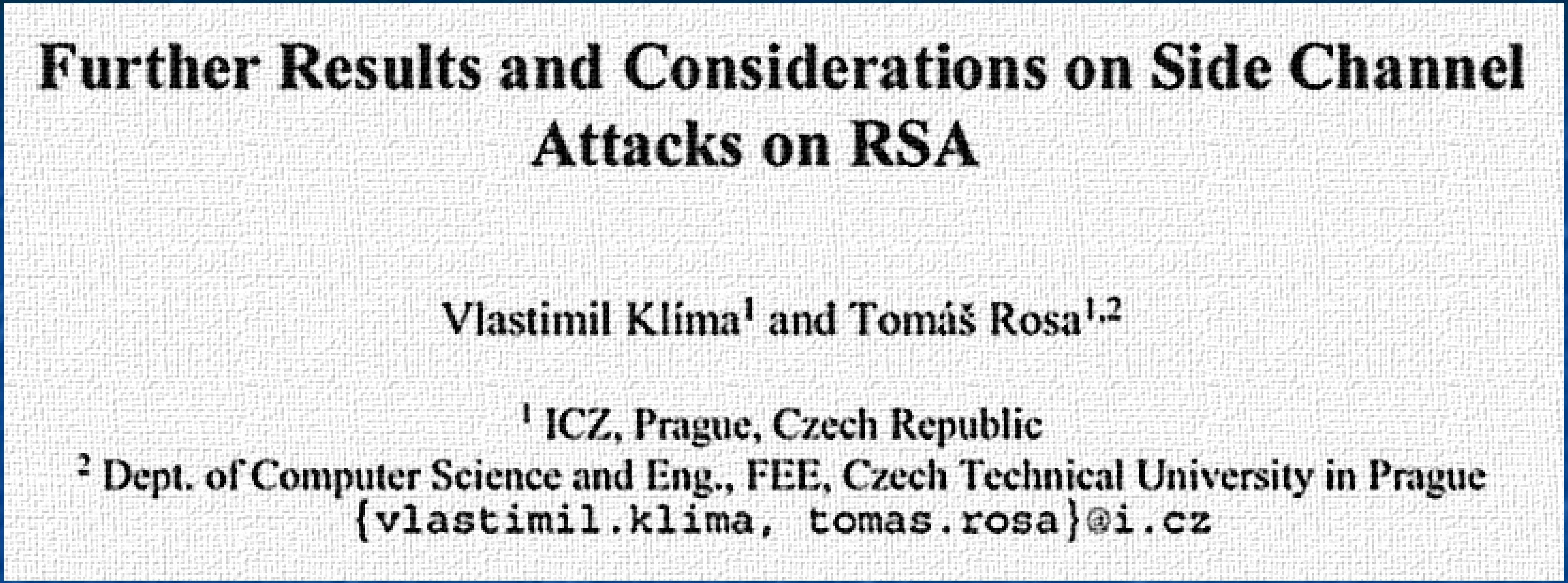


Further Results and Considerations on Side Channel Attacks on RSA

Vlastimil Klíma[1] and Tomáš Rosa[1,2]

[1] ICZ, Prague, Czech Republic
[2] Dept. of Computer Science and Eng., FEE, Czech Technical University in Prague
{vlastimil.klima, tomas.rosa}@i.cz

Presented at the workshop CHES 2002, August 13-15, 2002, Redwood Shores, California, USA

Main Topics

- Another possible side channel attack on RSAES-OAEP
- Fault side channel attacks on RSA-KEM
- Note on the conversion from plaintext-oriented attacks to signature-oriented ones

Side Channels

- Definition: *Side Channel (SC)*
  - An undesirable way, which a cryptographic module exchanges some information with its neighborhood in.
    - Timing
    - Power
    - Electromagnetic
    - Fault
    - Kleptographic

$\Bigg\}$ side channel

Side Channels

- Analysis of the side channel
  - A process of extracting a useful information from the particular side channel.
- Attack based on the side channel
  - A process of using an analysis of the particular side channel against a given cryptographic module.

PART I

- Another possible side channel attack on RSAES-OAEP

Partial Information Oracle

- Definition: *<u>Partial Information Oracle</u> (PIO) is a black-box function of the ciphertext c, $c \in$* ***C****, such that:*
  - PIO: $\boldsymbol{C} \rightarrow \mathrm{Im}(PIO)$,
  - for at least one bit m_i of the RSA plaintext m, $m=c^d \bmod n$, we have:
    - $H(m_i \mid \mathrm{PIO}(c)) < H(m_i)$,
    - information from PIO gained in this way must be non negligible – is induced by a <u>non negligible advantage</u> (*adv*).

Whole Information Oracle

- Definition: *<u>Whole Information Oracle</u> (WIO) is a known algorithm based on an access to the particular PIO. It takes as an input the value of ciphertext c and it <u>returns the value of the whole plaintext</u> m, $m = c^d \bmod n$.*
    - $\mathrm{WIO}_{\mathrm{PIO}}$: $\boldsymbol{C} \rightarrow \boldsymbol{M} : c \rightarrow m = (c^d \bmod n)$

WIO Existence

- Having an access to the particular PIO, it is possible to build up the WIO.
  - Given the ciphertext *c*, WIO returns the corresponding plaintext *m* in the random polynomial time.
  - The proof of WIO existence is based on the theorem of *The Security of Individual RSA Bits* [13].

PIO Hunting Problem

- Where and how to find an appropriate PIO…
    - Once we are having one, we are also having a chance to develop an efficient chosen ciphertext attack on the whole RSA scheme.

PIO – Examples (1)

- PIO_{lsb}
  - Returns the least significant bit of the plaintext (i.e. m_0).
- PIO_{half}
  - Says whether the plaintext *m* is less or higher then one half of the modulus *n*.
  - Note: A PIO_{lsb} and a PIO_{half} are polynomially equivalent ([25]).

PIO – Examples (2)

- $\text{PIO}_{\text{MSByte}}$
  - Indicates whether the most significant byte is zero or not.
  - Used in the Manger’s attack on the RSAES-OAEP [15].
- $\text{PIO}_{\text{PKCS1-v1\_5}}$
  - Indicates whether the plaintext is „PKCS1-v1_5 conforming“ or not.
  - Used in the Bleichenbacher’s attack on the RSAES-PKCS1-v1_5 [5].

PIO Based on a Power Side Channel

- We assume that Hamming weights of arguments of the particular inner operation in the RSAES-OAEP scheme are leaking through a power side channel. We have PIO_{PA}.

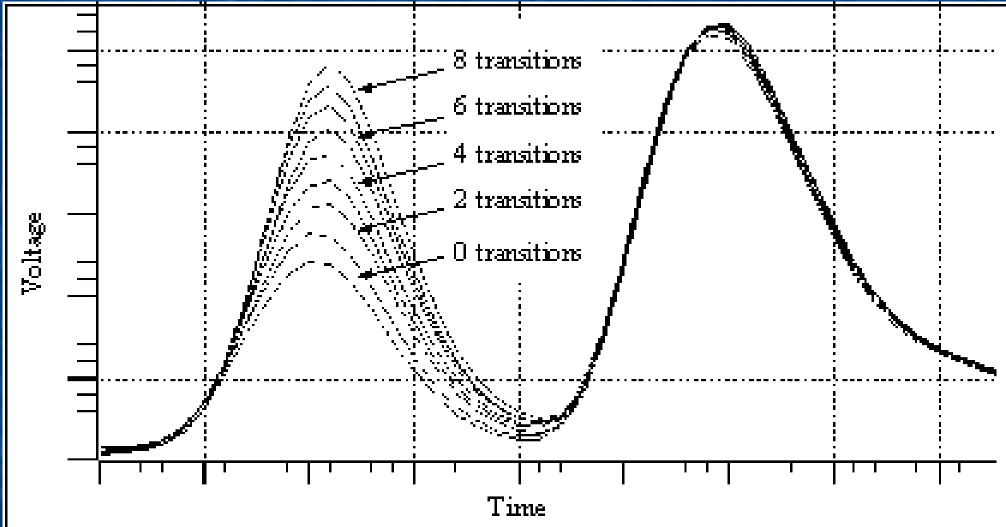


FIGURE 2. Number of Bit Transitions versus Power Consumption
These results show how the data effects the power levels. The nine overlayed waveforms correspond to the power traces of different data being accessed by an LDA instruction. These results were obtained by averaging the power signals across 500 samples in order to reduce the noise content. The difference in voltage between *i* transitions and *i*+1 transitions is about 6.5 mV.

([16])

Place of Our Attack on RSAES-OAEP

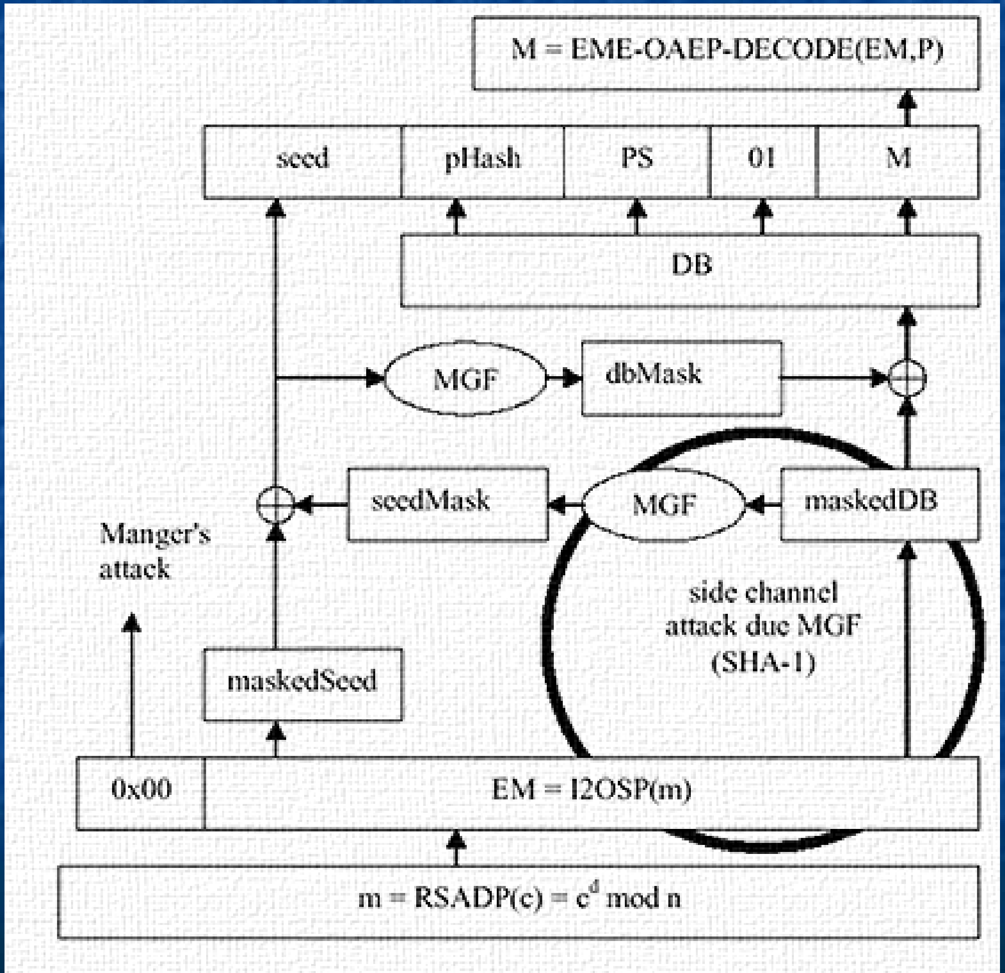


Fig. 1. New side channel attack against RSAES-OAEP

Assumptions on PIO_{PA}

- Let us send c and c' to a PIO_{PA} ...
  - Let $m = m[k\text{-}1] \,||\, \ldots \,||\, m[0]$.
  - Let us denote (we use MGF/SHA-1 notation):
    - $W_8 = m[10] \,||\, m[9] \,||\, m[8] \,||\, m[7]$,
    - $W_9 = m[6] \,||\, m[5] \,||\, m[4] \,||\, m[3]$,
    - $W_{10} = m[2] \,||\, m[1] \,||\, m[0] \,||\, 00$. *(00 appended by MGF)*
  - Analogically also for m' and W_8', W_9', W_{10}'.
- <u>Main assumption</u>
  - From PIO_{PA} we get triplets of Hamming weights:
    - HW = (w(W_8), w(W_9), w(W_{10}))
    - HW′ = (w(W'_8), w(W'_9), w(W'_{10})).

Converting PIO_{PA} to a PIO_{lsb} (1)

- Main aim
  - Convert a PIO_{PA} to a PIO_{lsb}
- Main idea
  - Let $c' = c*2^{-e} \bmod n$. Let $m = c^d \bmod n$ and $m' = (c')^d \bmod n$.
  - If $\text{lsb}(m) = 0$ then $m' = m >> 1$.
  - If $\text{lsb}(m) = 1$ then $m' = (m + n) >> 1$.
    - Here the operator ">> 1" denotes a right shift by one bit.
- Putting it together
  - If $\text{lsb}(m) = 0$ then HW and HW' are related linearly.
  - If $\text{lsb}(m) = 1$ then the probability of a random linear relationship is very low.
    - **From here we get lsb(m).**

Converting PIO_{PA} to a PIO_{lsb} (2)

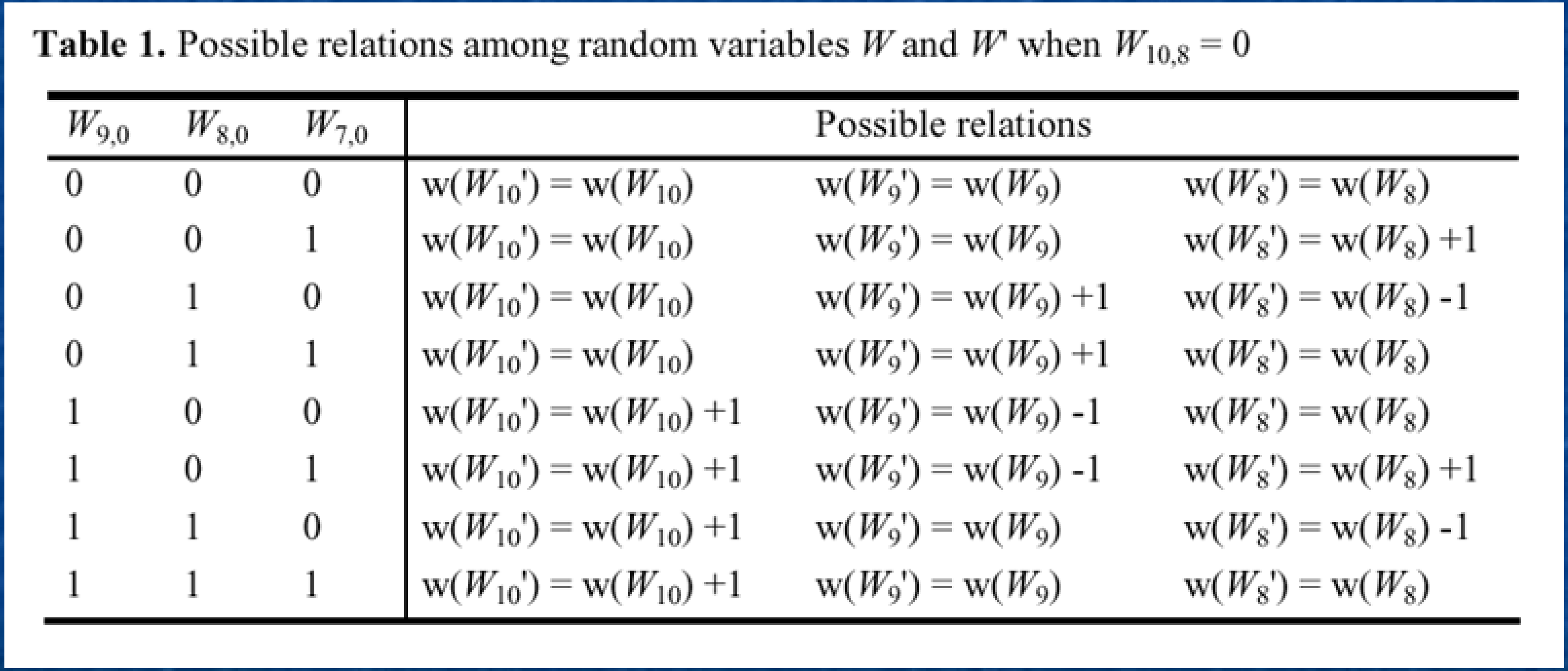


Table 1. Possible relations among random variables W and W' when $W_{10,8} = 0$

$W_{9,0}$	$W_{8,0}$	$W_{7,0}$	Possible relations		
0	0	0	$w(W_{10}') = w(W_{10})$	$w(W_9') = w(W_9)$	$w(W_8') = w(W_8)$
0	0	1	$w(W_{10}') = w(W_{10})$	$w(W_9') = w(W_9)$	$w(W_8') = w(W_8) +1$
0	1	0	$w(W_{10}') = w(W_{10})$	$w(W_9') = w(W_9) +1$	$w(W_8') = w(W_8) -1$
0	1	1	$w(W_{10}') = w(W_{10})$	$w(W_9') = w(W_9) +1$	$w(W_8') = w(W_8)$
1	0	0	$w(W_{10}') = w(W_{10}) +1$	$w(W_9') = w(W_9) -1$	$w(W_8') = w(W_8)$
1	0	1	$w(W_{10}') = w(W_{10}) +1$	$w(W_9') = w(W_9) -1$	$w(W_8') = w(W_8) +1$
1	1	0	$w(W_{10}') = w(W_{10}) +1$	$w(W_9') = w(W_9)$	$w(W_8') = w(W_8) -1$
1	1	1	$w(W_{10}') = w(W_{10}) +1$	$w(W_9') = w(W_9)$	$w(W_8') = w(W_8)$

Closure: Building WIO from $\mathrm{PIO}_{\mathrm{lsb}}$

- $\mathrm{PIO}_{\mathrm{lsb}}$ seems to be on of those best oracles suitable for building up an efficient WIO.
- We suggest the *RSA inversion* algorithm
  - c.f. [10, p. 226]: Fischlin, R. and Schnorr, C. P.: *Stronger Security Proofs for RSA and Rabin Bits*, Journal of Cryptology, Vol. 13, No. 2, pp. 221-244, IACR, 2000
  - This algorithm is based on a binary halving technique, errors are corrected through a majority decision.
  - It requires $O(\mathrm{L}(n)^2 adv^{-2})$ oracle calls, where:
    - $\mathrm{L}(n)$ is the length of an RSA modulus,
    - *adv* is the oracle advantage in the lsb prediction.
  - Post-processing complexity is $O(\mathrm{L}(n)^2 adv^{-6})$.
    - Doesn’t require further oracle access.

PART II

- Fault side channel attacks on RSA-KEM

Confirmation Oracle (CO)

- For arbitrary chosen integers r, y confirms whether $r = y^d \bmod n$.
  - Can be generalized for any encryption scheme (the condition tested may be also more general). Here we use RSA-CO only.
- If there <u>are faults</u> then an RSA-CO may reveal nontrivial information about the private key.
  - Depends on what kind of error it is and on a description accuracy.

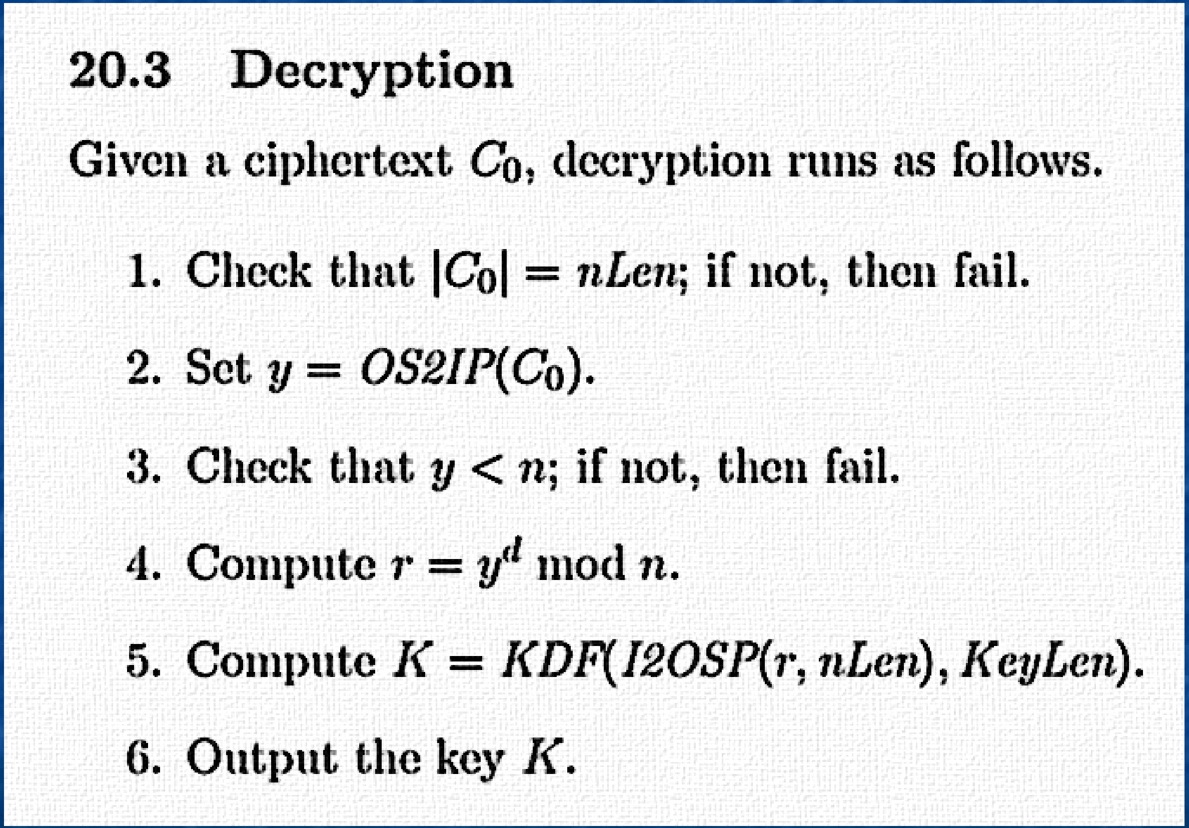
Example – Bit Errors

- In [3] it was shown that bit errors in a private exponent *d* enable us to compute the whole private key efficiently.
- We observed that it is not necessary to have an access to the whole result of the faulty computation.
  - Having an access to an RSA-CO is enough.
  - RSA-CO seems to be an effective generalization of some existing (as well as new ones) attacks developed with the assumption that an attacker could observe the whole output of an RSA computation (i.e. corrupted plaintext or signature).
- Bit errors are becoming even more interesting due to *Optical Fault Induction Attacks*.
  - Sergei Skorobogatov and Ross Anderson: *Optical Fault Induction Attacks*, here on CHES 2002.

Modified Pohlig-Hellman Algorithm

- This well known algorithm can also be modified to use only an access to an RSA-CO.
  - Suppose that we want to use it to compute the private exponent d from the triplet (r, g, n'), such that $r = g^d \bmod n'$.
    - Suppose that an attacker is able to force the change of modulus n to modulus n' inducing weak instances of DLP.
  - We show, that it is not necessary to know the value of r exactly – having an access to an RSA-CO$_{d,\, n'}$ is sufficient here.

RSA-KEM ([23])



20.3 Decryption

Given a ciphertext C_0, decryption runs as follows.

1. Check that $|C_0| = nLen$; if not, then fail.
2. Set $y = OS2IP(C_0)$.
3. Check that $y < n$; if not, then fail.
4. Compute $r = y^d \bmod n$.
5. Compute $K = KDF(I2OSP(r, nLen), KeyLen)$.
6. Output the key K.

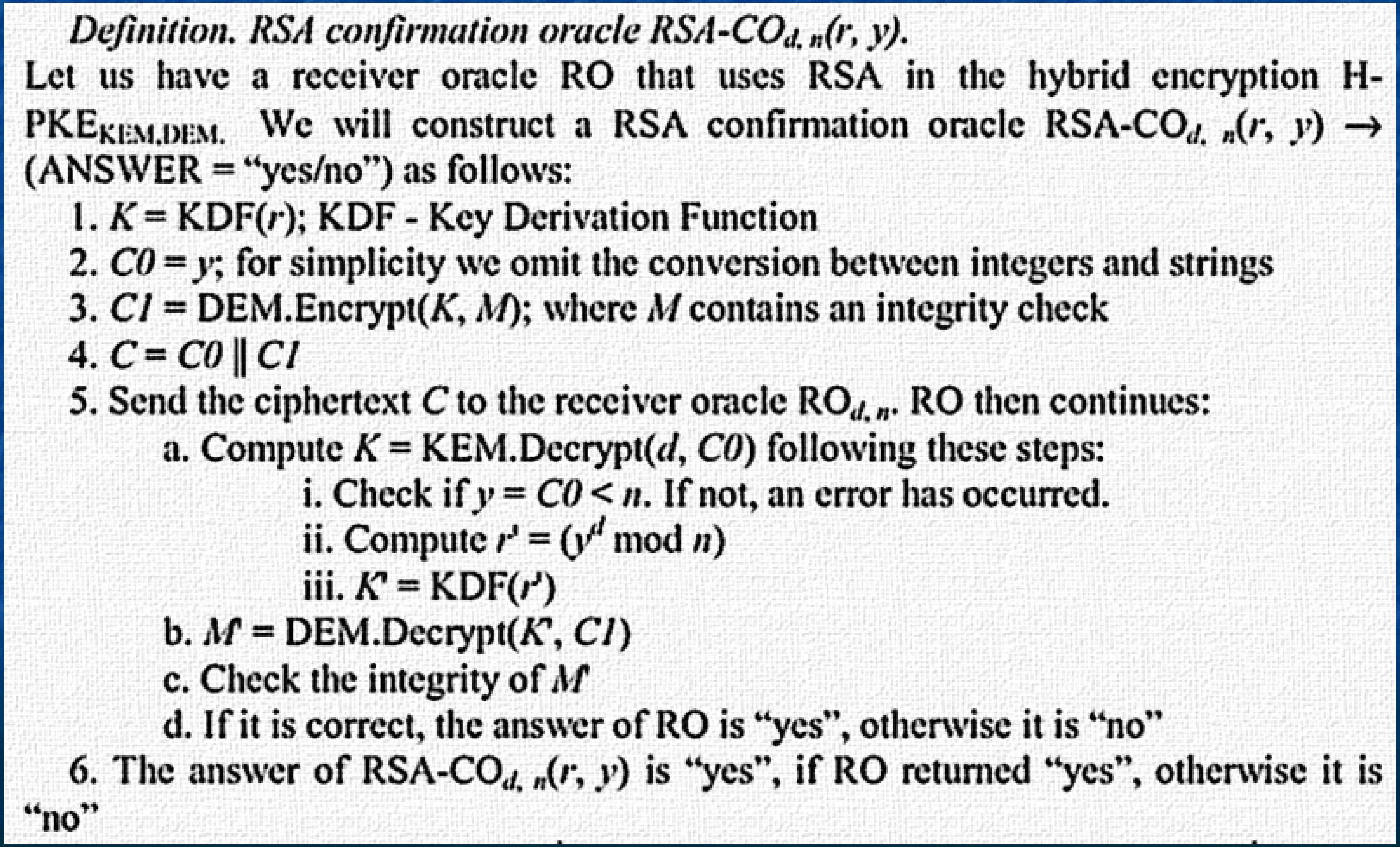
- Brief review of a RSA-KEM based H-PKE: the decryption phase

...then continue as: M = DEM.Decrypt(K, L, DEM.Encrypt(K, L, M)), M ~ message, L ~ label, K ~ symmetrical key

Building RSA-CO on RSA-KEM Properties

- We use the properties of the whole hybrid scheme H-PKE.
  - There is no integrity check for the RSA plaintext (*r*).
    - This is obviously good property against CCA_2, however it also implies that any <u>resulting RSA plaintext will be used for a symmetrical decryption</u>.
    - Further integrity controls applied on symmetrically decrypted message then <u>confirms our guess of *r*</u>.
  - Summary: What makes the RSA-KEM stronger in other areas, that makes it very vulnerable to a fault attacks.

RSA-CO Definition



Definition. RSA confirmation oracle RSA-CO$_{d, n}$(r, y).

Let us have a receiver oracle RO that uses RSA in the hybrid encryption H-PKE$_{\text{KEM,DEM}}$. We will construct a RSA confirmation oracle RSA-CO$_{d, n}(r, y) \rightarrow$ (ANSWER = "yes/no") as follows:

1. K = KDF(r); KDF - Key Derivation Function
2. $C0 = y$; for simplicity we omit the conversion between integers and strings
3. $C1$ = DEM.Encrypt(K, M); where M contains an integrity check
4. $C = C0 \| C1$
5. Send the ciphertext C to the receiver oracle RO$_{d, n}$. RO then continues:
   a. Compute K = KEM.Decrypt(d, $C0$) following these steps:
      i. Check if $y = C0 < n$. If not, an error has occurred.
      ii. Compute $r' = (y^d \bmod n)$
      iii. K' = KDF(r')
   b. M' = DEM.Decrypt(K', $C1$)
   c. Check the integrity of M'
   d. If it is correct, the answer of RO is "yes", otherwise it is "no"
6. The answer of RSA-CO$_{d, n}$(r, y) is "yes", if RO returned "yes", otherwise it is "no"

Summary

- RSAES-OAEP can be attacked when Hamming weights of data processed leak to an attacker.
  - All steps in the OAEP decoding should be resistant to a side channel leakage.
- RSA-KEM has serious problems with fault attacks.
  - It could lead to a private key disclosure.
  - Integrity of private key must be ensured and the computation must be checked for faults.
- Note that Manger's and Bleichenbacher's attacks can be easily converted to compute signatures instead of plaintexts.
  - Server's certificate should not be attributed for a signature and a key exchange purposes at the same time – this helps to reduce potential attack impacts.

Thank You