Observability Analysis
– Detecting When Improved Cryptosystems Fail –

Marc Joye[1], Jean-Jacques Quisquater[2], Sung-Ming Yen[3,⋆], and Moti Yung[4]

[1] Gemplus Card International, Card Security Group, Gémenos, France
marc.joye@gemplus.com – http://www.geocities.com/MarcJoye/
[2] UCL Crypto Group, Louvain-la-Neuve, Belgium
jjq@dice.ucl.ac.be – http://www.uclcrypto.org/
[3] Dept of Computer Science, National Central University, Taiwan, R.O.C.
yensm@csie.ncu.edu.tw – http://www.csie.ncu.edu.tw/~yensm/
[4] CertCo, New York NY, U.S.A.
moti@{certo.com,cs.columbia.edu} – http://wwww.certco.com/

Abstract. In this paper we show that, paradoxically, what seems like a "universal improvement" or a "straight-forward improvement" which enables better security and better reliability on a theoretical level, may in fact, within certain operational contexts, introduce new exposures and attacks, resulting in a weaker operational cryptosystem. We demonstrate a number of such dangerous "improvements". This implies that careful considerations should be given to the fact that an implemented cryptosystem exists within certain operational environments (which may enable certain types of tampering and other observed information channels via faults, side-channel attacks or behavior of system operators). We use our case studies to draw conclusions about certain investigations required in studying implementations and suggested improvements of cryptosystems; looking at them in the context of their operating environments (combined with their potential adversarial settings). We call these investigations *observability analysis*.

Keywords. Security analysis, observability, cryptanalysis, implementations, side-channel attacks, fault analysis, robustness, cryptosystems.

1 Introduction

The aim of this paper is to highlight that, contrary to the common belief, some popular measures which were suggested to enhance the reliability and security of a basic cryptosystem introduce new attacks, often more insidious (and thus more difficult to identify). These enhancements, *paradoxically*, result in a possibly weaker operational system. The attacks are applicable within a working environment and an attack model. In particular, we consider adversaries which

⋆ Supported in part by the Computer & Communication Research Laboratories, Industrial Technology Research Institute, Republic of China.

may inject faults (as was first suggested in [8]), but ones with "limited-feedback channel". Namely, adversaries which get back only a limited feedback from the system (e.g., an indication about the case of fault, which may be observed due to change of behavior of components or parties within the system). The adversaries do not get actual outputs of the decryption device. We also employ such adversaries which introduce faults into ciphertexts (as in [5]), and ones which perform power analysis [21].

Our test case is the RSA system. In fact, RSA is undoubtedly the most widely used and accepted public-key cryptosystem. Owing to this popularity, it is also perhaps the most cryptanalyzed system [7]. Furthermore, many optimizations and improvement measures are known for it. Therefore, exposures resulting from improper use of RSA seem nowadays to be minimal, which contrast our results. We note that we have chosen the RSA system for concreteness, the same conclusion may remain valid for various other cryptosystems, as well. We believe that our examples are quite basic yet demonstrative and educational.

Notations. Throughout, the public RSA modulus will be denoted by $n = pq$ for two secret primes p and q; the public encryption key (resp. secret decryption key) will denoted by e (resp. d).

Organization of the paper. In the next section, we review a simple precaution suggested to avoid the leakage of secrets due to faults in the context of RSA. We then show that this method (or any other method with similar logic behind it, in particular the one suggested for CRT-based RSA) which protects against such leakages, paradoxically, may also be used to recover some secret information. Section 3 illustrates that the optimal asymmetric encryption padding (OAEP) proposed by Bellare and Rogaway, which is certainly one of the best method currently available to encrypt with RSA (chosen-ciphertext secure in the random oracle model), paradoxically, is susceptible to some attacks (in some setting) which do not apply to the plain RSA encryption (i.e., the RSA primitive). In Section 4, we show that the same conclusion holds for the so-called 'RSA for paranoids', a stronger variant of RSA. Namely, using the stronger RSA version (in conjunction with some provably secure padding) introduces new exposures. Finally, we would like to resolve the "seemingly paradox phenomenon" and indeed we conclude in Section 5 where we explain the underlying issues behind the "paradoxes" where certain exposures which may be observable are in fact a result of an "improved" operational decryption mechanism. (This conclusion is, in fact, independent of and more important than whether the reader will consider our case studies to be really paradoxical or merely "seemingly paradoxical" examples!). We suggest a way to view and analyze possible implementations in light of observable events which may lead to activating possible countermeasures.

2 The Case of Added Reliability: RSA Enhanced with Fault Analysis Prevention

2.1 How to implement the RSA?

RSA in standard mode. The decryption process in the (plain) RSA goes as follows. Given a ciphertext $c = m^e \bmod n$, one recovers the plaintext as $m = c^d \bmod n$.

Suppose that an error occurs during the computation of $m = c^d \bmod n$; more precisely, suppose that one bit of d, say bit d_j, has flipped (let d' denote the corrupted value of d). It is then very easy to recover the flipped bit and its value [3, 8, 17]. The decryption process will yield $m' = c^{d'} \bmod n$ instead of m. Let $d = \sum_{i=0}^{k-1} d_i\, 2^i$ denote the binary expansion of d. Since

$$d' = \sum_{\substack{i=0 \\ (i \neq j)}}^{k-1} d_i\, 2^i + \overline{d_j}\, 2^j = d + (\overline{d_j} - d_j) 2^j$$

it follows that

$$\frac{(m')^e}{c} \equiv c^{e(\overline{d_j} - d_j) 2^j} \equiv \begin{cases} c^{e\, 2^j} & (\bmod\ n) \quad \text{if } d_j = 0\,, \\ 1/c^{e\, 2^j} & (\bmod\ n) \quad \text{if } d_j = 1\,. \end{cases} \tag{1}$$

Therefore *if an adversary can get access to the value of* m', she can recover the bit d_j by exhaustively testing whether $(m')^e / c \equiv c^{\pm e\, 2^j} \pmod{n}$ for some $0 \leq j \leq k-1$. This attack readily extends to the case where several bits of secret exponent d have flipped. As noted by Kaliski [19], such an attack is easily defeated by checking whether the decrypted message, m', satisfies $(m')^e \equiv c \pmod{n}$ and outputting the plaintext message $m = m'$ if and only if the comparison is successful.

RSA in CRT mode. The RSA decryption can be speeded up by a factor of 4 using *Chinese remaindering* (CRT) [25]. From $c = m^e \bmod n$, the corresponding plaintext m is recovered as

$$m = m_p + p[p^{-1}(m_q - m_p) \bmod q] \tag{2}$$

where $m_p = c^{d_p} \bmod p$, $m_q = c^{d_q} \bmod q$, $d_p = d \bmod (p-1)$, and $d_q = d \bmod (q-1)$.

An error within the CRT mode of operation has much more devastating consequences than within the standard mode. Suppose that the computation modulo p is corrupted (we let m'_p denote the corrupted value) while the computation modulo q is not. Then from Eq. (2), the CRT recombination will yield $m' = m'_p + p[p^{-1}(m_q - m'_p) \bmod q]$ instead of m. Since $m \not\equiv m' \pmod{p}$ and $m \equiv m' \pmod{q}$, it follows that the $\gcd({m'}^e - c \pmod{n}, n)$ gives the secret factor q [15], whereas the original fault analysis idea is presented in [8]. Note that this attack is stronger than the previous one: there is no particular assumption on the kind of (induced) errors; the attack is successful as soon as there is an error (*any* error) during the exponentiation modulo one prime factor. Note also that this attack can be used by an adversary to break the system *only if she can get access to the value of* m'.

An elegant method to protect against such kind of failure models was suggested by Shamir at the rump session of *EUROCRYPT '97* [29] (see also [16]). From a randomly chosen small integer r, the decryption algorithm first computes $m_{rp} = c^{d_{rp}} \bmod rp$ and $m_{rq} = c^{d_{rq}} \bmod rq$ (where $d_{rp} = d \bmod \phi(rp)$ and

$d_{rq} = d \bmod \phi(rq)$). Then if $m_{rp} \not\equiv m_{rq} \pmod{r}$, an error has occurred and the system outputs an `error` message[1]; otherwise the computations are supposed correct and the plaintext m is recovered by applying Chinese remaindering on $m_p = m_{rp} \bmod p$ and $m_q = m_{rq} \bmod q$ (see Eq. (2)). The probability that an error is undetected is equal to $1/r$. For example, if r is a 20-bit integer, this probability is already smaller than 10^{-6}. The advantage of Shamir's countermeasure resides in its universality: it is applicable even when the value of e is not available.

2.2 First paradox

From the above discussion, it appears that the right way to implement the RSA primitive is to use Chinese remaindering (for efficiency) along with Shamir countermeasure (for security in the induced-fault model). However, as we will see, the conclusion is not so straight-forward. The introduction of Shamir's countermeasure, or more generally of any other method for detecting errors, makes the system *irregular*: it now behaves differently depending on whether the computations are error-free or not, namely it outputs the correct decryption or an `error` message, respectively. The system may thus be used as an *oracle* to try to collect some secret information. The next paragraph sketches a possible method to devise such an oracle for many existing implementations of the RSA primitive. See [31] for details and further discussions.

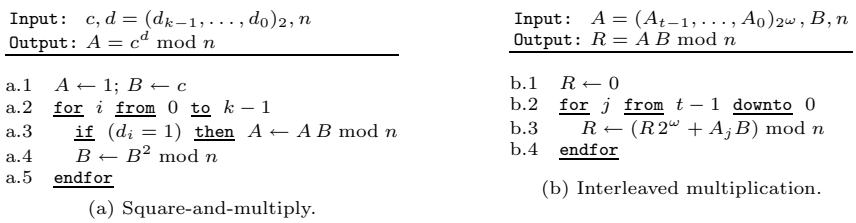
The adversarial setting is the induced-fault model with *limited-feedback* channel. We assume that an adversary is merely an observer who *only knows whether a ciphertext decrypts correctly or not*, and that she has no access to the corresponding plaintext m (or m'). This assumption is much weaker than an attacker who chooses plaintexts and ciphertexts and such weaker attack is more likely to occur, making the system more vulnerable. We will next outline an attack which in this situation will demonstrate that:

> *"A more reliable system (i.e., designed to detect faults) may, in fact, be weaker".*

Let us, next, review the attack. RSA exponentiation is usually implemented with the square-and-multiply technique (Fig. 1a) where multiplication and modular reduction are interleaved to fit the word-size $\Omega = 2^\omega$ (Fig. 1b) (e.g., see [20]). Imagine that an adversary wants to guess the value of the i^{th} bit d_i of the decryption exponent d.[2] Suppose that $d_i = 1$, the interleaved multiplication $A\,B \bmod n$ (Line a.3, Fig. 1) is thus performed at iteration i. Suppose furthermore that one

[1] Remark that if the decryption algorithm attempts to recompute the answer, this will take a longer time to complete the computation and will be revealed by a timing analysis.

[2] Here d has to be understood as the secret exponent involved in the exponentiation. It stands for the decryption key d in standard mode and for $d_p = d \bmod (p-1)$ (or $d_q = d \bmod (q-1)$) in CRT mode.



(a) Square-and-multiply.

(b) Interleaved multiplication.

Fig. 1. RSA exponentiation.

or several bits of error are introduced into the more significant positions of register A, or more precisely into some words A_j for $j > j_\tau$ where j_τ represents the current value of counter j (Line b.2, Fig. 1) when the faulty bits are introduced. Since the words containing the errors are no longer required for the next iterations (i.e., for $j = j_\tau, j_\tau - 1, \ldots, 0$, Line b.2, Fig. 1), the computation of $R = A\,B \bmod n$ will be correct. Moreover, since R is restored into register A, $A \leftarrow A\,B \bmod n$ (Line a.3, Fig. 1), the error located in register A will be cleared and the final result $c^d \bmod n$ will be correct, too. Conversely, if the value of bit d_i was 0, then the interleaved multiplication (Line a.3, Fig. 1) is bypassed at iteration i and the errors induced into register A will not be cleared, resulting in an incorrect value for the final result $c^d \bmod n$. Remember that we made the explicit assumption that the adversary knows whether a ciphertext decrypts correctly or not. So, by inducing faulty bits as previously described, she knows the value of bit d_i according to whether the decryption algorithm returns an `error` message or not.

Let us emphasize again that if Shamir's countermeasure (or any other means to detect errors) was not implemented, then the adversary would not be able to guess the correct value of d_i because, due to the limited feedback she is allowed to have, she doesn't know whether a ciphertext decrypts correctly or not. In this case, the only way for her to recover the value of d_i is to raise to the e the value of $c^d \bmod n$ returned by the decryption algorithm and to compare it with the original value of c; if they match then $d_i = 1$, otherwise $d_i = 0$. However, this supposes a much stronger assumption. Namely that (besides inducing faults) the adversary has access to the (raw) decrypted values of many ciphertexts, which, in most cases, is quite unrealistic.

3 The Case of Added Robustness to Stronger Attacks: Optimal Asymmetric Encryption Padding

In this section we will again consider the induced-fault attack model and the same passive attacker who is an observer of the system's reaction.

3.1 How to encrypt with RSA?

Recall that there is a big difference between the *RSA primitive* also called plain RSA encryption (that is, the modular exponentiation function $f : x \mapsto f(x) =$

$x^e \bmod n$) and an *RSA encryption scheme* (that is, a particular way to use the RSA primitive to encrypt a message). Plain RSA encryption is definitively not a reasonable way to encrypt with RSA: as observed by Goldwasser and Micali [14], an encryption scheme had better to be probabilistic. This stems from the fact that a deterministic scheme, in essence, does not offer the desired property of *indistinguishability*. Informally, indistinguishability is defined as the adversary's inability to make the difference between the encryptions of bits '0' and '1', or more generally, given a challenge ciphertext, to learn any information about the corresponding plaintext. This does not imply that the converse is necessarily true: Bleichenbacher [5] has shown that the *(probabilistic)* encryption standard RSA PKCS #1 v1.5 does not achieve indistinguishability and exploited this failure to mount a chosen ciphertext attack on some interactive key establishment protocols (e.g., SSL) constructed from it. Other problems of plain schemes are presented in [9]. The commonly recommended way to encrypt with RSA is the Optimal Asymmetric Encryption Padding (OAEP) by Bellare and Rogaway [4] (which was claimed to achieve chosen-ciphertext security [26]).[3] This method was supported by the RSA standardization process [6] following the Bleichenbacher attack. It was then published as RSA PKCS #1 v2.0, which will be followed by the IEEE and ANSI X9 standards.

A simplified version of OAEP (called basic embedding scheme in [4]) goes as follows. Let $k = \lfloor \log_2(n) \rfloor$. A message $m < 2^{k-k_0}$ is encrypted as

$$c = \Big(m \oplus G(r) \;\Big\|\; r \oplus H\big(m \oplus G(r)\big) \Big)^e \bmod n \tag{3}$$

for a (public) "generator" function $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$, a (public) hash function $H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$ and where r is a random integer uniformly chosen in $\{0,1\}^{k_0}$. To decrypt the ciphertext c, the decryption algorithm computes $x := c^d \bmod n$. Then setting x_0 to the k_0 least significant bits of x and x_1 to the remaining bits of x (i.e., $x = x_1 \| x_0$), it computes $r' = x_0 \oplus H(x_1)$ and returns $x_1 \oplus G(r')$ as the plaintext message corresponding to c.

OAEP differs from the above in that some extra bits are padded: they are used to check the integrity of the message. In particular, OAEP achieves *plaintext-awareness*, that is, informally, the impossibility of producing a ciphertext without the knowledge of the corresponding plaintext, a random oracle property which implies chosen-ciphertext security. Let again $k = \lfloor \log_2(n) \rfloor$. To encrypt a message $m < 2^{k-k_0-k_1}$, choose a random integer r in $\{0,1\}^{k_0}$ and compute

$$c = \Big(m\{0\}^{k_1} \oplus G(r) \;\Big\|\; r \oplus H\big(m\{0\}^{k_1} \oplus G(r)\big) \Big)^e \bmod n \tag{4}$$

for $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$ and $H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$. Then, given c, the decryption algorithm computes $x = c^d \bmod n$ and sets x_0 to the k_0 least

[3] Recently, Shoup [30] has shown that the original proof was enough to claim only security against a non-adaptive adversary [23], and a new proof of security was constructed [12].

significant bits of x and x_1 to the the remaining bits of x (i.e., $x = x_1 \| x_0$). Next, it computes $r' = x_0 \oplus H(x_1)$ and $y = x_1 \oplus G(r')$, and sets y_0 to the k_1 least significant bits of y and y_1 to the remaining bits of y (i.e., $y = y_1 \| y_0$). If $y_0 = \{0\}^{k_1}$ then it returns y_1 as plaintext; otherwise it returns an `error` message.

3.2 Second paradox

Here too, we see that the decryption algorithm acts as an *oracle*. Hence as in Section 2, we suppose that the RSA exponentiation is carried out with an algorithm such as the one depicted in Fig. 1, then by introducing some errors, an adversary is able to recover the value of the bits d_i of the secret decryption key d according to the decryption is possible or not. It is worth remarking that the "basic embedding scheme" (see Eq. (3)) or even the plain RSA encryption are not susceptible to this attack because they do not check the integrity.[4] In those two cases, the adversary must have access to the decrypted value, encrypt it and finally compare it to the initial ciphertext to guess the value of d_i; with OAEP, the value of d_i is deduced from *the only knowledge that an* `error` *message is returned or not*, the knowledge of the decrypted value is not necessary. We thus have a second paradox:

> *"A more robust implementation (i.e., secure against active attacks) may, in fact, be weaker."*

3.3 Setting-dependent robustness in another cases

The attacks described in this section (and in Section 2 and the subsequent paradoxes) assume some variant of the induced-fault model of [8]. However, the idea that improving a system to increase its robustness in one sense, may not suffice for other considerations can be widely applicable (regardless of the specific model). In fact, we can draw exactly this conclusion by considering a model where the decryption is carried out by a device which is really tamper-proof (and no faults are possible). Our attack follows the recent attack by Manger [22] against (some implementations of) RSA PKCS #1 v2.0 combined with the power analysis of [21]. RSA PKCS #1 v2.0 [1] is a slight variation of OAEP where the most significant byte (MSB) of the encoded message, $\widetilde{m}$, being encrypted is forced to `00h` to ensure that the resulting padding is always smaller than modulus n.

Current implementations of the decryption operation decrypt c to get $\widetilde{m} = c^d \bmod n$, *check whether the first byte of* $\widetilde{m}$ *is zero* and if so, check the OAEP integrity of $\widetilde{m}$. If both verifications are successful, the plaintext message m corresponding to $\widetilde{m}$ is output; otherwise, there is an `error` message. If we can distinguish between an error resulting from a too large message (i.e., $\mathrm{MSB}(\widetilde{m}) \neq$ `00h`) and an error resulting from an incorrect decoding, then a chosen ciphertext attack similar in spirit to that of Bleichenbacher [5] can be mounted (see [22] for details).

[4] Note, however, that we do not suggest to use a weaker form of encryption for many other reasons.

The trivial solution consists of course in making the two error messages identical, as already recommended in the PKCS#1 v2.0 standard (*cf.* [1, § 7.1.2]). The last PKCS#1 v2.1 draft [2] explicitly requests to made the two kinds of error indistinguishable; in particular, it insists that the execution time of the decryption operation (timing channel) must not reveal whether the first byte is `00h` or not. So, it is suggested that, in the case of $\mathrm{MSB}(\widetilde{m}) \neq \texttt{00h}$, to proceed to the OAEP integrity check by setting $\widetilde{m}$ to a string of zero octants. However, such an "improved" solution is not satisfactory in all respects, since it is very likely that power analysis (power consumption channel) will reveal information as the power consumption is related to the manipulated data, making easy the distinction between the zero string (when $\mathrm{MSB}(\widetilde{m}) \neq \texttt{00h}$) and a random-looking string $\widetilde{m}$ (when $\mathrm{MSB}(\widetilde{m}) = \texttt{00h}$).

4 The Case of Increased Security Parameter: Unbalanced RSA

We go back to investigate a seemingly paradoxical situation. Actually, this section shows that a seemingly improved variant of RSA is subject to somewhat stronger attacks in the physical sense, since they can be mounted remotely. Namely, we increase the attacker's power to choose ciphertexts, which she can "transmit only remotely" (with or without errors) to the device; at the same time we limit the adversary's device tampering power as before to be an observer of the error messages (and not allowing it access to decrypted values).

4.1 RSA for paranoids

The security of the RSA system is based on the difficulty of factoring large integers. Therefore, a larger modulus offers further security, at the expense, however, of a larger computational effort. A good compromise is to use an *unbalanced* RSA modulus [28], that is, a modulus $n = pq$ where p, q are primes and $q \gg p$ (e.g., $|p| = 500$ bits and $|q| = 4500$ bits). The best factorization algorithms [24] cannot take advantage of these special moduli and they seem thus as secure as moduli constructed from the product of two 2500-bit primes. Shamir [28] observed that if the plaintext m being encrypted is smaller than p, then, from the corresponding ciphertext $c = m^e \bmod n$, it can be recovered as $m = c^{d_p} \bmod p$ (where $d_p = d \bmod (p-1)$). This follows immediately from Eq. (2) by noting that $m = m \bmod p = m_p$. The resulting system is called *RSA for paranoids*.

4.2 Third paradox

Is the name 'RSA for paranoids' really justified? If a plaintext m larger than p is encrypted (let $c' = m^e \bmod n$ denote the corresponding ciphertext), then the decryption gives $m' = {c'}^{d_p} \bmod p = m \bmod p \neq m$. Consequently, $\gcd(m - m', n)$ gives the secret prime p and the system is broken [13]. Note that this can be turned into an active attack *only if the adversary can get access to the value of*

m', which, as in the previous sections, may be considered in many cases as an unrealistic assumption.

The usual way to prevent such an attack is to enhance the purely algebraic "plain" scheme and add redundancy and randomness to the plaintext prior to encryption. The redundancy enables one to check the integrity of the plaintext and the randomness serves to avoid many drawbacks inherent to any deterministic encryption algorithm (*cf.* Section 3). We thus assume that the system is implemented using an appropriate embedding of the kind of OAEP[5] (or any other applicable variant thereof); the main point being that the decryption system "internally" checks the *integrity* of the plaintexts (and assures message awareness even against active attacks).

Thus, as before, we now make the weaker assumption that *the adversary only knows whether a ciphertext can be decrypted or not*, but in no way, she can get access to a decrypted value. However, we allow her to choose ciphertexts. The attack demonstrates our principle of observing behavior under an oracle. Technically, it follows the basic properties shown in [13] (see also [18]).

Since the plaintexts being encrypted must be smaller than p, we set $k = \lfloor \log_2(p) \rfloor$ in the OAEP description given by Eq. (4):

$$\mathrm{OAEP}(m) = \underbrace{m\{0\}^{k_1} \oplus G(r)}_{k-k_0 \text{ bits}} \Big\| \underbrace{r \oplus H\big(m\{0\}^{k_1} \oplus G(r)\big)}_{k_0 \text{ bits}}$$

for a message $m \in \{0,1\}^{k-k_0-k_1}$ and a random $r \in \{0,1\}^{k_0}$, where $G : \{0,1\}^{k_0} \to \{0,1\}^{k-k_0}$ and $H : \{0,1\}^{k-k_0} \to \{0,1\}^{k_0}$. The adversary can fix the value of the $(k+1-k_0-k_1)$ most significant bits of $\mathrm{OAEP}(m)$ by an appropriate choice for message m: if she wants that these bits represents a chosen value T, she simply sets $m = T \oplus [G(r)]^{k-k_0-k_1}$ where $[G(r)]^{k-k_0-k_1}$ denotes the $(k-k_0-k_1)$ most significant bits of $G(r)$. Futhermore, the adversary is not restricted to probe with valid messages; she can choose an m out of the prescribed range (i.e., $m \geq 2^{k-k_0-k_1}$) so that $\mathrm{OAEP}(m)$ will be a $(k+1)$-bit number or more. Consequently, the adversary has a total control on all the bits of $\mathrm{OAEP}(m)$ except the (k_0+k_1) least significant ones. From this observation, we will now explain how she can recover the $(k+1-k_0-k_1)$ most significant bits of the secret prime p. The remaining bits of p may be found by appropriate choices for r, exhaustion or more elaborated techniques (e.g., [10]).

As the value of k is public, the adversary knows that p lies in the interval $I_0 = \big(2^k, 2^{k+1}\big)$. Then she chooses an m so that its OAEP embedding $x_0 := \mathrm{OAEP}(m)$ belongs to I_0 and computes the corresponding ciphertext $c_0 = {x_0}^e \bmod n$. If c_0 can be decrypted[6] then she knows that $x_0 < p$; otherwise (i.e., if an `error` message is returned) she knows that $x_0 \geq p$. She then reiterates the process with

[5] Remark that OAEP, as is, does not apply to 'RSA for paranoids' since its usage is limited to permutations. One has to use, for example, the more general construction of [11].

[6] The attack supposes that the decryption algorithm does not check, implicitly or explicitly, that the decrypted m is $< 2^{k-k_0-k_1}$ (see [18, Section 3]).

the interval $I_1 = \left(x_0, 2^{k+1}\right)$ or $I_1 = \left(2^k, x_0\right]$, respectively. And so on... until the $(k+1-k_0-k_1)$ most significant bits of p are disclosed. Note that this attack does not make sense in the standard RSA; that is, when the decryption is computed modulo $n = pq$. So, in this view, the RSA for paranoids with a 5000-bit RSA modulus ($|p| = 500$ bits and $|q| = 4500$ bits) is weaker than the standard RSA using a 1000-bit modulus where $|p| = |q| = 500$ bits. Hence, the paradox:

> *"A stronger system with increased length parameters (which possibly make the underlying problem harder) may, in fact, be weaker."*

5 Conclusions

What we saw are three improvements that actually in some operational setting induce exposures (vulnerabilities). In each of the three cases, we show that there are implementation environments where:

1. Increasing Reliability may weaken the system;
2. Improving Security and Robustness to attacks (OAEP) may weaken the system;
3. Improving certain security parameters (especially if the growth is not uniform in all parameters) may weaken the system.

The thesis put forth and demonstrated in this paper is that a system exists within certain operational environment which may enable adversaries to tamper and to perform other (somewhat limited) observations regarding the system's operations. Paradoxically, what seems like a "universal improvement" or "straight-forward improvement" which enables better security and better reliability, may in fact, within these specific operational contexts, introduce new exposures and attacks.

Where do the paradoxes come from? Given the attack, one notices that the improved "more secure and reliable system" takes additional computational measures to improve itself. However, these measures may increase the **observability** of the system w.r.t. certain (possibly newly) introduced "events". Under certain allowed attack scenarios (interaction with the implemented environment) this increased observability in fact, makes the systems less smooth in the sense that the events (reported by the oracle) reveal the inner workings. The adversary in the less smooth operation mode, can play with or observe the increased set of possible events. This, in turn, may enable her to break the system. On the other hand, the system without the improvement, may operate more smoothly and produce less noticeable differences in reported events, thus not enabling attacks as above. It is expected that as cryptography becomes very strong, future adversaries will concentrate on exploiting observable events and information emitted by the working environment. The basic rule which may resolve the paradoxes can (informally) be stated as follows:

> *"A stronger system whose operation, however, is less smooth than a weaker counterpart (namely where its interaction with the adversarial*

environment enables enhanced "observability" of events which reflect on its inner workings), may, in fact, be weaker."

In conclusion, the natural belief that obvious straight-forward security improvements, are always good (namely, are universal) is a fallacy. In fact, security measures are *context sensitive*, especially when we consider the full cycle of a cryptographic system (design, implementation, and operation). The interplay of the *basic design* with the *implementation* and *operational environment* and the potential adversaries in each stage, should be considered very carefully in the deployment of actual working systems. While the above issue may have been implicitly noticed earlier, our examples demonstrate it on quite modern state-of-the-art constructions. We feel that the notion of "event observability" and its implied exposures and weaknesses, have to be taken into consideration when analyzing the security of a working cryptosystem. Our investigation points out that one should conduct what may be called "observability analysis" of the working environment, analyzing events under an expected or a reasonable (well modeled) set of exposures. It should then be made sure that the potentially observable events are not made available to the possible adversary (by either technical, physical or operational measures).

References

1. RSA Laboratories. PKCS #1 v2.0: RSA cryptography standard, October 1, 1998. Available at `http://www.rsasecurity.com/rsalabs/pkcs/`.
2. RSA Laboratories. PKCS #1 v2.1: RSA cryptography standard, Draft 2, January 5, 2001. Available at `http://www.rsasecurity.com/rsalabs/pkcs/`.
3. F. Bao, R. Deng, Y. Han, A. Jeng, A. D. Narasimhalu, and T.-H. Ngair. Breaking public key cryptosystems on tamper resistant devices in the presence of transient faults. In B. Christianson, B. Crispo, M. Lomas, and M. Roe, eds, *Security Protocols*, vol. 1361 of *Lecture Notes in Computer Science*, pp. 115–124, Springer-Verlag, 1998.
4. Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption — How to encrypt with RSA. In A. De Santis, ed., *Advances in Cryptology – EUROCRYPT '94*, vol. 950 of *Lecture Notes in Computer Science*, pp. 92–111, Springer-Verlag, 1995.
5. Daniel Bleichenbacher. A chosen ciphertext attack against protocols based on the RSA encryption standard RSA PKCS #1. In H. Krawczyk, ed., *Advances in Cryptology – CRYPTO '98*, vol. 1462 of *Lecture Notes in Computer Science*, pp. 1–12, Springer-Verlag, 1998.
6. Daniel Bleichenbacher, Burt Kaliski, and Jessica Staddon. Recent results on PKCS #1: RSA encryption standard. *RSA Laboratories' Bulletin*, no. 7, June 1998.
7. Dan Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.
8. Dan Boneh, Richard A. DeMillo and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In W. Fumy, ed., *Advances in Cryptology – EUROCRYPT '97*, vol. 1233 of *Lecture Notes in Computer Science*, pp. 37–51, Springer-Verlag, 1997.
9. Dan Boneh, Antoine Joux, and Phong Q. Nguyen. Why Textbook El Gamal and RSA encryption are insecure. In T. Okamoto, ed., *Advances in Cryptology –*

ASIACRYPT 2000, vol. 1976 of *Lecture Notes in Computer Science*, pp. 30–43, Springer-Verlag, 2000.
10. Don Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. *Journal of Cryptology*, 10(4):233–260, 1997.
11. Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In H. Imai and Y. Zheng, eds., *Public Key Cryptography*, vol. 1560 of *Lecture Notes in Computer Science*, pp. 53–68, Springer-Verlag, 1999.
12. Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. RSA–OAEP is secure under the RSA assumption. In J. Kilian, ed., *Advances in Cryptology – CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001.
13. Henri Gilbert, Dipankar Gupta, Andrew Odlyzko, and Jean-Jacques Quisquater. Attacks on Shamir's 'RSA for paranoids'. *Information Processing Letters*, 68:197–199, 1998.
14. Shafi Goldwasser and Silvio Micali. Probabilistic encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
15. Marc Joye, Arjen K. Lenstra, and Jean-Jacques Quisquater. Chinese remaindering cryptosystems in the presence of faults. *Journal of Cryptology*, 12(4):241-245, 1999.
16. Marc Joye, Pascal Paillier, and Sung-Ming Yen. Secure evaluation of modular functions. In R.J. Hwang and C.K. Wu, eds., *Proc. of the 2001 International Workshop on Cryptology and Network Security (CNS 2001)*, pp. 227–229, Taipei, Taiwan, September 26–28, 2001.
17. Marc Joye, Jean-Jacques Quisquater, Feng Bao, and Robert H. Deng. RSA-type signatures in the presence of transient faults. In M. Darnell, ed., *Cryptography and Coding*, vol. 1355 of *Lecture Notes in Computer Science*, pp. 155–160, Springer-Verlag, 1997.
18. Marc Joye, Jean-Jacques Quisquater, and Moti Yung. On the power of misbehaving adversaries and security analysis of the original EPOC. In D. Naccache, ed., *Topics in Cryptology – CT-RSA 2001*, vol. 2020 of *Lecture Notes in Computer Science*, pp. 208–222, Springer-Verlag, 2001.
19. Burton S. Kaliski Jr. Comments on a new attack on cryptographic devices. RSA Laboratories Technical Note, October 23, 1996.
20. Çetin K. Koç. RSA hardware implementation. Technical Report TR 801, RSA Laboratories, April 1996.
21. Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397, Springer-Verlag, 1999.
22. James Manger. A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS #1. In J. Kilian, ed., *Advances in Cryptology – CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 230–238, Springer-Verlag, 2001.
23. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *Proc. of the 22nd ACM Annual Symposium on the Theory of Computing (STOC '90)*, pp. 427–437, ACM Press, 1990.
24. Andrew Odlyzko. The future of integer factorization. *Cryptobytes*, 1(2):5–12, 1995.
25. Jean-Jacques Quisquater and Chantal Couvreur. Fast decipherment algorithm for RSA public-key cryptosystem. *Electronics Letters*, 18:905–907, 1982.
26. Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In J. Feigenbaum, ed., *Advances in Cryp-*

tology – CRYPTO '91, vol. 576 of *Lecture Notes in Computer Science*, pp. 433–444, Springer-Verlag, 1992.
27. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
28. Adi Shamir. RSA for paranoids. *Cryptobytes*, 1(2):1–4, 1995.
29. Adi Shamir. How to check modular exponentiation. Presented at the rump session of *EUROCRYPT '97*, Konstanz, Germany, 11–15th May 1997.
30. Victor Shoup. OAEP reconsidered. In J. Kilian, ed., *Advances in Cryptology – CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001.
31. Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on Computers*, 49(9):967–970, 2000.