COMPUTER SECURITY

Attack models[1]

Definition

- specification of the kind of access a cryptanalyst has when attempting to break a targeted cryptographic system[2]

Attack types

- normal
  - cryptanalyst can only obtain ciphertexts that are available (and that he does not control)
- known plaintext ("passively" obtained)
  - cryptanalyst has obtained the enciphered counterparts of some plaintexts

1 or: classification of attacks
2 Cryptanalyst limitations due to time and computational power are not considered (or are assumed to be the maximum reasonably possible).

...Attack models...

- chosen plaintext (“actively” obtained)
  - cryptanalyst can obtain the ciphertexts of any plaintext she chooses[1]
    - trivial with public key cryptography! (Why?)
    - Exercise: show how to use this attack to obtain the key used by a (not so truly) *one-time pad*.
- adaptive chosen plaintext (“interactively” obtained)
  - cryptanalyst can obtain the ciphertexts of additional plaintexts after seeing previous pairs (plaintext, ciphertext); usually this implies a (real time?) interaction with encryption oracle
- chosen ciphertext
  - cryptanalyst can gather information by decipherment of chosen ciphertexts[2]
- adaptive chosen ciphertext
  - cryptanalyst may also be able to choose ciphertexts to be deciphered based on previous decipherments; usually this implies a (real time?) interaction with decryption oracle

1 i.e. from an encryption oracle
2 i.e. from an decryption oracle

...Attack models...

- open key
  - cryptanalyst has obtained sufficient knowledge about the key to allow the decipherment of ciphertexts[1]
- side-channel
  - cryptanalyst can gather information not obviously related to the encipherment operations (electronic noise, sound, elapsed time) that allow the grabbing of the key[2]
- social engineering
  - cryptanalyst tricks some human to decipher ciphertexts or yield the secret key

Study of defense

- the defense from an attack depends on the characteristics of the cryptographic system; the formal[3] approach uses *cryptographic models*

1 of course, made with that key
2 or, at least, the decipherment of some ciphertexts
3 kind of theoretical

Cryptographic Models

Definition

- formal descriptions of the security properties and assumptions of cryptographic systems. Should define: adversarial capabilities; security goals[1]; security assumptions (environmental and operation details[2])...

Models

- standard
  - the cryptanalyst is only limited by the amount of time and computational power she has available (so, access to all operation details and capabilities is granted)

1 e.g. confidentiality
2 such as computing resources

...Cryptographic Models...

- random oracle[1]
  - the cryptanalyst may use an (ideal) function (or black box) 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
- generic group
  - the cryptanalyst is given access to a randomly chosen "encoding" of a group, instead of specific "encodings" (eventually used in practice)
- common reference string
  - the cryptanalyst is given access to a special string, taken from some distribution, that is shared by all involved parties
  - common random string
    - when the string is taken from a uniform random distribution

1 Oracle is a "black box" that is able to produce a (true) solution for any instance of a given computational problem (i.e. a decision problem).

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 (hopefully random) sources (e.g. mouse movements)
      - Linux's `getrandom()` (`/dev/random`, `/dev/urandom`)
  - bad: algorithmic-based
    - pseudorandom number generators
      - POSIX's `random()`

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 is maximum, and so is entropy

1 Calculation of entropy varies: in computing, if values occur with equal probability, $E = \log_2$ (no. of possible values) ; if value is a 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 a 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!

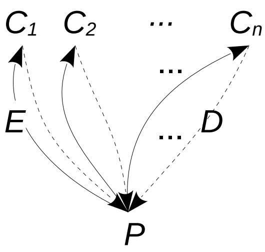
General enciphering schemes

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



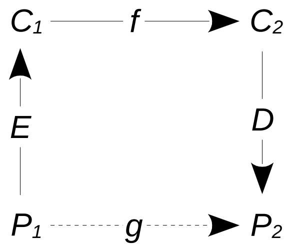
1 An example is ElGamal's encryption system.
2 The meaning of "format" varies: e.g. only letters from English alphabet are used; e.g. *n*-bit block cipher (only *n*-bit numbers are accepted and produced).

...General enciphering schemes...

- perfect secrecy encipherment
  - the ciphertext reveals no information at all about the plaintext
- semantic security encipherment
  - the ciphertext could reveal some information about the plaintext, but it cannot be feasibly extracted
- indistinguishable encipherment
  - a ciphertext does not reveal information to allow distinguishing which plaintext produced it from a group of chosen plaintexts; or the distinction is no better then that of random guessing
- malleable encipherment
  - the ciphertext produced for a given (possibly unknown) plaintext can be transformed into another ciphertext which deciphers to a plaintext related to the first

...General enciphering schemes...

- <u>homomorphic encipherment</u>
  - the ciphertexts are able to suffer computations that, when deciphered, are identical to related computations on the corresponding plaintexts
  - Ex.: RSA public key with modulus n, encryption exponent e and plaintext P:
    $E(P) = P^e \bmod n$. The homomorphic property is then:
    $E(P_1) * E(P_2) = P_1^e * P_2^e \bmod n = (P_1 * P_2)^e \bmod n = E(P_1 * P_2)$



- <u>(perfect) forward secrecy encipherment</u>[1]
  - being able to grab a session key (and so being able to decipher the session) does not allow the decipherment of previous sessions. (Also, knowledge of a long-term key does not allow the decipherment of past sessions.)[2]

1 this has to do more with key exchange schemes than with the encipherment operations by themselves

2 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.

Long[1] texts' encipherment: operation modes

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 See that, in practice, almost any text is "long"!
2 for simplicity
3 Real single key with additional (and different) information per block: overall, 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: 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

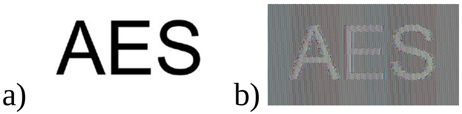


Fig. a) original picture;
b) enciphered with AES 256b, ECB mode

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

Pictures' notation

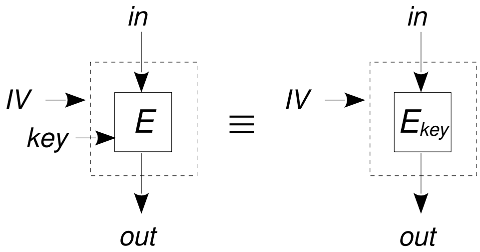


Fig. *IV* is Initialization Value (or Vector), public value that, as a rule, should be random.

Some operation modes

Stream method

- Some properties:
  - usually, $E = D =$ XOR[1] (⊕)
  - 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:
  - exercise!

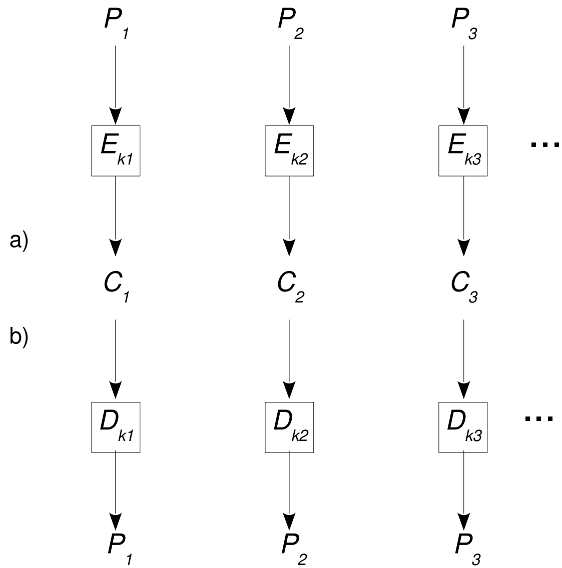


Fig. Use of plain stream method: a) enciphering; b) deciphering

1 bitwise

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!

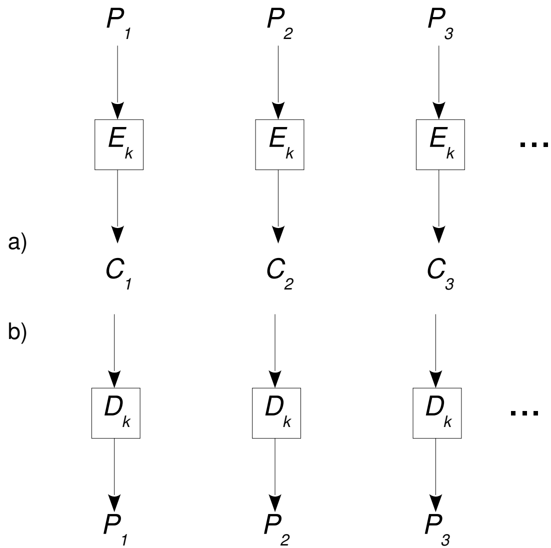


Fig. Use of (plain) block method: a) enciphering; b) deciphering.

“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!

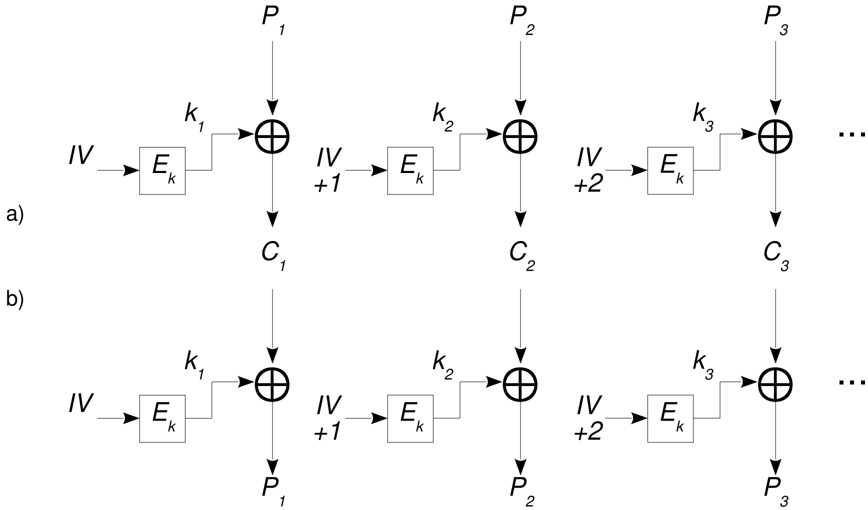


Fig. Use of “mixed” method CTR: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

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!

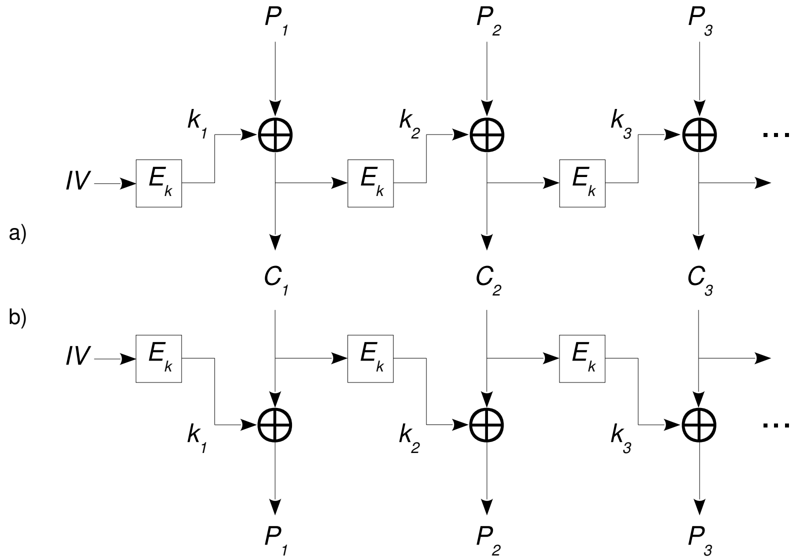


Fig. Use of “mixed” method CFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

“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!

a)

b)

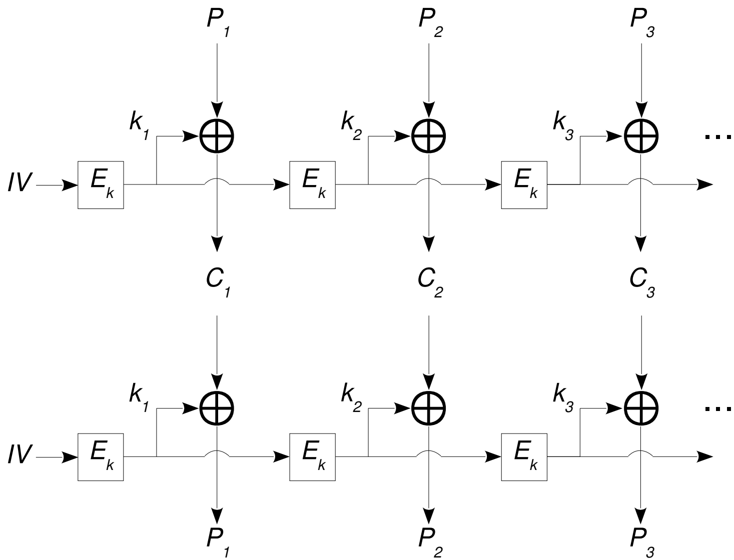


Fig. Use of “mixed” method OFB: a) enciphering; b) deciphering. (Notice the virtual keys k_i.)

"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:
  - $C_0 = IV$; $C_i = E_k (P_i \oplus C_{i-1})$ $i>0$
  - Write the decipherment formula.
- Error propagation:
  - exercise!

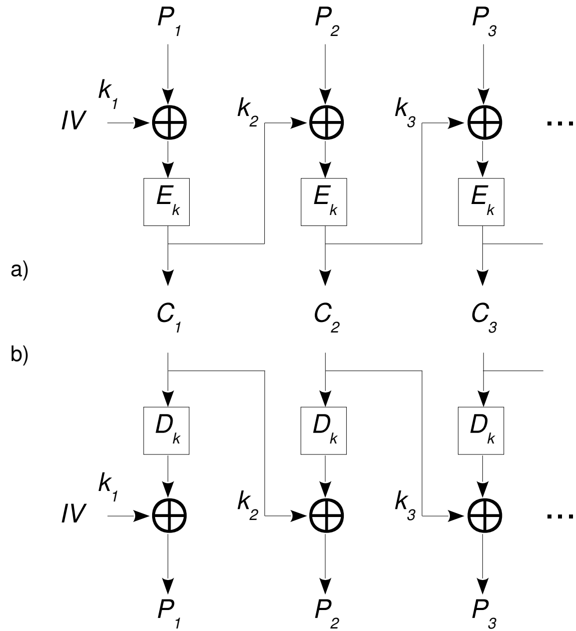


Fig. Use of "mixed" method CBC: a) enciphering; b) deciphering (Notice the virtual keys k_i.)

..."Long" texts' encipherment: operation modes…

Another view of some operation modes

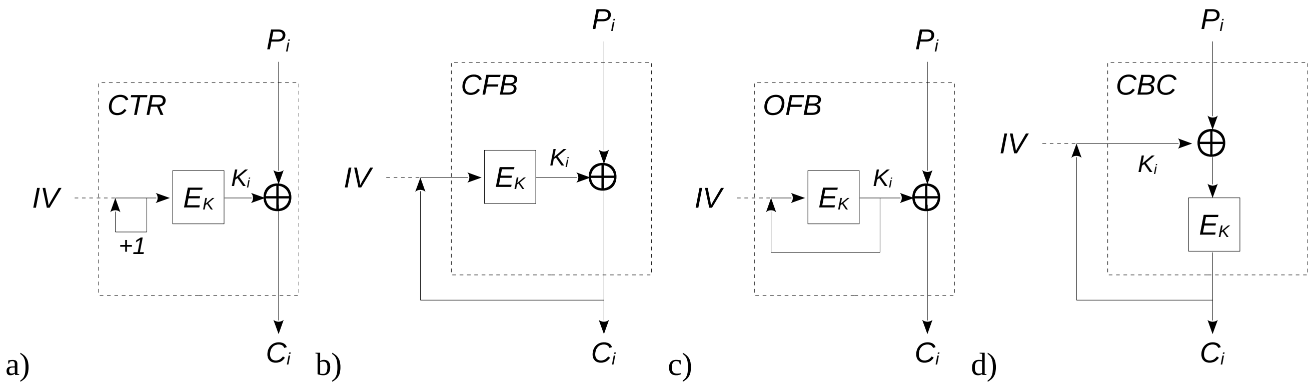


Fig. The software-view of some of the operation modes. In b) and c) the reason for the modes' names is apparent...

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, an attacker 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]
  - 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)...

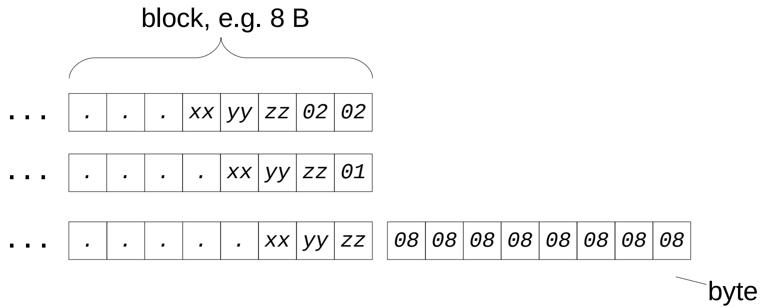


Fig: Shared-key cryptography padding: examples for PKCS #5 (8B blocks); #7 will be similar, but appropriate to 16B blocks.

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

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

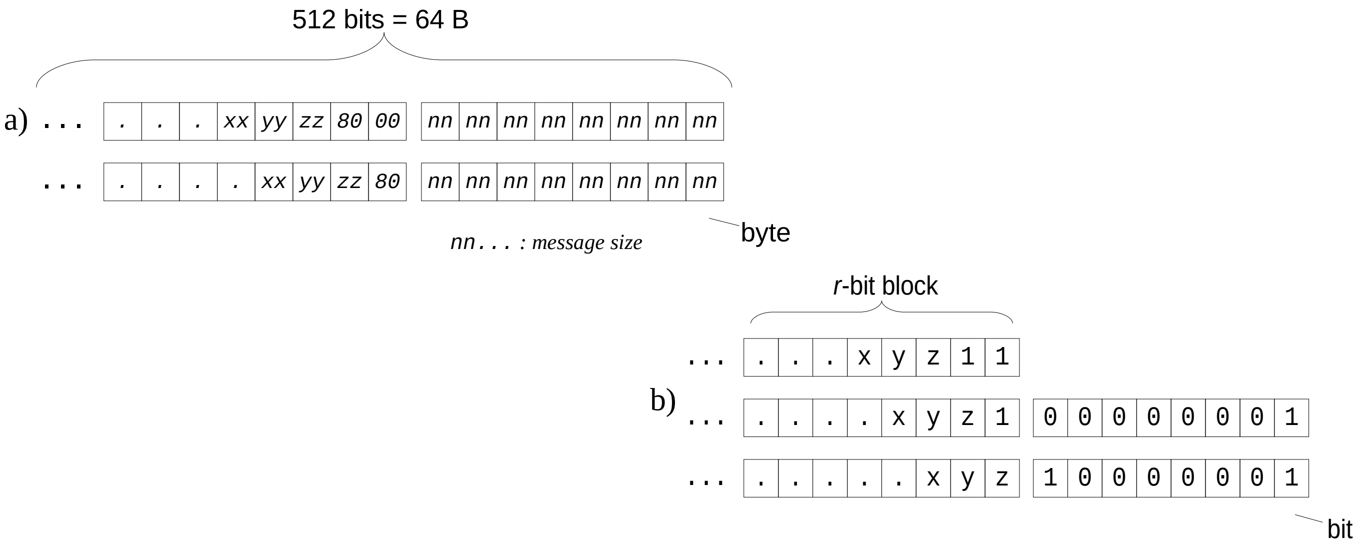
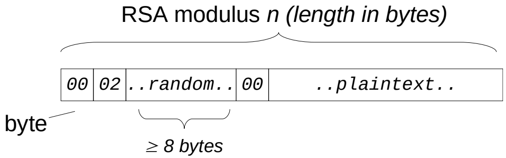


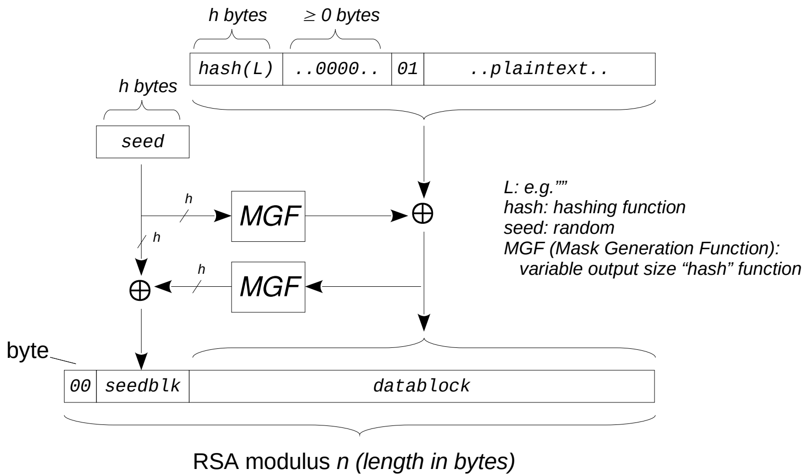
Fig: Instances of one-way cryptography padding:
a) RFC 6234 padding: (SHA1, SHA256...) - sequence of *nn*s is message size;
b) Sponge *multirate* padding: `10*1` (*r* is the number of bits of input block).

..."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)

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)

...”Long” texts' encipherment: Padding...

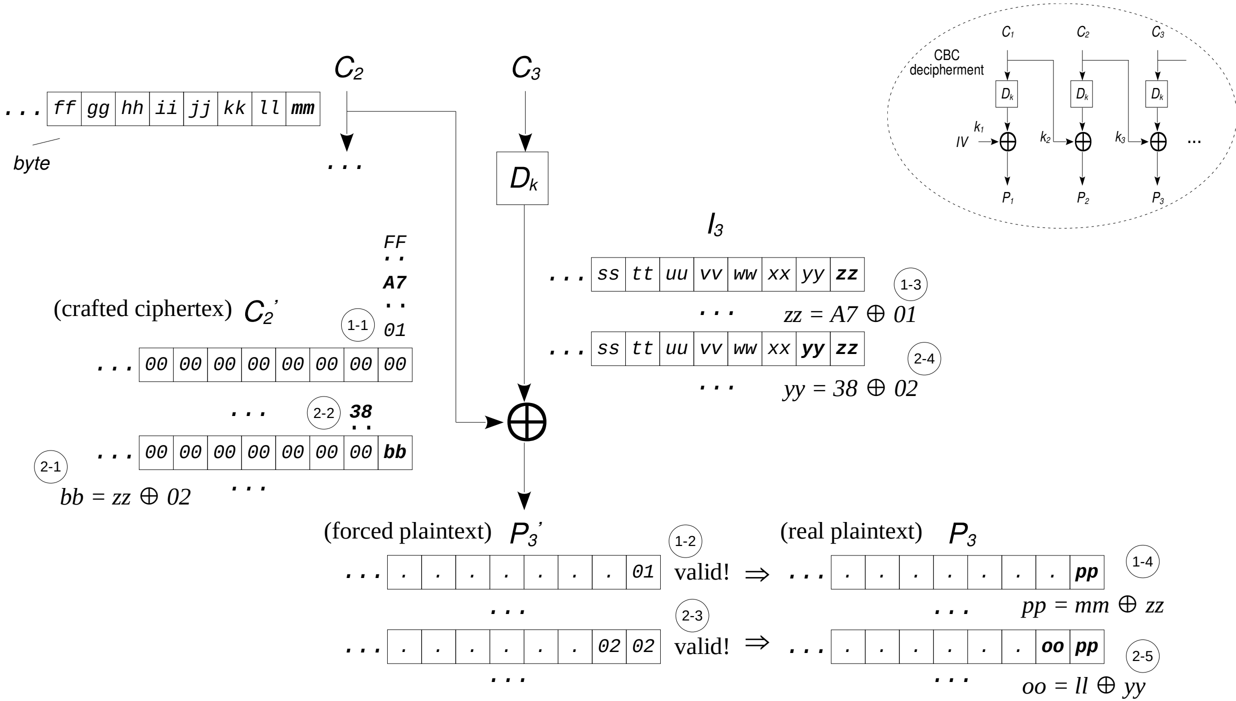
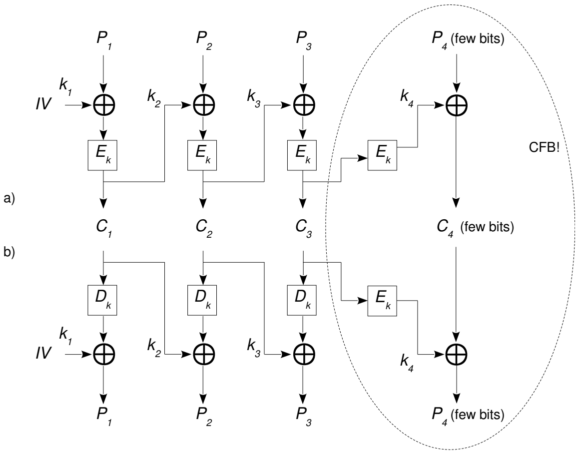


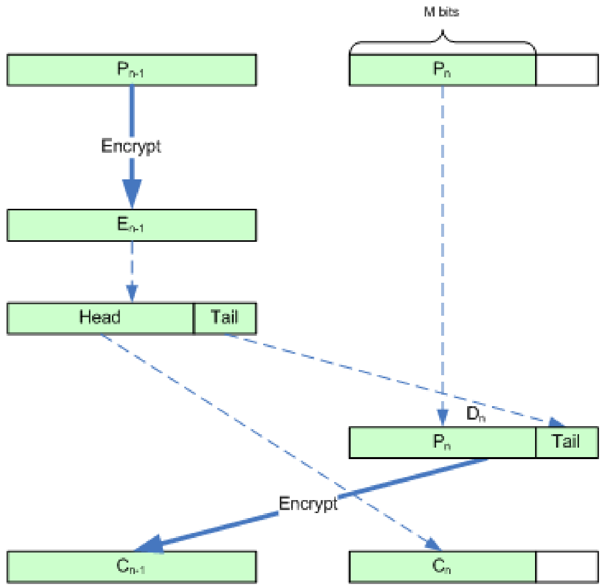
Fig. Padding oracle attack procedure for PKCS #5, #7 padding (CBC mode). C_3 is last cipher block.

...”Long” texts' encipherment: Padding...

Real need for padding?

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





(to be continued...)