Further Results and Considerations on Side Channel Attacks on RSA

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

[1] ICZ, V Olšinách 75, 100 97 Prague 10, Czech Republic, http://www.i.cz
[2] Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
{vlastimil.klima, tomas.rosa}@i.cz

May 24, 2002

Abstract. This paper contains three parts. In the first part we present a new side channel attack on plaintext encrypted by EME-OAEP PKCS#1 v.2.1. In contrast with Manger´s attack, we attack that part of the plaintext, which is shielded by the OAEP method. In the second part we show that Bleichenbacher's and Manger's attack on the RSA encryption scheme PKCS#1 v.1.5 and EME-OAEP PKCS#1 v.2.1 can be converted to an attack on the RSA signature scheme with any message encoding (not only PKCS). This is a new threat for those implementations of PKI, in which the roles of signature and encryption keys are not strictly separated. This situation is often encountered in the SSL protocol used to secure access to web servers. In the third part we deploy a general idea of fault-based attacks on the RSA-KEM scheme and present two particular attacks as the examples. The result is the private key instead of the plaintext as with attacks on PKCS#1 v.1.5 and v.2.1. These attacks should highlight the fact that the RSA-KEM scheme is not an entirely universal solution to problems of RSAES-OAEP implementation and that even here the manner of implementation is significant.

Category / Keywords: public-key cryptography / side channel attack, confirmation oracle, RSA-KEM, RSAES-OAEP, PKCS#1 v.1.5, PKCS#1 v.2.1, Bleichenbacher's attack, Manger's attack, power analysis, fault analysis.

1 Introduction

In 1998, Bleichenbacher [5] described an attack on the PKCS#1 v.1.5 encoding and in 2001 Manger [13] described an attack on the improved scheme PKCS#1 v.2.1, called also RSAES-OAEP. These attacks underline the significance of the theorem of RSA individual bits [11] which states that: *If RSA cannot be broken in a random polynomial time, then it is not possible to predict the value of any selected bit of the plaintext with a probability not negligibly different from 1/2.* A negligible difference for the purpose of this theorem is such $\varepsilon(n)$ that for any constant $c > 0$ it holds that $\varepsilon(n) < n^{-c}$, where n is the appropriate modulus of RSA. From the standpoint of side channels it is important to understand this theorem as saying: *If the value of any chosen bit of the plaintext can be predicted with a probability not negligibly different from 1/2 then RSA can be broken within a random polynomial time.* Breaking RSA [19] is understood here to mean that a value of the plaintext is obtained. Bleichenbacher's and Manger's attacks [5, 13] use side channels which provide the attacker with a relatively large amount of information about the plaintext (at least the two most significant bytes are 00 02 or one byte is 00, respectively).

In this paper plaintext will always mean a value of m which is created immediately after an operation with a private RSA key, $m = c^d \ mod \ n$, not the value of M obtained after decoding m.

In section 2 we present another realistic attack on the RSAES-OAEP (PKCS#1 v.2.1) scheme. It is a side channel attack using only the information about Hamming weight of certain 32-bit words produced in the process of decoding m by the EME-OAEP-DECODE procedure according to PKCS#1 v.2.1. Theoretically, it is a weakening of the assumptions of Manger's and Bleichenbacher's attacks. From the practical point of view, the new attack can be used especially on smart cards. It follows from the theorem of RSA individual bits that it is necessary to prevent the leakage of any information about the individual bits of the plaintext. Our attack demonstrates that the Hamming weight of a part of the plaintext can be used to carry out a successful attack.

In section 3 we present a very simple but efficient conversion of the Manger/Bleichenbacher breaking oracle to a universal (signature) oracle. The principle that a private RSA key should not be used simultaneously for encryption and for digital signature is well known but is very often violated in practice. Typical examples include some of the current implementations of PKI, the SSL protocol etc. We show that if we can perform Bleichenbacher's or Manger's attack on the encryption scheme using PKCS#1 (v.1.5 or v.2.1) in such way that we can obtain the plaintext then we can also obtain the digital signature of any message (encoded in any way) using the same private RSA key. In the SSL protocol this means the ability to create signatures with the server-side private key and even create false servers with the identity of the original server, provided that sufficient decrypting speed can be ensured.

In section 4 we present a new fault side channel attack on the RSA-KEM. RSA-KEM attempted to remove the structural relations in order to prevent leaking of information about the plaintext. Despite this we discovered a natural method of obtaining such information. Input plaintext for RSA-KEM consists of symmetric encryption keys, information about which can be obtained by means of an integrity check of the messages they encrypt. A typical integrity check consists of block cipher padding, e.g. PKCS#5 [16]. The result produced by the attack that uses this information is a private RSA key whilst the attacks on PKCS#1 v.1.5 and 2.1 always discovered only a plaintext.

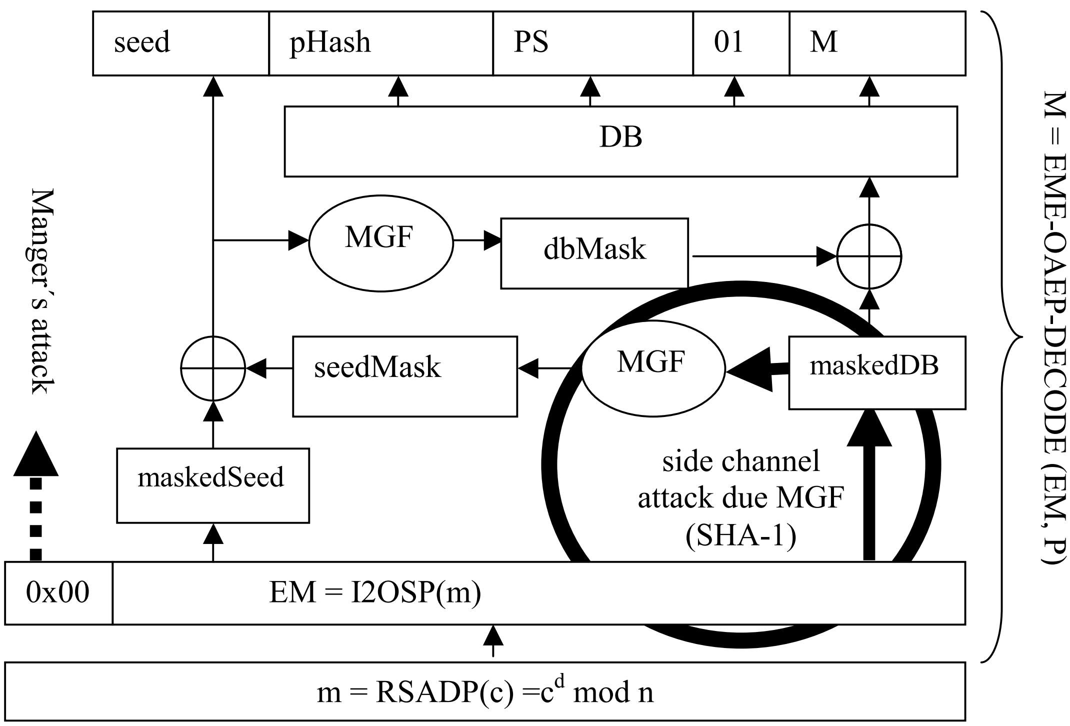


Fig.1: New side channel attack against RSA-OAEP

2 Side Channel Attack on RSAES-OAEP Plaintext

In this section we will demonstrate a new method of attacking the RSAES-OAEP scheme (PKCS#1 v.2.1 [15]) at the time when decoding operation EME-OAEP-DECODE (EM, P) is performed, see fig. 1. The attack is based on the assumption that there is a side channel carrying some information about the plaintext. In particular we assume that the attacker can obtain the Hamming weight $w(x)$ (i.e. the number of '1' bits) of a word x during the time when the plaintext (m) is being processed in the MGF operation (to be specified later). As it was shown in [14], this assumption is realistic for instance in power side channels which tend to leak this information in a relatively readable way.

We note that this attack is possible with some modifications even when we have access to the Hamming distance of processed data rather than the weight.

2.1 Attack Description

Consider RSA with a modulus n which has the length of N bits where N is the multiple of 512, i.e. $N = 512*k$, where k is a natural number. The attack will target the RSAES-OAEP scheme during the processing of the plaintext immediately after the RSA decryption operation $c^d \bmod n$, see fig. 1. *SeedMask* will be calculated according to [15] as

$$seedMask = MGF(maskedDB, 20) = SHA\text{-}1(maskedDB \,||\, 00\ 00\ 00\ 00),$$

where the four zero octets are appended to the message by the MGF function. It follows from the definition of OAEP encoding that *maskedDB* always contains $64*k - 1 - 20$ octets, so that $64*k - 17$ octets (4 extra zero octets) enter SHA-1. By the definition of SHA-1 [20] the message is divided into blocks of 64 octets, which are processed sequentially by the compression function. Note that the least significant bit of the original message m is processed in the last block. It is followed by four zero octets and 17 octets of the SHA-1 padding.

For various N the particular value of the padding is different, but it is a constant known to the attacker. To present an example, we will consider $N = 1024$.

Let us denote the i-th octet of the plaintext as $m[i]$ where $m[0]$ is the least significant octet. The last block entering the SHA-1 compression function is in this case equal to

$m[42.....0]\ 00\ ||\ 00\ 00\ 00\ 80\ ||\ 00\ 00\ 00\ 00\ ||\ 00\ 00\ 00\ 00\ ||\ 00\ 00\ 00\ 00\ ||\ 00\ 00\ 03\ 78,$

where m is followed by 4 zero octets (from MGF) and the SHA-1 padding. The padding consists of bit 1, 71 zero bits and a 64-bit representation of the message bit length. The length is $888 = 0x\ 00\ 00\ 00\ 00\ 00\ 00\ 03\ 78$ bits in this case ($64*2 - 17 = 111$ octets). The SHA-1 compression function fills this last block into 32-bit variables $W_0, ..., W_{15}$, where

$W_8 = m[10]\ m[9]\ m[8]\ m[7]$	$W_9 = m[6]\ m[5]\ m[4]\ m[3]$
$W_{10} = m[2]\ m[1]\ m[0]\ 00$	$W_{11} = 00\ 00\ 00\ 80$
$W_{12} = 00\ 00\ 00\ 00$	$W_{13} = 00\ 00\ 00\ 00$
$W_{14} = 00\ 00\ 00\ 00$	$W_{15} = 00\ 00\ 03\ 78$

And then expansion to words $W_{16}, ..., W_{79}$ is performed according to the following relations

$$W_{16} = S^1(W_{13}\ xor\ \mathbf{W_8}\ xor\ W_2\ xor\ W_0),$$
$$W_{17} = S^1(W_{14}\ xor\ \mathbf{W_9}\ xor\ W_3\ xor\ W_1),$$
$$W_{18} = S^1(W_{15}\ xor\ \mathbf{W_{10}}\ xor\ W_4\ xor\ W_2),\ \text{etc.}$$

When calculating W_{16}, the first operation performed is $W_{13}\ xor\ W_8$, where W_{13} is a known constant. This moment is an example of a general situation when D-1 known parameters and one unknown enter a D-ary operation. Here various side channels are often applicable, especially the power side channel.

We assume that the attacker is able to gather the Hamming weight $\mathbf{w(W_8)} \in \{0, ..., 32\}$ of word W_8 during the $W_{13}\ xor\ W_8$ operation (W_8 is the only unknown operand in it). The same situation arises in the following two operations as well, so we are able to gather $\mathbf{w(W_9)}$ and $\mathbf{w(W_{10})}$.

We number the bits of the word W_i (from the msb to the lsb) as $W_{i,31}\ W_{i,30}\ W_{i,29} ...\ W_{i,0}$. We will show that now we can predict the value of $W_{10,8}$ with a probability not negligibly different from $1/2$. Note that this is the value of the least significant bit (lsb) of the plaintext. Hence, using the theorem of RSA individual bits [11] we can design an attack on the entire plaintext. It is widely known that information about the lsb of the plaintext leads to very efficient attacks [23].

2.2 Obtaining the Least Significant Bit of the Plaintext

The procedure which leads to obtaining the value of $W_{10,8}$ is as follows. We denote the ciphertext to be attacked by c, the modulus by n and the public RSA exponent by e. First we let the attacked device decrypt and decode the original ciphertext c. During decoding we gather the values of Hamming weights $w(W_8)$, $w(W_9)$ and $w(W_{10})$. In the next step we request the equipment to decrypt and decode a value $c' = c*2^{-e} \bmod n$. Plaintext m' is the result of this and during the calculation we will obtain Hamming weights $w(W_8')$, $w(W_9')$ and $w(W_{10}')$. If the $W_{10,8}$ bit is zero, then the decryption returns the value $m' = m >> 1$, where "$>> 1$" means a shift one bit to the right. Otherwise $m' = (m + n) >> 1$.

If we assume $W_{10,8} = 0$ then (W_8', W_9', W_{10}') will be created of (W_8, W_9, W_{10}) by a shift one bit to the right (with the exception of W_{10}, where the shift only affects the leftmost bits which are then independently complemented by eight zero bits). The difference between appropriate Hamming weights $(w(W_8), w(W_9), w(W_{10}))$ and $(w(W_8'), w(W_9'), w(W_{10}'))$ is therefore 0 or 1. More precisely

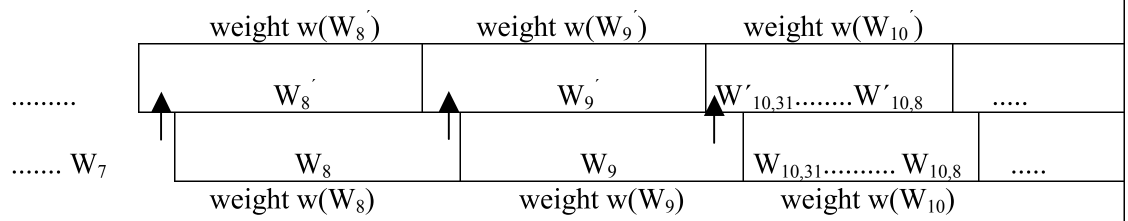
$w(W_8') = w(W_8) - W_{8,0} + W_{7,0}$, $w(W_9') = w(W_9) - W_{9,0} + W_{8,0}$, $w(W_{10}') = w(W_{10}) - W_{10,8} + W_{9,0} = w(W_{10}) + W_{9,0}$ and therefore the three relations included in exactly one of the eight rows of table 1 are valid.

However, if $W_{10,8} = 1$, m' is not created by a shift of m, but produced as $(m + n) >> 1$. This, with a high probability, destroys the linear relations in the table 1. By the obtained weights $(w(W_8), w(W_9), w(W_{10}))$ and $(w(W_8'), w(W_9'), w(W_{10}'))$ we determine whether they fit all relations in any single row. If so, we adopt a hypothesis that $W_{10,8} = 0$, otherwise we refuse it and assume that $W_{10,8} = 1$. The probability of establishing the bit $W_{10,8}$ correctly is close to 1 for an ideal side channel. It will be sufficient to realise that m is randomised by a hash function in MGF and n is assumed to be common, not specially constructed. Therefore, the probability of adopting the hypothesis that $W_{10,8} = 0$ if it was $W_{10,8} = 1$, can be estimated as the probability that the random variables W_8, W_9, W_{10} and W_8', W_9', W_{10}' (with the properties that lower nine bits of W_{10} are 100000000 and lower eight bits of W_{10}' are 00000000) will fit any of the relations in table 1, which is approximately 0.008.

That enables us to obtain the least significant bit of the plaintext m with a high probability and therefore, in accordance with [11] we can establish the remaining part of m. We presume that procedures in [11] will be used directly, in particular the methods based on computing gcd (for details see [2]). In this way we are able to handle errors during the reception of information from the side channel.

In this paper we strive to show that such an attack is realistic and that it operates in a random polynomial time, following from the above analysis and the results of [2, 11]. We would like to emphasize the importance of a thorough implementation. The significance of the implementation stage cannot simply be reduced to the problem of finding “the right encoding method” as was perhaps deemed earlier.

In practice the described attack can be further modified with respect to what information (Hamming weight or distance) at what level of accuracy the attacker can obtain. At a low level of side channel interference special breaking methods based on the knowledge of the lsb [23] can be applied. Those will obviously be more efficient than the general ones of [2, 11].



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

Tab.1: Possible relations among random variables W and W´ when $W_{10,8} = 0$

3 Note on Converting the Deciphering Oracle to a Signing Oracle

In this section we will demonstrate that if the attacker can use Bleichenbacher's or Manger's attack on the PKCS#1 v.1.5 or 2.1 encryption scheme, he/she is also able to create false signatures using the same private RSA key with any encoding of the message to be signed. This conversion is technically very simple but it has interesting practical consequences on the applications where the same key is used both for encryption and for digital signature. One example is the SSL protocol used to secure access to web servers. In its application the public key certificate at the server sometimes permits the use of the key both for encryption and for signature. That means that a signature made by the server's private key is meaningful in the PKI system and it is not appropriate that it should be forgeable. Conversion will be demonstrated for both Bleichenbacher's attack on PKCS#1 v.1.5 and for Manger's attack on PKCS#1 v.2.1.

Manger's attack uses only one element of the EME-OAEP PKCS#1 v.2.1 encoding - whether a zero occurred in the most significant octet (MSB) of the plaintext decrypted by the private key. We will denote the oracle which tells the attacker this as "Partial information oracle" $\mathrm{PIO}_{\mathrm{MSB}}$:

$$PIO_{MSB}(c) = \text{"yes" } iff\ c = m^e \bmod n,\ MSB(m) = 0x00.$$

Using this oracle a decryption machine (Whole information oracle) $\mathrm{WIO}_{\mathrm{MSB}}$ is constructed in [13]. If the plaintext has a format of $m = 00 \,||\, ...$, then the $\mathrm{WIO}_{\mathrm{MSB}}$ (using $\mathrm{PIO}_{\mathrm{MSB}}$) can extract from the ciphertext *c* the original plaintext

$$m = WIO_{MSB}(c) = c^d \bmod n.$$

Now, we will assume that the same private key (*d*) is used in another RSA scheme (with any encoding) for digital signature. The attacker can now easily forge the digital signature of any message using the same private key (*d*) if he/she has access to $\mathrm{PIO}_{\mathrm{MSB}}$. Let *c* be the message that the attacker prepares for signing. He/she then randomly selects various natural numbers $r = r_1, r_2, ...$ different from one another, smaller than *n* and sends numbers $c' = c*r^e \bmod n$ to the oracle $\mathrm{PIO}_{\mathrm{MSB}}$ successively. After decryption $m' = m*r \bmod n$ is produced on the recipient's side. Unless the most significant octet of m' is zero, it is rejected by $\mathrm{PIO}_{\mathrm{MSB}}$:

$$PIO_{MSB}(c') = \text{"no"}.$$

Because the random value *r* produces a random most significant octet in m', this octet will be zero with a probability of *1/256*. After several hundreds of trials the value of c' will conform with the initial condition of Manger's attack and $\mathrm{WIO}_{\mathrm{MSB}}$ then decrypts c':

$$m' = WIO_{MSB}\,(c') = (c')^d \bmod n.$$

The attacker then only has to calculate

$$m = m' * r^{-1} \bmod n$$

as a valid signature of the message *c*. The particular type of encoding for a signature is irrelevant here.

The attacker follows the same procedure when converting Bleichenbacher's attack. This attack assumes the oracle $\mathrm{PIO}_{\mathrm{PKCS\_CONF}}$, which tells the attacker whether the plaintext produced by decryption is "PKCS#1 conforming" [5]. That means that the two most significant octets of the plaintext must be equal to *00 || 02* and from the 11th octet onwards some octet must be zero (separator). On the basis of $\mathrm{PIO}_{\mathrm{PKCS\_CONF}}$ a decryption machine $\mathrm{WIO}_{\mathrm{PKCS\_CONF}}$ is then constructed. If the plaintext is "PKCS#1 conforming", then $\mathrm{WIO}_{\mathrm{PKCS\_CONF}}$ can use $\mathrm{PIO}_{\mathrm{PKCS\_CONF}}$ on the corresponding ciphertext *c* to obtain the original plaintext

$$m = WIO_{PKCS\_CONF}(c) = c^d \bmod n.$$

Using the same procedure as above, i.e. by a randomly selected *r*, we test whether $\mathrm{PIO}_{\mathrm{PKCS\_CONF}}$ on $c' = c*r^e \bmod n$ responds "yes". This time the probability of such answer is several hundred times lower than in the case of Manger's attack (depending on the number of bits of n; for 1024 it is approximately 715-times less, see [13]). As soon as such a situation occurs, the attacker can again compute $m = m' * r^{-1} \bmod n$ as a valid signature of the message c.

Note that the attack described in section 2 of this paper does not place any special requirements on the ciphertext. It is therefore suitable for forging signatures even without any changes.

4 Side Channel Attack on RSA-KEM

After Bleichenbacher's attack on the scheme PKCS#1 v.1.5, a new scheme PKCS#1 v.2.1, based on the EME-OAEP encoding, was recommended for use. However, Manger's attack [13] showed that RSAES-OAEP is also vulnerable to side channel attacks. After that Shoup [21] proposed the new key encapsulation mechanism RSA-KEM. This mechanism was believed to have eliminated problems with side channels. We show that RSA-KEM is also vulnerable to some types of side channel attacks, and therefore has to be implemented carefully. In these attacks, the private key may be obtained. In this sense, the RSA-KEM mechanism is even more vulnerable to side channel attacks than PKCS#1. Next we will describe an RSA confirmation oracle (CO) based on RSA-KEM. We will show how to use a CO to obtain a RSA private key.

4.1 Confirmation Oracle

In this text, we are using the terminology of [21], except for the term RSAES-OAEP that is defined in PKCS#1 v.2.1. The purpose of RSA-KEM is to transmit the symmetric key to the receiver, and so it is natural to consider the properties of the whole hybrid public-key encryption scheme $\text{H-PKE}_{\text{KEM, DEM}}$, consisting of the Data Encapsulation Mechanism (DEM) and the Key Encapsulation Mechanism (KEM). Our attack on RSA-KEM is based on the behaviour of the entire hybrid scheme. Later we will see that our requirements are sufficiently general and make it easily realizable in practical applications. We will start by reviewing some important terms (algorithms) from [21] in a simplified form:

The Key Encapsulation Mechanism (KEM) has this abstract interface:

KEM.Encrypt(PubKey) → (K, C0) - generates a symmetric encryption key *K* and using the public key *PubKey*, creates a corresponding ciphertext *C0*

KEM.Decrypt(PrivKey, C0) → (K) - decrypts *C0* using the private key *PrivKey*, and derives the symmetric key *K* by applying the key derivation function KDF to that result

The Data Encapsulation Mechanism (DEM) has this abstract interface:

DEM.Encrypt(K, M) → (C1) - encrypts the message *M* with the symmetric key *K* and returns the corresponding ciphertext *C1*

DEM.Decrypt(K, C1) → (M) - decrypts the ciphertext *C1* with the symmetric key *K* and returns the plaintext *M*

The hybrid public-key encryption scheme $\text{H-PKE}_{\text{KEM, DEM}}$ is a combination of the KEM and DEM schemes. The algorithm for the encryption of a message *M* by the public key *PubKey* resulting in the ciphertext *C* is as follows:

1. *(K, C0) = KEM.Encrypt(PubKey)*
2. *C1 = DEM.Encrypt(K, M)*
3. *Ciphertext C = C0 || C1*

On the receiving end, the decryption of the ciphertext *C* with the private key *PrivKey* is carried out as follows:

1. *Let C = C0 || C1*
2. *K = KEM.Decrypt(C0)*
3. *M = DEM.Decrypt(K, C1)*

We assume that there is **no integrity check for the key *K*** (e.g. analogous to a check used in the encoding method OAEP) however **an integrity check exists for the message *M*** in step 3. It can be based on the message padding check, as in the standard PKCS#5 [16], on the use of labels as described in [21], or on any other technique. We assume the attacker will find out whenever the receiver's integrity check rejects a ciphertext *C*. In this situation we can expect that the receiver will send an error message to the sender. Acceptance or rejection of a ciphertext *C* defines the *receiver oracle* (RO). On the basis of RO we can define the confirmation oracle (CO). The term CO may be defined more generally, however, we will only define the RSA confirmation oracle (RSA-CO).

We assume that the private key *PrivKey* is a private exponent *d* and *n* is a public modulus. Later we will show that the modulus *n* should be part of the private key rather than independently taken from the public key, as it is recommended in [21].

Definition: RSA confirmation oracle $\boldsymbol{RSA\text{-}CO_{d,n}(r, y)}$. Let us have a receiver oracle RO that uses RSA in the hybrid encryption $H\text{-}PKE_{KEM,\ DEM.}$ We will construct a RSA confirmation oracle

$RSA\text{-}CO_{d,n}(r, y) \rightarrow (ANSWER = \text{“yes/no”})$ 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 the maximum possible integrity check
4. *C = C0 || C1*
5. Send the ciphertext *C* to the receiver oracle RO. 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 $\boldsymbol{r' = (y^d \bmod n)}$
        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 $RSA\text{-}CO_{d,n}(r, y)$ is “yes”, if RO returned “yes”, otherwise it is “no”

We note that whenever $r = (y^d \bmod n)$, the oracle returns “yes”. If $r \neq (y^d \bmod n)$ then the oracle returns “no” with a high probability close to 1 (the value depends on collisions in the function KDF and the strength of the integrity check).

The key point is that an attacker may use the oracle $\boldsymbol{RSA\text{-}CO_{d,n}(r, y)}$ to check the congruence $r \equiv y^d \pmod n$ without knowledge of the particular value of the private key *d* used in the step 5.a.ii above.

4.2 Fault Side Channel Attacks

The congruence $r \equiv y^d \pmod n$ can be confirmed with the public key. However, using $RSA\text{-}CO_{d,n}(r, y)$ is the natural way of exploiting the receiver's behaviour. The oracle becomes far more interesting when an error occurs in step 5.a.ii of the algorithm above. This confirmation oracle can be used to design many attacks. Therefore we will only present a brief description of two examples to illustrate the core of this problem. We note that these attacks are targeted at the private key, rather than the plaintext. This is paradoxically caused by the absence of structural checks of the plaintext in RSA-KEM, which is really a positive quality in other contexts.

4.2.1 Faults in the Bits of the Private Exponent d

The impact of faults in the bits of the private exponent RSA was described in [3]. We will show that the confirmation oracle $RSA\text{-}CO_{d,n}$ can be used to mount these attacks on the hybrid encryption scheme based on RSA-KEM. As an example we will assume that the attacker is able to swap the i-th bit $d(i)$ of the receiver's private exponent d (in step 5.a.ii), and this change will go undetected by the receiver. Such a situation can occur with chip cards.

Let us assume that a fault occurred in the i-th bit $d(i)$ and let us denote by d' the defect value of the private exponent. Depending on the value of $d(i)$, either $d' = d+2^i$ or $d' = d-2^i$. Let $I = 2^i$, $\alpha \equiv y^I \ (mod\ n)$ and $\alpha * \alpha^{-1} \equiv 1\ (mod\ n)$. For the value $r = y^{d'} \ mod\ n$ we have:

- $r = (y^d * \alpha\ mod\ n)\ if\ d(i) = 0$
- $r = (y^d * \alpha^{-1}\ mod\ n)\ if\ d(i) = 1$

Using the access to the confirmation oracle $RSA\text{-}CO_{d',n}$ we can find out the value of $d(i)$ in this way:

1. Randomly pick x, $0 < x < n$
2. Let $y = x^e\ mod\ n$, where e is the corresponding public exponent RSA
3. Let $r = x * \alpha\ mod\ n$
4. *If* $RSA\text{-}CO_{d',n}(r, y)$ returns "yes" then set $d(i) = 0$ else set $d(i) = 1$.

We can repeat this procedure for various bit positions (and their combinations) and thus obtain the whole private key d. In the case of irreversible changes we will gradually carry out an appropriate correction in step 3 using the previously obtained bits. In this way the corruption of d is allowed to be irreversible. Moreover, it is enough to obtain only a part of d from which the remaining bits can be computed analytically in a doable time, see overview in [6]. In [3, 7] we may find other sophisticated attacks of this type. We have presented the confirmation oracle as an "interface" that allows the attacker to apply some general attacks on "unformatted RSA" to RSA-KEM.

4.2.2 The Usage of Trojan Modulus

We have mentioned that in the RSA-KEM scheme, the modulus n is not part of the private key. This would allow for a change of the modulus n without any security alarm. The following attack shows the need to change this set up.

Let us assume that we can obtain the value $r = g^d\ mod\ n'$ for an unknown exponent d and any given values of g and n'. It is widely known that one such value r is sufficient to discover d. We can, for instance, choose a modulus n' to be a prime in the form $n' = t*2^s + 1$, where t is a very small prime number and s is a very large natural number. Further we choose g to be a generator of the multiplicative group $\boldsymbol{Z}_{n'}^{*}$.

Now we can solve the discrete logarithm problem in $\boldsymbol{Z}_{n'}^{*}$ by a simple modification of the Pohlig-Hellman algorithm [17]. This algorithm requires the value of $g^d\ mod\ n'$ directly, which we cannot obtain from the confirmation oracle. We can only ask the oracle whether the pair (r, g) satisfies the congruence $r \equiv g^d\ mod\ n'$. On a closer look at the Pohlig-Hellman algorithm we notice that it can be modified so that the value of r is not needed directly, but only in comparisons of the type $x =^? (r^\alpha\ mod\ n')$ for some integers x, α. If we substitute $(g^\alpha)^d\ (mod\ n')$ for r^α, we want to know whether $x =^? ((g^\alpha)^d\ mod\ n')$, which can be obtained from the confirmation oracle

$RSA\text{-}CO_{d,n'}(x, g^{\alpha} \bmod n')$. This is the main idea of the modification. The complete algorithm A1 is presented in the next subsection.

This attack is also possible even if the modulus n is part of the private key. However in this case we can expect that it will be a little bit more difficult to plant a false value of n'.

This idea can also be extended to the case when a method based on the Chinese Remainder Theorem is used for operations with the private key.

4.2.3 Algorithm A1: Computation of the Private Exponent Using the Access to a Confirmation Oracle RSA

In the following we will describe an efficient algorithm for a private exponent d computation, based on a modified Pohlig-Hellman algorithm for the solving the discrete logarithm problem in the multiplicative group $\mathbf{Z}_p^*$. This group has a special structure chosen by an attacker, because the value of p is taken to be the fraudulent modulus n'.

Proposition. *Let us assume to have access to a confirmation oracle* $RSA\text{-}CO_{d,p}$ *, where* p *is a prime such that* $p = t*2^s + 1$ *and* t *is a small prime. Let* g *be the generator of* $\mathbf{Z}_p^*$*. The following procedure computes the private exponent d. We note that the order of* $\mathbf{Z}_p^*$ *has to be larger than the highest possible value of d*

Step 1: Computation of the value $D_s = d \bmod 2^s$

Let $d = d(b-1)*2^{b-1} + d(b-2)*2^{b-2} + ...+ d(0)$, where b is the number of bits of the binary form of d, and $d(i) \in \{0,1\}$, for $0 \le i \le b-1$. We assume that $p-1$ is divisible by 2^i and we define $r = g^d \bmod p$ and $D(i) = d \bmod 2^i$. Let $I = 2^i$ and $J = 2^j$. Then

$r^{(p-1)/I} \equiv [g^d]^{(p-1)/I} \equiv [g^{(p-1)/I}]^d \equiv [g^{(p-1)/I}]^{d \bmod I} \equiv [g^{(p-1)/I}]^{D(i)} \pmod p$, and hence

$$r^{(p-1)/I} \equiv [g^{(p-1)/I}]^{D(i)} \pmod p. \qquad (1)$$

The value of $D(i)$ can be expressed as $D(i) = d(i-1)*2^{i-1} + d(i-2)*2^{i-2} + ...+ d(0)$. We will show that having access to the confirmation oracle we can easily compute the lowest s bits of the private exponent d (one bit of d per one oracle call).

We will start with the lowest bit $d(0)$ and inductively extend to the bit $d(s-1)$. For $i = 1$ from (1) we have $r^{(p-1)/2} \equiv [g^{(p-1)/2}]^{d(0)} \pmod p$. From the definition of r we have

$r^{(p-1)/2} \equiv [g^{(p-1)/2}]^{d} \pmod p$, and so

$$[g^{(p-1)/2}]^{d} \equiv [g^{(p-1)/2}]^{d(0)} \pmod p. \qquad (2)$$

We note that $g^{(p-1)/2} \equiv p-1 \pmod p$, and $[g^{(p-1)/2}]^{d(0)} \bmod p$ can achieve only two possible values, depending on the bit $d(0)$. Using the confirmation oracle, we can either confirm or refute the value of $d(0)$ in (2). Let $d(0) = 1$ and call the oracle in the form $RSA\text{-}CO_{d,p}(p-1, p-1)$, which represents the congruence (2). If the oracle returns "yes" we set $d(0) = 1$, otherwise we set $d(0) = 0$. We note that a correctly generated private exponent RSA should induce $d(0) = 1$, therefore this step can be omitted.

We determine the remaining bits of $D(s)$ inductively. We assume that we know the value $D(j)$ for some $0 < j < s$. Next we will compute the value $D(j+1)$. From (1) we have

$$r^{(p-1)/(2J)} \equiv [g^{(p-1)/(2J)}]^{D(j+1)} \pmod p. \qquad (3)$$

Let $\alpha = d(j) * 2^j = d(j) * J$. Then $D(j+1) = d \bmod 2^{j+1} = \alpha + D(j)$. For the value on the right-hand side of (3) we have that

$[g^{(p-1)/(2J)}]^{D(j+1)} \equiv [g^{(p-1)/(2J)}]^{\alpha} * [g^{(p-1)/(2J)}]^{D(j)} \equiv [g^{(p-1)/2}]^{d(j)} * [g^{(p-1)/(2J)}]^{D(j)} \equiv$

$\equiv (p-1)^{d(j)} * [g^{(p-1)/(2J)}]^{D(j)} \pmod p$, so we get $r^{(p-1)/(2J)} \equiv (p-1)^{d(j)} * [g^{(p-1)/(2J)}]^{D(j)} \pmod p$. Using the definition of r ($r = g^d \bmod p$) we obtain:

$$[g^{(p-1)/(2J)}]^{d} \equiv (p-1)^{d(j)} * [g^{(p-1)/(2J)}]^{D(j)} \pmod p. \qquad (4)$$

On the right-hand side of (4), almost entirely known values appear, with the exception of the value of $d(j)$. We will again use the confirmation oracle to decide between the two possible values of the bit $d(j)$. We guess that $d(j) = 0$ and call the oracle in the form $RSA\text{-}CO_{d,\,p}(\ [g^{(p-1)/(2J)}]^{D(j)}\ mod\ p,\ g^{(p-1)/(2J)}\ mod\ p)$, which represents the congruence (4). If the oracle returns "yes", we set $d(j) = 0$, otherwise we do the correction $d(j) = 1$. The inductive step is finished and we have obtained $D_s = D(s)$.

Step 2: Computation of the value $D_t = d\ mod\ t$

It is simple to show that an integer j, under the condition $r^{(p-1)/t} \equiv [g^{(p-1)/t}]^j\ (mod\ p)$, satisfies that $D \equiv j\ (mod\ t)$. Whenever $j < t$, then we directly obtain that $D_t = j$. Therefore we can identify the value D_t in this step by testing every number $j = 0, ..., t-1$, until we find the j that satisfies the congruence $r^{(p-1)/t} \equiv [g^{(p-1)/t}]^j\ (mod\ p)$. This j is then the sought value of D_t. In order to determine this value we rewrite the congruence (using the definition of r) as follows:

$$[g^{(p-1)/t}]^d \equiv [g^{(p-1)/t}]^j\ (mod\ p) \tag{5}$$

and use the oracle in the form $RSA\text{-}CO_{d,\,p}(\ [g^{(p-1)/t}]^j\ mod\ p,\ g^{(p-1)/t}\ mod\ p)$ gradually for $j = 0,..., t-1$. The correct value of j is reached when the oracle returns "yes" and we set $D_t = j$.

Step 3: Computation of the value d

In the previous steps we have obtained two congruencies:

- $d \equiv D_s\ (mod\ 2^s)$
- $d \equiv D_t\ (mod\ t)$

It also holds that $gcd(t, 2^s) = 1$, and so by the Chinese Remainder Theorem, there exists a single value $0 \le d < t*2^s$, satisfying both congruencies. The value of d can be computed directly as bellow:

1. Compute γ, $\gamma*2^s \equiv 1\ (mod\ t)$, a unique value exists because $gcd(t, 2^s) = 1$
2. Compute $v = (D_t - D_s)*\gamma\ mod\ t$
3. $d = D_s + v*2^s$

4.2.4 Other Computational Faults

So far we have only considered the attacks based on modifications of the private exponent d and the modulus n. However, similar attacks may be developed, considering general permanent or transient faults that appear during RSA computations within the function KEM.Decrypt. A discussion on these attacks, however, is beyond the scope of this paper. For more details, the reader may consult papers [3, 7]. We can realistically assume that certain types of attacks described there can be used on RSA-KEM with the use of the confirmation oracle. Some additional research can be done on this subject.

4.2.5 Comparison of attacks on RSA schemes

We recapitulate that from the theorem about RSA individual bits [11], all implementations of the cryptosystem RSA have to be carried out with caution because of side channels. Manger [13] showed that the RSAES-OAEP scheme has certain problems with the most significant octet. These problems must be avoided by proper implementation. We have shown that RSA-KEM has similar problems, when fault side channel attacks can occur. The attacker has the possibility of disclosing the value of the private key instead of the plaintext. Whenever we use RSA-KEM it is then essential to exclude fault side channels. We must carry out reliable private key integrity checks (the modulus should be a natural part of the private key) as well as using fault tolerant computations. We still need to consider the consequences of the RSA individual bit theorem and make sure that no information about any individual bit of the plaintext has leaked. Table 1 below contains a brief overview of the current state of most used RSA schemes when side channel attacks are considered.

	PKCS1 v.1.5	RSAES-OAEP	RSA-KEM
Public attack	Yes	Yes	Yes
Side channel (information) used in attack	The information whether the plaintext is PKCS#1 v.1.5 conforming	• Information about whether the most significant octet of plaintext is zero • Hamming weight of processed data	Fault side channel
Information obtained in attack	Plaintext	Plaintext	Private key

4.3 General Countermeasures

When we consider the state-of-the-art in cryptanalysis, we can specify three basic security criteria that need to be satisfied in every cryptosystem design on the RSA basis. These are:

a) Resistance to adaptive chosen ciphertext attacks
b) Resistance to side channel information leakage
c) Resistance to fault side channels

Imperfect resistance to any of these types of attack can result in the ability to decrypt ciphertext (mainly (a)) or to obtain directly the value of the private key (mainly (c)). We have purposely omitted from the list resistance to purely algebraic attacks, such as problems with a low value of the private or public exponent, among other similar ones (their overview appears in [6]), since most successful attacks are based on an incorrect use of RSA and implementation faults. The problem of the correct use of RSA is rooted in the mathematics underlying the algorithm (for details see [11, 2, 6, 7, 13, 5] and attacks presented there) and thus it should be examined from a mathematical perspective. It seems too risky to leave the issue in the hands of implementators. We also note that cryptanalysis has gradually accepted the assumption that an attacker has nearly unlimited access to an attacked system. We do not merely consider attacks on "data passing through" but direct attacks on autonomous cryptographic units. This approach is logically enforced by the realistic situation when users have access to certain features of the cryptographic modules, without knowledge of their inner set-up (mainly cryptographic keys).

Furthermore, we can see that it is not possible to satisfactorily solve the defence against the types of attacks specified above by a single universal encoding of data being encrypted. This is a consequence of the fact that the encoding mechanism is only part of the whole scheme and as such can only affect part of its properties.

Now we will look at basic defence mechanisms against the above types of attacks. The first category, adaptive chosen ciphertext attacks, has not been considered in this paper. We think that a satisfactory solution is the random oracle paradigm [4], which has been successfully applied [21, 22, 9]. For category (b), we need to constantly bear in mind the claim in [11], and prevent any leakage of plaintext information. It is not possible to limit our attention only to the easily visible information such as the value of the most significant octet of plaintext in RSAES-OAEP. In section 2, we showed that the leakage of information from completely other part of the scheme has also a negative effect on security. Power side channel attacks [12, 14, 1] and nascent theory of electromagnetic side channel attacks [18, 10] is necessary to be considered a particularly high threat. However, defence measures against these channel attacks [8] are beyond the scope of this paper. It was our aim to show that these countermeasures need to be used in every single function that deals with individual parts of the plaintext. Here we focused our attention on the function SHA-1 as an example.

The last category is fault attacks. The vulnerability of RSA to these attacks does not originate directly from the theorem [11]. However, it seems to be an innate quality of the RSA system [3, 6, 7]. As well as with the other types of attacks, certain types of encoding can more or less eliminate fault attacks. We showed that RSA-KEM, despite it seems to be well resistant to other types of attacks [21], can be easily and straightforwardly effected by fault side channel attacks. To avoid fault attacks it is recommended especially:

a) To consistently check the integrity of the private key and of the other parameters used with it in its processing
b) To minimize the range of error messages
c) Wherever possible, to use platforms equipped with fault detection and eventually also correction facilities (fault tolerant systems)

As a rather strong countermeasure, even though not 100% sure, we can recommend to check every result $x = (y^d \bmod n)$ as $y =^? (x^e \bmod n)$, where d is the private exponent, e is the public exponent and n is the respective RSA modulus. This measure effectively prevents both attacks presented as the examples in this paper. The proof is simple: with a high probability, the relationship $e*d \equiv 1 \ (\mathrm{mod}\ ord(y))$, where $ord(y)$ is the order of y in the multiplicative group $\boldsymbol{Z_n}^*$, will be violated in both examples.

5 Conclusion

The RSA individual bits theorem [11] is generally considered to be a good property of RSA [19]. However, it also shows the way for attacks based on side channels. As it was shown in [5, 13], the RSA scheme is prone to these attacks not only theoretically, but also practically.

We have presented another possible attack on the encryption scheme RSAES-OAEP where, in contrast with the previous work [13], we attack that part of the plaintext “shielded” by the OAEP method. In this, we use the algebraic properties of RSA, rather than some weakness of the OAEP encoding. To prevent this attack, we need to eliminate the parasitic leakage of information from individual operations in partial procedures of the entire scheme. This goes well beyond the scope of the general description of the OAEP encoding method. Also Manger´s and Bleichenbacher´s attacks mainly employ the basic properties of the RSA algorithm. From the type of the encoding (EME-OAEP or PKCS#1 v.1.5) they only choose those RSA features that allow an attack.

Next we presented a new side channel attack on the RSA-KEM. This scheme was built to prevent the parasitic leakage of information about the plaintext, especially under the consideration of chosen ciphertext attack. However, we managed to point out a side channel that allows the leakage of this information. Unlike previous attacks that returned the plaintext, this time the attacker obtains the RSA private key. The attack was again made possible by the basic multiplicative property of RSA.

Our contribution underlies the significance of the known algebraic properties of RSA in relation to rapidly evolving attacks based on side channels. Consequently, it is possible to expect similar side channel attacks in other RSA schemes that may employ different message encoding. Therefore, it is necessary to pay more attention to side channel countermeasures in implementations of these cryptographic schemes.

As a small note in our paper, we pointed out the rule of keeping RSA keys for encryption and digital signature strictly separated, which is often neglected. We assumed that the rule is not adhered to (as in often the case in SSL), and described an approach to convert both Manger´s and Bleicherbacher´s oracles for plaintext decryption into oracles that can create valid digital signatures for arbitrarily encoded messages.

6 References

1. Akkar, M.-L., Bevan, R., Dischamp, P. and Moyart, D.: *Power Analysis, What Is Now Possible...*, in Proc. of ASIACRYPT 2000, pp. 489-502, 2000.
2. Alexi, W., Chor, B., Goldreich, O. and Schnorr, C.: *RSA and Rabin functions: Certain parts are as hard as the whole*, SIAM Journal on Computing, 17(2), pp. 194-209, 1988.
3. Bao, F., Deng, R.-H., Han, Y., Jeng, A., Narasimhalu, A.-D. and Ngair, T.: *Breaking Public Key Cryptosystems on Tamper Resistant Devices in the Presence of Transient Faults*, in Proc. of Security Protocols '97, pp. 115-124, 1997.
4. Bellare, M. and Rogaway, P.: *Random Oracles are Practical: A Paradigm for Designing Efficient Protocols*, October 20, 1995, originally published in Proc. of the First ACM Conference on Computer and Communications Security, ACM, November 1993.
5. Bleichenbacher, D.: *Chosen Ciphertexts Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1*, in Proc. of CRYPTO '98, pp. 1-12, 1998.
6. Boneh, D.: *Twenty Years of Attacks on the RSA Cryptosystems*, Notices of the American Mathematical Society, vol. 46, no. 2, pp. 203-213, 1999.
7. Boneh, D., DeMillo, R. A. and Lipton, R. J.: *On the Importance of Checking Cryptographic Protocols for Faults*, in Proc. of EUROCRYPT '97, pp. 37-51, 1997.
8. Chari, S., Jutla, C.-S., Rao, J. and Rohatgi, P.: *Towards Sound Approaches to Counteract Power-Analysis Attacks*, in Proc. of CRYPTO '99, pp. 398-411, 1999.
9. Fujisaki, E., Okamoto, T., Pointcheval, D. and Stern, J.: *RSA-OAEP Is Secure under the RSA Assumption*, in Proc. of CRYPTO 2001, pp. 260-274, 2001.
10. Gandolfi, K., Mourtel, C. and Olivier, F.: *Electromagnetic Analysis: Concrete Results*, in Proc. of CHES 2001, pp. 251-261, 2001.
11. Håstad, J. and Näslund M.: *The Security of Individual RSA Bits*, in Proc. of FOCS '98, pp. 510-521, 1998.
12. Kocher, P., Jaffe, J. and Jun, B.: *Differential Power Analysis: Leaking Secrets*, in Proc. of CRYPTO '99, pp. 388-397, 1999.
13. Manger, J.: *A Chosen Ciphertext Attack On RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized In PKCS #1*, in Proc. of CRYPTO 2001, pp. 230-238, 2001.
14. Messegers, T.-S., Dabbish, E. A. and Sloan, R. H.: *Investigations of Power Analysis Attacks on Smartcards*, in Proc. of USENIX Workshop on Smartcard Technology, pp. 151-161, 1999.
15. PKCS#1 v2.1: *RSA Cryptography Standard*, RSA Laboratories, DRAFT2, January 5 2001.
16. PKCS#5 v2.0: *Password-Based Cryptography Standard*, RSA Laboratories, March 25, 1999.
17. Pohlig S.C., Hellman M.E.: *An improved algorithm for computing logarithms over GF(p) and its cryptographic significance*, IEEE Transactions on Information Theory, 24 (1978), 106-110.
18. Rao, J.-R and Rohatgi, P.: *EMpowering Side-Channel Attacks*, preliminary technical report, May 11 2001.
19. Rivest, R., L., Shamir, A. and Adleman L.: *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, pp. 120-126, 1978.
20. Secure Hash Standard, Federal Information Processing Standards Publication 180-1, 1995 April 17.
21. Shoup, V.: *A Proposal for an ISO Standard for Public Key Encryption (version 2.0)*, September 17, 2001.
22. Shoup, V.: *OAEP Reconsidered (Extended Abstract)*, in Proc. of CRYPTO 2001, pp. 239-259, 2001.
23. Stinson, D., R.: *Cryptography – Theory and Practice*, CRC Press, 1995.