
  - permutation: 12 + 2×6 rounds of five steps: θ ρ π χ ι





1 Keccak is pronounced as "ketchak" (keccak.team/keccak_specs_summary.html).

...One-way cryptography (cont.): SHA-3 (sponge construction)

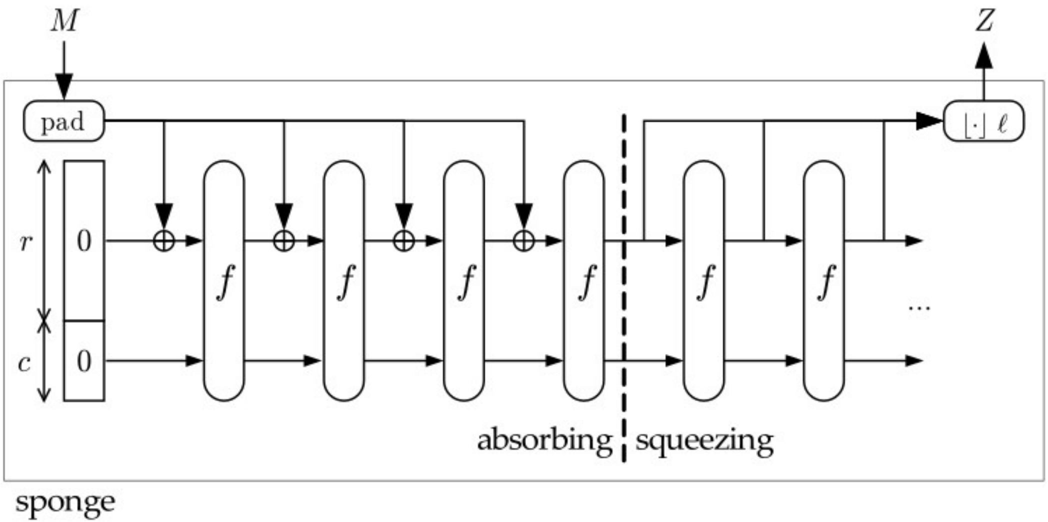


Fig. Sponge construct (time-view): *M* is input that, after padding, is divided in blocks of *r* (rate) bits; *Z* is output of *l* bits of length (specified by input parameter), concatenation of *r* bits' blocks; *c* is capacity, inner state bits, never output. (*in* keccak.team/sponge_duplex.html)

Overall weaknesses of irreversible systems

Problem:

- The number produced by the hashing operation is usually fixed (finite)
  - So, there **have to be** collisions, in an infinite universe of inputs!
  - Will they be likely or easy to cause?

Answer:

- that depends
  - on the randomness of the values resulting from the operation
  - on the size of those values (number of bits)
  - on the intended application

Attacks?

- certain: only brute force! (if one can live for enough time...)
  - the intention is to find an entry with a specific result?
    - try 2^n inputs (n, number of bits of *hash*)
- likely: perhaps by using certain curious techniques...
  - the intention is to find two entries with the same result?
    - **birthday attack**: try $\sqrt{2^n} = 2^{n/2}$ inputs for 50% chance of success
    - 2 sets of documents with the same *hash*: one "good" set, one "evil"![1]
- possible: scientifically search for construction weaknesses
  - research, research, research
    - MD5: MD5 considered harmful today
    - SHA-1: We have broken SHA-1 in practice
    - **...**

1 Diversity of possibilities for trying different documents are as simple as varying the number of spaces between words...

Ideal strength of hash function of n-bit output:

- security is as good as a random oracle with output truncated to *n* bits
- implies resistance of size:
  - $2^{n/2}$ for strong collision attacks
  - 2^{n} for weak collision attacks

Example: sponge construction (SHA-3) strength

- with random permutation: as strong as a random oracle
- capacity *c* determines resistance size:
  - 2^{c} for both strong and weak collision attacks
    - unfortunately, security is traded for speed, for constant *b* (= *r*+*c*) size
      - higher security (*c*), lower speed (more *r*-bit input blocks to process)

Pointers...

- "**Block cipher mode of operation**", Wikipedia
  - en.wikipedia.org/wiki/Block_cipher_mode_of_operation
- "**The sponge and duplex constructions**", G. Bertoni, J. Daemen, S. Hoffert, M. Peeters, G. Van Assche, R. Van Keer
  - keccak.team/sponge_duplex.html