COMPUTER SECURITY

Randomness

- essential in Cryptography!
    - one time pad, IV (initialization values), stream cipher seeds, hashes, *nonces*, key generation…
- generation
    - excellent: physical source
        - inherent: radioactive decay, brownian movement, ...
        - depending on initial conditions: (non-biased) roulette or dice, ...
    - reasonable: algorithmic-based with physical seed
        - cryptographically secure pseudorandom number generators
            - use physical (≅ random) sources (e.g. mouse movements)[1]
    - bad: algorithmic-based
        - pseudorandom number generators (e.g. POSIX's `random()`)

1 Linux's `getrandom()` (`/dev/random`, `/dev/urandom`)

Evaluation

- frequency analysis
  - determine the frequency distribution of digits or bit patterns of a sequence of values:
    - if (truly) random, each digit or bit occurs with approximately equal frequency
- entropy measurement[1]
  - measure of the unpredictability of the values in sequence:
    - if values are (truly) random, unpredictability (so, entropy) is maximum

1 Calculation of entropy varies. In computing, if values occur with equal probability, $E = \log_2$ (no. of possible values) ; if value is one bit, it can be 0 and 1; then $E = 1$ (bit).
In information theory (Shannon!) E (in bits) = $-\sum_i$ [(probability of occurrence of value i)*log2 (probability of occurrence of value i)] , where i is a value from a possible set. Again, if i is one bit, and its 0 or 1 value occurs with equal probability, $E = 1$ (bit).

...Randomness: evaluation...

- statistical tests
  - examination of properties such as uniformity, independence and distribution of sequence values. Examples: Chi-square[1], Kolmogorov-Smirnov[2], RUNS[3].
    - if sequence is (truly) random, results depend on specific test performed
- serial correlation measurement
  - check for correlations between successive values:
    - if (truly) random sequence of values, correlation should be zero
- *randomness* tests
  - run specialized tests. Examples of test suites: NIST Statistical[4], Dieharder[5], ENT[6].
    - if sequence is (truly) random, results depend on specific test performed

1 en.wikipedia.org/wiki/Chi-squared_test
2 en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test
3 en.wikipedia.org/wiki/Wald%E2%80%93Wolfowitz_runs_test
4 nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf
5 webhome.phy.duke.edu/~rgb/General/dieharder.php
6 www.fourmilab.ch/random/

ENT, A Pseudorandom Number Sequence Test Program

- battery of tests:
    - frequency (ideal: all values with same number of occurrences)
    - entropy (ideal: 8 bits per byte)
    - compression (ideal: 0 % compression)
    - Chi-square (ideal:] ~10%, ~90% [)
    - arithmetic mean (ideal: 50% of possible values)
    - Monte Carlo value for Pi (ideal: Pi with very "low" error)
    - Serial correlation coefficient (ideal: 0)
- used in a SEED lab!

Cryptographic models

Definitions

- cryptographic model
  - mostly, formal description of the security properties and assumptions of a cryptographic system
  - should define: adversarial capabilities; security goals[1]; security assumptions (environmental and operational details[2])...
  - so, includes attack models

1 e.g. confidentiality
2 such as computing resources

...Cryptographic models' definitions...

- attack model[1]
  - specification of the assumptions attributed to cryptanalysts targeting[2] a cryptographic system
  - depend on several perspectives: goals, knowledge, capabilities
- (computational) oracle
  - "black box" that is able to produce a (true) solution for any instance of a given computational problem (i.e. a decision problem)
- random oracle
  - specific oracle that
    - for each input, outputs a unique and (truly) random value, uniformly distributed in the (infinite) co-domain
    - is deterministic: always outputs the same value every time the same input is submitted

1 or: classification of attacks
2 attempting to break

Attack models: cryptanalyst perspectives

- goals
- knowledge
- capabilities

Goals of cryptanalyst:

- capture the keys
  - break the system, as cryptographic protection failed!
- capture plaintexts
  - partial break of confidentiality protection
- forge (or replay) plaintexts
  - partial break of integrity protection
- deny services (or communication)
  - break of availability protection

Knowledge of cryptanalyst:

- knows almost nothing of system's details (black-box, closed system)
  - in principle, great attack difficulty if system is really robust
- knows some system's details (grey-box system)
  - before attack, additional information gathering is needed (e.g. with social engineering)
- knows all system's details (white-box, open system)
  - in principle, least difficult to attack, unless strength of system relays in its inner robustness

Capabilities of cryptanalyst:

- standard
  - limitations are just the amount of time and computational power available (so, not knowledge)
- passive (mostly)
  - basic
    - has access to ciphertexts only (that is not able to choose)
  - known plaintext
    - some (plaintext, ciphertext) pairs are available

...Attack models: capabilities...

- active interaction
  - basic
    - can query and interact with target system
  - chosen plaintext
    - is able to prepare plaintexts and obtain their ciphertexts[1] [2]
  - adaptive chosen plaintext (real time interaction?...)
    - is able to iteratively query the system with a succession of plaintexts, after receiving corresponding ciphertexts
  - chosen ciphertext
    - is able to prepare ciphertexts and obtain their deciphered counterparts[3]
    - is able to prepare ciphertexts that will decipher to predictable plaintexts
  - adaptive chosen ciphertext
    - is able to iteratively query the system with a succession of ciphertexts, after receiving corresponding plaintexts

1 e.g. from an encryption oracle. Trivial with public key cryptography! Why?
2 Exercise: show how to use this attack to obtain the key used by a (not so truly) one-time pad.
3 e.g. from a "decryption" oracle (makes more sense for digital signatures' attacks, as public key is used for "deciphering" signed docs)

...Attack models: capabilities...

- side-channel
  - can gather information not obviously related to the cryptographic protective operations: electronic noise, sound, elapsed time...
- social engineering
  - is able to trick some humans to give away partial or essential secrets

Defense models: cryptographer perspective

- defense will depend on knowledge of previous attack perspectives
- can be guided by more or less formal approaches, included in the mentioned cryptographic *models* (not covered here)

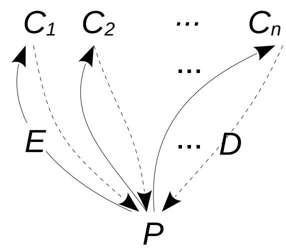
General enciphering schemes

Definition

- sets of algorithms and protocols used to transform plaintext (clear data) into ciphertext (concealed data) in such a way that unauthorized users cannot reverse the transformation.

Types

- deterministic encipherment
  - the same ciphertext is always produced for a given plaintext and key
- probabilistic encipherment [FIG]
  - different ciphertexts are, in general, produced for a given plaintext and key[1]
- format-preserving encipherment
  - ciphertext is produced in the same format[2] as the plaintext



1 An example is ElGamal's encryption system.
2 The meaning of "format" varies: only letters from English alphabet are used; *n*-bit block cipher (only *n*-bit numbers are accepted and produced), etc.

...General enciphering schemes: types...

- perfect secrecy encipherment
  - the ciphertext reveals no information at all about the plaintext
  - ideal goal: works even with an all-mighty cryptanalyst
- semantic security encipherment
  - the ciphertext possible informations about the plaintext, cannot be feasibly extracted
  - realistic goal: protects even with adaptive chosen plaintext attacks
- indistinguishable encipherment
  - a ciphertext does not reveal information to allow distinguishing which plaintext produced it from a group of chosen plaintexts[1]
- malleable encipherment
  - the ciphertext produced from a given plaintext can be modified in a way that the deciphered new plaintext is predictably related to the first
  - dangerous: does not protect against (adaptive) chosen ciphertext attacks

1 or the distinction is no better then that of random guessing

...General enciphering schemes...

$$\begin{tikzcd} C_1 \arrow[r, "f"] & C_2 \arrow[d, "D"] \\ P_1 \arrow[u, "E"] \arrow[r, dashed, "g"] & P_2 \end{tikzcd}$$

- homomorphic encipherment
  - the ciphertexts are able to suffer computations that, when deciphered, are identical to related computations on the corresponding plaintexts
  - useful with cloud computing, as cloud server will not need to know clients' deciphering keys
  - Ex.: RSA is homomorphic![1]
- (perfect) forward secrecy encipherment[2]
  - the capture of a session key (and so being able to decipher the session) will not allow the decipherment of previous sessions
  - Also, knowledge of a long-term key does not allow the decipherment of past sessions.)[3]

1. In RSA, if n is the modulus and e the encryption exponent: $C = E(P) = P^e \bmod n$. The homomorphic property is immediate: $E(P1) * E(P2) = E(P1*P2)$.
2. This has to do more with key exchange schemes than with the encipherment operations by themselves
3. However, the breaking of the encipherment *algorithm*, in the sense of being able to operate it without a cryptographic key, might allow the decipherment of past sessions.

Cipher modes of operation[1]

Base method

- $P = P_1 \, P_2 \ldots$ parts (blocks) of equal size
  - block size: 1 b, 1 B, 8 B (typical), 16 B (typical)...
- enciphering methods:
  - stream
    - $K = K_1 \, K_2 \ldots : C = E_{K1}(P_1) \, E_{K2}(P_2) \ldots =^{2} K_1(P_1) \, K_2(P_2) \ldots$
  - block
    - $K : C = K(P_1) \, K(P_2) \ldots$
  - "mix" of previous
    - $K, v_1, v_2 \ldots^{3} : C = E_K(P_1, v1) \, E_K(P_2, v2) \ldots = K_{v1}(P_1) \, K_{v2}(P_2) \ldots$

1 Necessary for the symmetric encipherment of "long" texts. But, in practice, almost any text is "long"!...
2 for simplicity
3 real single key with additional (and different) information per block: overall, looks like a different "virtual" key per block

Rationale for "operation modes"[1]

- stream
  - Pro: most secure[2]
  - Con: long, one-time usable, (random) key
- block
  - Pro: simplicity and 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
    - several possibilities

a) 
b) 

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

1 Goal is *confidentiality* protection; *integrity* protection is not guaranteed: with some modes, even the "mixed", modifications of ciphertext might go undetected; for confidentiality <u>and</u> integrity protection, <u>authenticated encipherment</u> is used.
2 even *provable* secure with *One-time pad*

Pictures' notation

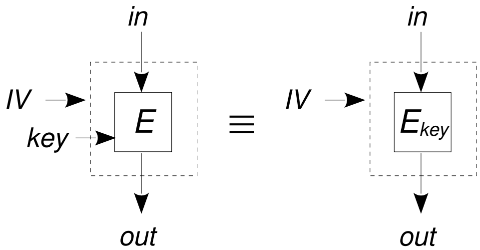


Fig. *IV* is Initialization Value (or Vector), public value that, as a rule, should be random.

Some operation modes

Stream method

- Some properties:
  - usually, $E = D = \text{XOR}$[1] ($\oplus$)
  - no padding of last block
  - parallelizable en/deciphering
  - ultimate security: K_i random, one-time value
- Formulas:
  - $C_i = E_{ki} (P_i)$, $i>0$
  - usually, $P_i = E_{ki} (C_i)$
- Error propagation:[2]
  - exercise!

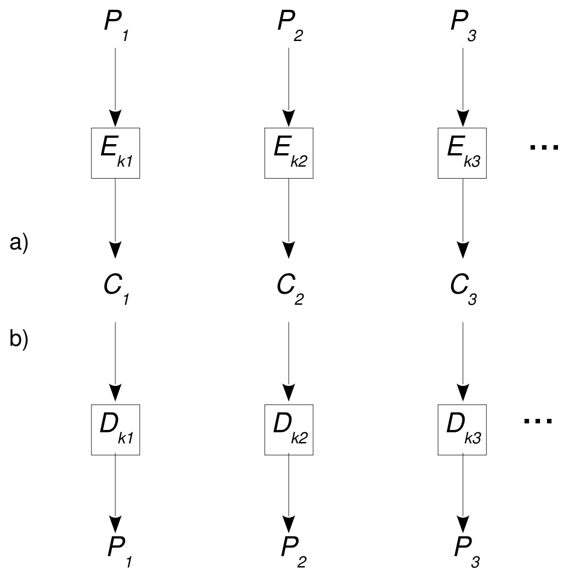


Fig. Use of plain stream method: a) enciphering; b) deciphering

1 bitwise
2 When at least one bit/byte of C_i is garbled, how that is reflected in following blocks.

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. :-)
- Error propagation:
  - exercise!

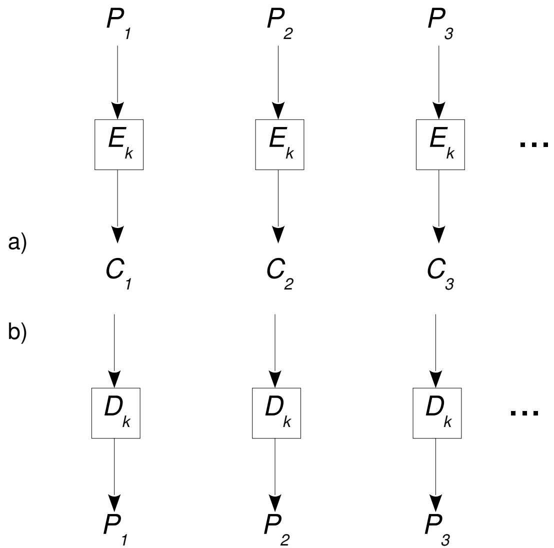


Fig. Use of (plain) block method: a) enciphering; b) deciphering.

“Mix” method: CTR

- *CTR, Counter Mode*
- Some properties:
  - *IV*[1] (random + counter)
  - no padding
  - parallelizable en/deciphering
- Formulas:
  - Write the en/decipherment formulas.
- Error propagation:
  - exercise!

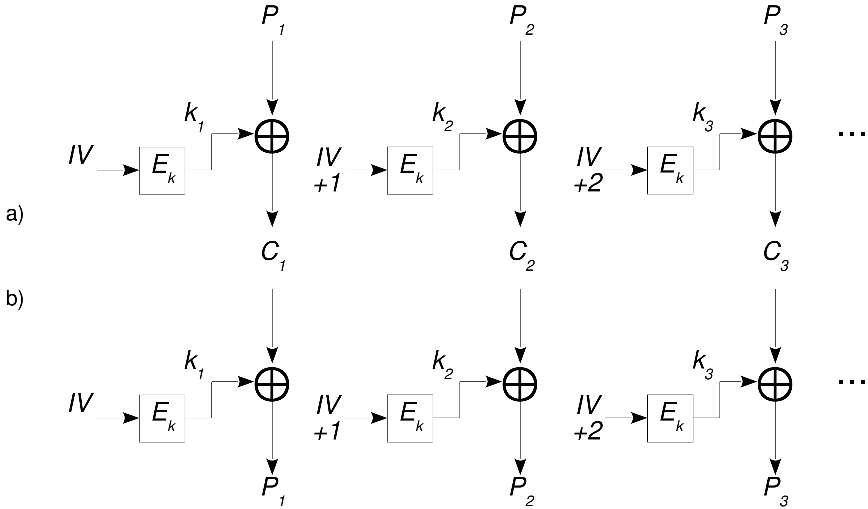


Fig. Use of “mixed” method CTR: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

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

“Mix” method: CFB

- *CFB, Cipher FeedBack*
- Some properties:
  - *IV* (random)
  - 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.
- Error propagation:
  - exercise!

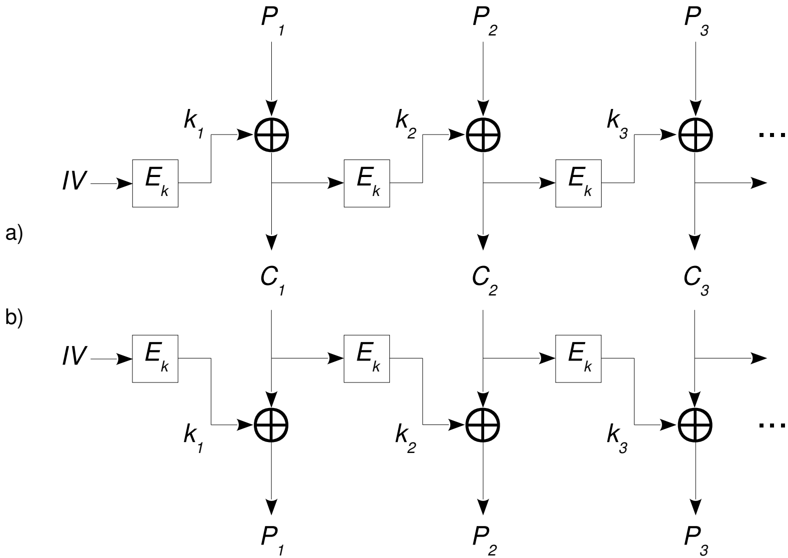


Fig. Use of “mixed” method CFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

“Mix” method: OFB

- *OFB, Output FeedBack*
- Some properties:
  - *IV* (random)
  - 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.
- Error propagation:
  - exercise!

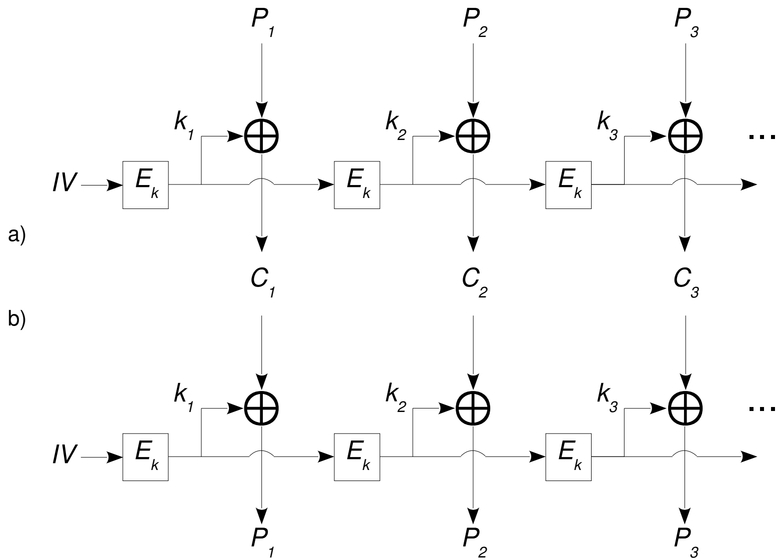


Fig. Use of “mixed” method OFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

“Mix” method: CBC

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

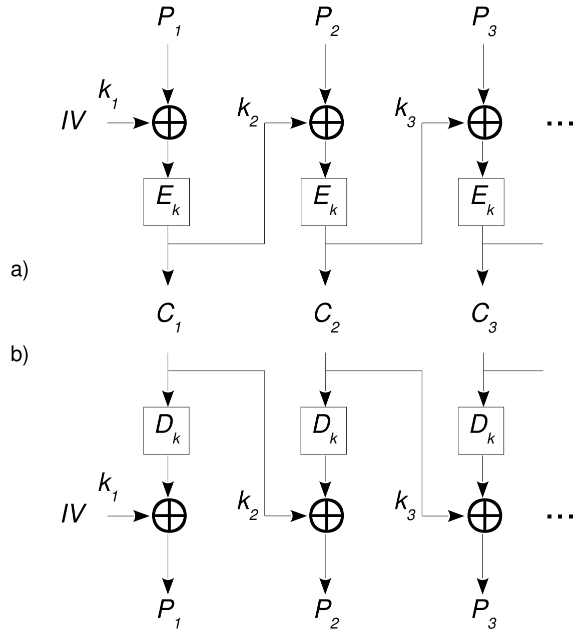


Fig. Use of “mixed” method CBC: a) enciphering; b) deciphering
(Notice the virtual keys k_i.)

Another view of some operation modes

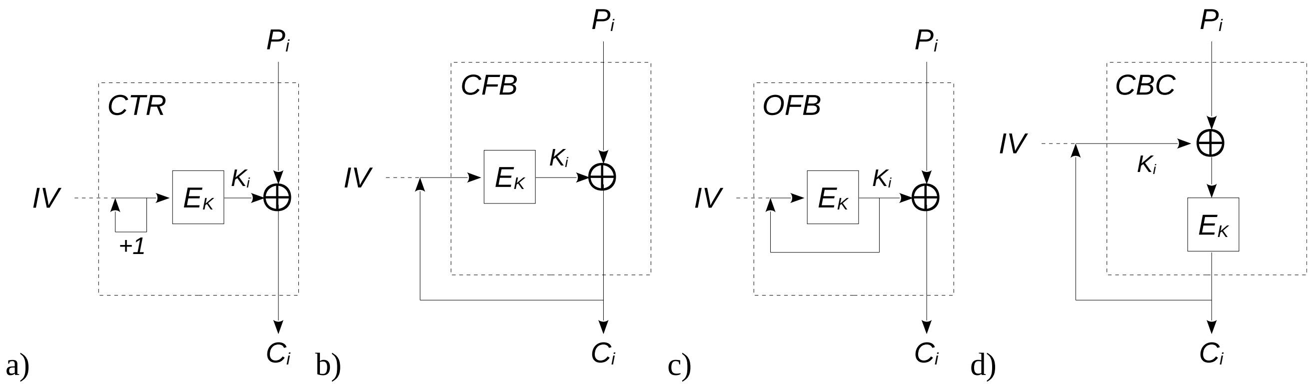


Fig. The software-view of some of the operation modes (*i>0*). In b) and c) the reason for the modes' names is apparent...

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 Why?... Also, some "modes of operation" do not need padding... why?
2 interception and examination of communications (ciphered or not) to deduce information (e.g. from patterns)
3 asecuritysite.com/encryption/crackrsa2
4 As same plaintext always produces same ciphertext, a cryptanalyst may build a collection of plaintext/ciphertext pairs and look for cipher matches in communication media; it is specially feasible with "public-key cryptography" (why?)!

Padding schemes

- several schemes (bit padding or, more usually, byte padding)
  - shared-key cryptography
    - e.g. PKCS[1] #5[2], #7[3] (enciphering) [Fig. ShKey]
  - one-way cryptography
    - e.g. RFC 6234 (SHA-1, SHA-256) [Fig. OneWay a)]
    - e.g. SHA3 (sponge) [Fig. OneWay b)]
  - public-key cryptography
    - e.g. PKCS #1 v2 (RFC 8017)
      - RSA's PKCS1-v1_5 [Fig. PKCS1]
      - RSA's OAEP, Optimal Asymmetric Encryption Padding [Fig. OAEP]
        - 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

...Cipher modes of operation: padding examples (figs)...

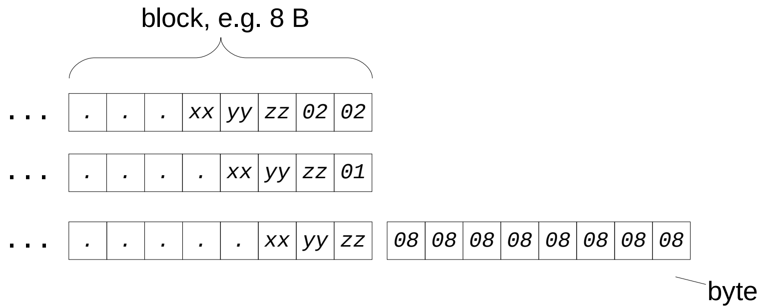


Fig. ShKey: Shared-key cryptography padding: examples for PKCS #5 (8B blocks); #7 will be similar, but appropriate to 16B blocks.

Algorithm: add (`block_size - P_length mod block_size`) bytes; all with value equal to number of added bytes: e.g. if 3 bytes are needed to complete last block, each added byte's value is 3.

...Cipher modes of operation: padding examples (figs)...

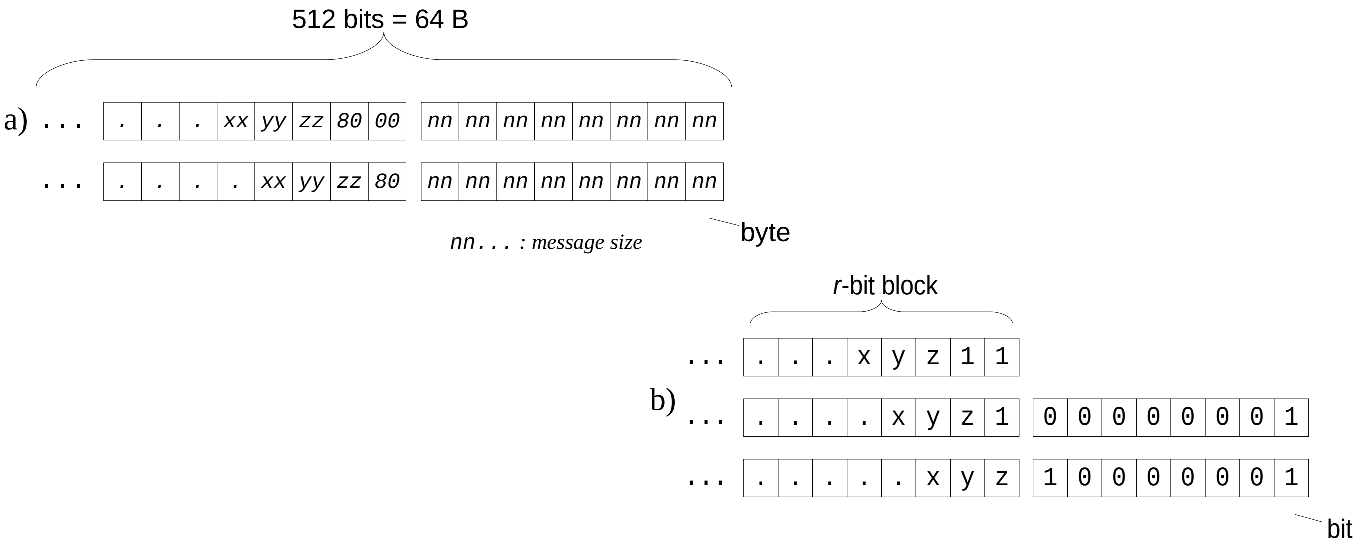


Fig. OneWay: Instances of one-way cryptography padding:
a) RFC 6234 padding: (SHA1, SHA256...) - sequence of *nn*s is message size;
b) Sponge *multirate* padding: `10*1` (*r* is the number of bits of input block).

...Cipher modes of operation: padding examples (figs)...

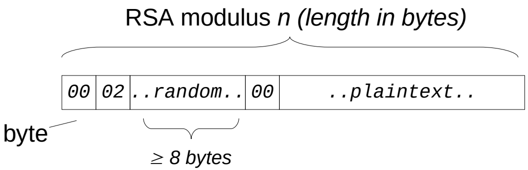


Fig. PKCS1. RSA padding: PKCS1-v1_5.

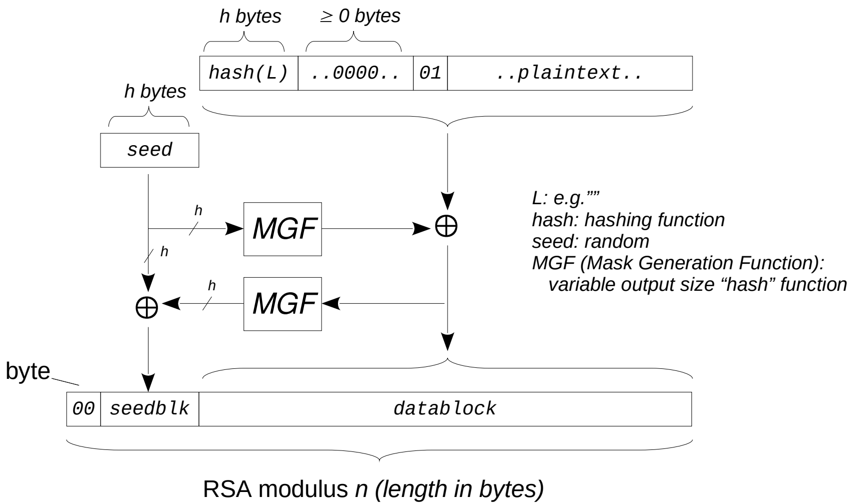


Fig. OAEP. RSA padding: OAEP, Opt*imal Asymmetric Encryption Padding.*
After padding, RSA enciphering proceeds with final data being treated as of *n*-byte hex number.

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 of previous block until padding is valid; then repeat for previous bytes
    - see [FIG] (PKCS #5, #7 padding)

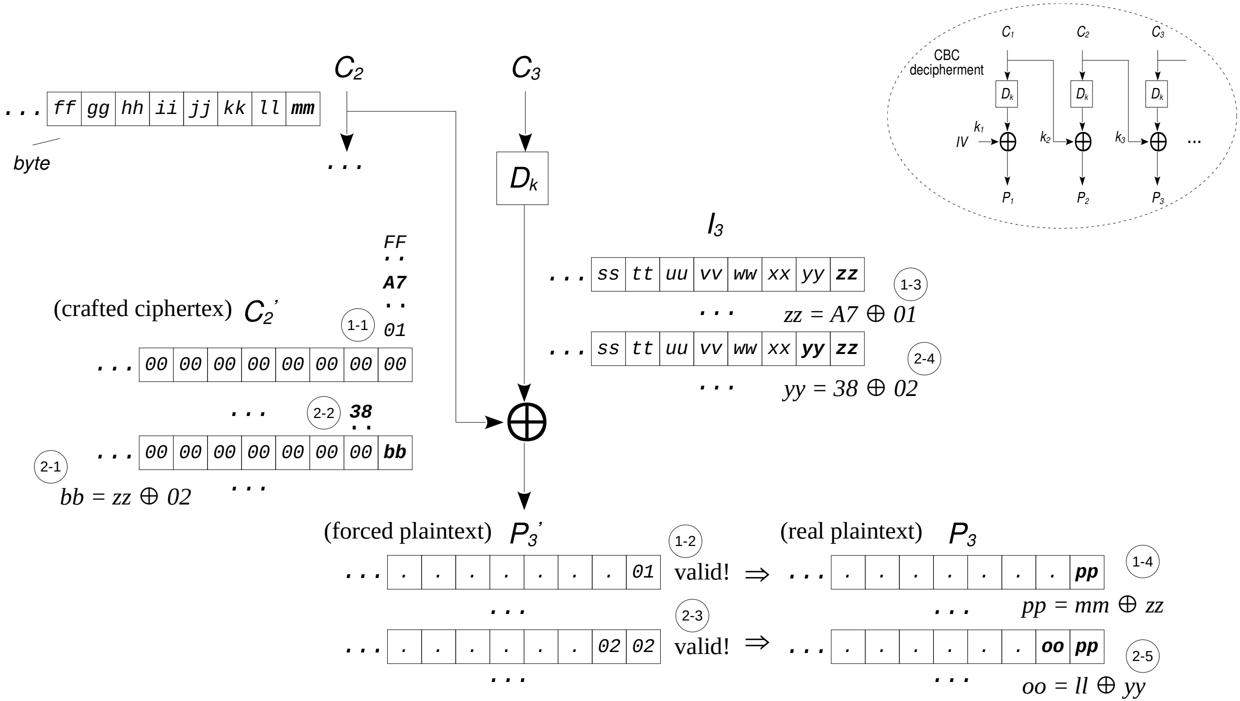
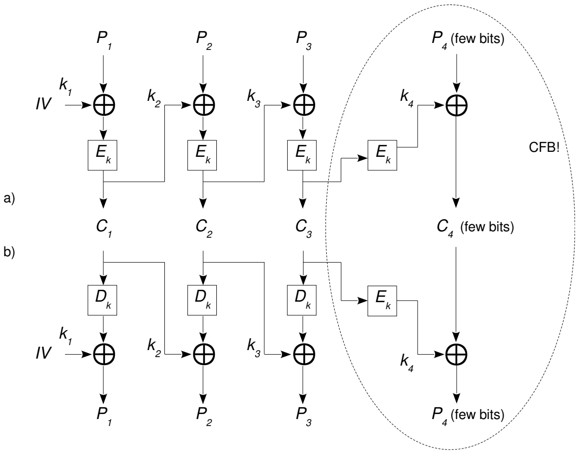
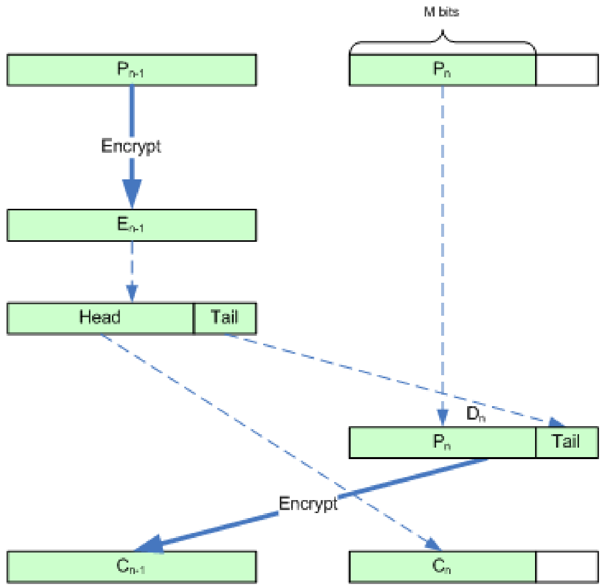


Fig. Padding oracle attack procedure for PKCS #5, #7 padding (CBC mode). C_3 is last cipher block.

Real need for padding?

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





(to be continued...)