Side-Channel Attacks on Textbook RSA and ElGamal Encryption

Ulrich Kühn

Dresdner Bank, IS-STA 5, Information Security
D-60301 Frankfurt, Germany
ulrich.kuehn@dresdner-bank.com
ukuehn@acm.org

Abstract. This paper describes very efficient attacks on plain RSA encryption as usually described in textbooks. These attacks exploit side channels caused by implementations that, during decryption, incorrectly make certain assumption on the size of message. We highlight different assumptions that are easily made when implementing plain RSA decryption and present corresponding attacks.
These attacks make clear that plain RSA is a padding scheme that has to be checked carefully during decryption instead of simply assuming a length of the transported message.
Furthermore we note that the attacks presented here do also work against a similar setting of ElGamal encryption with only minimal changes.

Keywords: RSA encryption, ElGamal encryption, Side-channel attack.

1 Introduction

In general RSA is described as the modular exponentiation applied directly to a message M – plain RSA. Boneh, Joux and Nguyen [6] have shown that this method is insecure when the bit-length m of the encrypted message is fixed to a small amount of bits, say 64 bits, by giving a meet-in-the-middle attack that uses $2 \cdot 2^{m/2}$ modular exponentiations and $2^{m/2}m$ bits of memory; this result is independent of the size of the modulus. But if longer messages are involved, e.g. 128 bits or more in length, this method becomes impractical. Attacks against RSA encryption of messages with related or stereotyped content do not seem applicable when only random session keys of considerable size are encrypted as part of a hybrid encryption scheme (see Boneh [5] for an overview of attacks on RSA).

On the other hand, attacks using side channels caused by incorrect implementations have been presented by Bleichenbacher [3] against PKCS #1 v1.5 RSA block type 2 padding as well as by Manger [12] against PKCS #1 v2.0 (RSA-OAEP); here the adversary learns one bit of information about the resulting plaintext from a server that sends detailed error codes for different failures in the decryption process. The query complexities of these attacks essentially depend exponentially on the number of bits that the padded message is shorter

Y.G. Desmedt (Ed.): PKC 2003, LNCS 2567, pp. 324–336, 2003.

than the modulus; the attacks are efficient because for the PKCS #1 padding schemes these numbers are 16 resp. 8 bits for usual choices of the length of the modulus.

In this paper we present attacks that exploit side channels in plain RSA encryption and make use of the homomorphic property of RSA. The side channels result from implementation flaws in the decryption process, namely certain assumptions about the message being encrypted; we argue that these assumptions are easily made when implementing the decryption of plain RSA.

We model the decryption process in a server as an oracle that tests whether a given message is – under some equivalence relation specific to the oracle – equivalent to the original message that was given to the adversary as a challenge. The oracle-specific assumptions on the message are either that the relevant message bits (the session key) occur at certain positions or that the message has a specific, a-priori known (short) size when interpreted as an integer. The latter assumption is similar to the one used by Manger [12], but here we have to deal with messages that are a lot shorter than the modulus.

We present the *approximation attacks* that can very efficiently break the confidentiality of the messages using the oracles that assume positions of the message bits, provided that the size of the message is at most about one third of the size of the modulus. We also present the *divisibility attack* that works when the adversary can gain one bit of knowledge about the size of a (short) decrypted message. All our attacks are independent of the size of the modulus, they depend only on the size of the encrypted message.

Related to our attacks are the attacks of Bleichenbacher [3] and Manger [12], as all these attacks exploit side channels that result from incorrect implementations of the RSA decryption and unpadding process. The query complexity of Bleichenbacher's resp. Manger's attack e.g. for a message encrypted under a 1024-bit modulus is about one million resp. roughly a thousand queries; the approximation attacks need only 130 queries to reveal a 128-bit message, while an improved divisibility attack requires for example roughly 7000 queries and 2^{32} offline work with a probability of success of about one in 82 messages. It should be noted that our attacks do not work for PKCS #1 padded messages, while Bleichenbacher's and Manger's attacks effectively do not work in the setting of plain RSA.

The practical aspect of our attacks is that they exploit implementation errors one can easily step into. We found that the home banking system HBCI, an early version of the e-government protocol OSCI (the current specification uses PKCS #1 padding) and an early (now obsolete) version of PEM do apply plain RSA, and we are aware of implementations seemingly vulnerable to the approximation attacks.

The rest of this paper is organised as follows. Section 2 describes the different types of server oracles that are examined here, Sections 3 to 5 give attacks on each of these oracles, while Section 6 shows that nearly the same attacks might work against improperly implemented plain ElGamal encryption. Then Section 7

discusses the practical impact on the above-mentioned systems in more detail. Finally some conclusions are drawn.

2 Decryption Oracles

While we model our attacks in terms of oracles they should be viewed with a hybrid encryption setting in mind.[1] We further assume that redundancy to check for decryption errors is present. In the sequel we refer to the session key also as the *message* of the plain RSA encryption.

When a message M of fixed and a-priori known size is transmitted after encryption with plain RSA useful information might leak from the receiver, depending on the way the receiver reacts on the result of decryption. In the sequel we will refer to the receiver of the message as the server.

Throughout this paper we will use the following notation:

Notation. We denote the size of a value X measured in bits by $L(X)$. Let N be an RSA modulus, i.e. $N = PQ$ for two primes P and Q, and e resp. d the public resp. private exponent. When using modular reduction, we identify integers in the interval $[0, N)$ with the elements of $\mathbb{Z}_N$; elements of $\mathbb{Z}_N$ are represented by the least positive residue modulo N in the interval $[0, N)$. For y with $\gcd(y, N) = 1$ we use the notation $x/y \mod N$ to denote $x \cdot a \mod N$ with a being the inverse of y modulo N, i.e. the unique integer a, $0 < a < N$, such that $ay \equiv 1 \pmod N$.

Modeling the Oracles. Throughout this paper we will use the following definitions to refer to the oracles and the queries involved:

Definition. We model the server as an oracle $\mathcal{O}$, holding a secret $M_{\mathcal{O}}$ known only to the oracle. We assume that $M_{\mathcal{O}} < B$ for a fixed, publicly known bound $B = 2^b$, i.e. $M_{\mathcal{O}}$ has at most b bits. $\mathcal{O}$ has an associated public RSA-key (N, e) and a private exponent d. The adversary is given the public key (N, e) and the challenge ciphertext $C_{\mathcal{O}} = M_{\mathcal{O}}^e \mod N$.
The oracle answers a query $\mathcal{O}(C)$ either with success or failure, depending on whether the decrypted $M = C^d \mod N$ is equivalent to $M_{\mathcal{O}}$ with the equivalence relation depending on the oracle actually used. The oracle treats the query message $M = C^d \mod N$ after RSA decryption in one of the following ways, depending on the oracle-specific secret $M_{\mathcal{O}}$:

LSB-oracle: The oracle uses the least significant b bits of M, i.e. checks if $M \equiv M_{\mathcal{O}} \pmod B$.

[1] A payload is encrypted under a random session key using symmetric cryptography while the session key itself is encrypted with plain RSA; both ciphertexts are transmitted to the receiver. Some redundancy, i.e. a MAC or a digital signature is assumed to be present to distinguish between a correct and an incorrect decryption.

MSB-oracle: If $L(M) < L(B)$, then the oracle checks that $M = M_{\mathcal{O}}$. If M has $L(M) \geq L(B)$ bits, the most significant b bits of M (i.e. the value $\lfloor M/2^{(L(M)-b)} \rfloor$) are tested if they equal $M_{\mathcal{O}}$.
A variant checks if the most significant octets, words etc. of M, e.g. the most significant $\lceil \frac{b}{8} \rceil$ octets[2] of M equal $M_{\mathcal{O}}$, i.e. uses the value

$$\left\lfloor \frac{M}{2^{8(\lceil \frac{L(M)}{8} \rceil - \lceil \frac{b}{8} \rceil)}} \right\rfloor .$$

Size-checking oracle: The oracle checks that $M < B$, i.e. checks if M is conforming to the size assumption on the message.

Remark 1. Note that none of these oracles represents a correct implementation of the decryption process for plain RSA.

Remark 2. A server can be modeled as the size-checking oracle if it provides error codes that allow to distinguish between a message $M \geq B$ and a message $M < B$ that is incorrect for some other reason, i.e. does not yield a valid decryption of the payload in a hybrid encryption scheme. This oracle has been used in Manger's attack on RSA-OAEP [12], but his attack does not work efficiently if B is much shorter than the modulus.

3 Attacking the LSB-Oracle

The attack on the LSB-oracle is based on two crucial observations: First, encryptions of multiples of $M_{\mathcal{O}}$ of the form $M = (aB + 1)M_{\mathcal{O}}$ result, as $M \equiv M_{\mathcal{O}} \pmod{B}$, in the oracle answering success whenever $M < N$; if a is too large (thus $M \geq N$) then the modular reduction in the RSA decryption is likely[3] to yield an incorrect message.

Second, if z is such that $zM_{\mathcal{O}} \approx N$, $zM_{\mathcal{O}} < N$, then $M_{\mathcal{O}} \approx \lfloor N/z \rfloor$ with a small margin of error. We will show below that an approximation of the most significant b plus a small constant number of bits of N are sufficient to work reliably. Therefore we call this kind of attack the *approximation attack*.

The idea of the attack is to use the homomorphic property of RSA to compute ciphertexts from the oracle's challenge (the encryption of $M_{\mathcal{O}}$) yielding multiples of $M_{\mathcal{O}}$. Furthermore the modular reduction in the RSA encryption / decryption operation indicates whether $M < N$ or not; this information is employed in

[2] Such a behavior can occur if the long integer package used for implementation returns numbers without leading zeros and the integer-to-octet-array conversion places the highest octets first into the array.

[3] The heuristic here is that if $M = (aB + 1)M$ is only slightly larger than N, changes are wrapped by the modular reduction into the result of decryption and truncation. This assumption might not hold if N has a special form $N \equiv c \pmod{B}$ where the least significant $L(c)$ bits of the session key are ignored (e.g. a Triple-DES key where the parity bits are ignored has $c = 1$).

a (truncated) binary search to find the approximation $M = zM_{\mathcal{O}} \approx N$. Informally speaking, an additional copy of the original message is shifted towards the more significant bits as far as possible without making it bigger than the modulus and is subsequently padded out by other copies not shifted that far while still keeping the result below the modulus. This is basically done by binary search.

The attack algorithm works as follows, where the bound B is assumed to be known to all parties. Furthermore, it is assumed that $M_{\mathcal{O}} \neq 0$ and N is of size $L(N) > 3b + c$, where c is some small constant ($c = 3$ is sufficient).

Input: LSB-oracle $\mathcal{O}$ with public RSA-key (N, e), challenge $C_{\mathcal{O}} = M_{\mathcal{O}}^e \mod N$ derived from $M_{\mathcal{O}}$ with $0 < M_{\mathcal{O}} < B$; $M_{\mathcal{O}}$ is known only to $\mathcal{O}$.
Output: The message $M_{\mathcal{O}}$.

1. Set $z \leftarrow \max\{2^w | w > 0 \text{ and } 2^w < N/B\}$, $b' \leftarrow b$.
2. Compute the query $C = C_{\mathcal{O}}(2z+1)^e \mod N$; if $\mathcal{O}(C) = \mathsf{success}$, set $z \leftarrow 2z$, $b' \leftarrow b' - 1$ and repeat this step.
3. Set $y \leftarrow z/2$.
4. Do for $b' + 1$ times:
   (a) Compute the query $C = C_{\mathcal{O}}(z + y + 1)^e \mod N$; if $\mathcal{O}(C) = \mathsf{success}$, set $z \leftarrow z + y$.
   (b) Set $y \leftarrow y/2$.
5. Compute a candidate $\hat{M} \leftarrow \lfloor N/(z+1) \rfloor$ for $M_{\mathcal{O}}$ and return it.

To show the correctness of the attack algorithm we show that a partial approximation of N is sufficient.

Proposition 1. *Let N, b, $k < b$ be positive integers; let $B = 2^b$. If $N > 2^{b+k}$ then for each integer M, $0 < M < B$ there exists an integer z with*

$$N - zM < 2^{L(N)-1-k}, \tag{1}$$

i.e. an approximation of the most significant k bits of N by multiples of M is possible.

Proof. Equation 1 can be fulfilled if $0 < M < 2^{L(N)-1-k}$, as the points in the set $\{iM \mid i \in \mathbb{Z}\}$ have a distance of less than the error bound, and thus a point exists with the claimed property.

As Equation 1 has to hold for all $M < B$, this can be fulfilled if $B = 2^b \leq 2^{L(N)-1-k}$, thus $2^{L(N)-1-k-b} \geq 1$. As $N > 2^{L(N)-1}$, the claim follows. □

Proposition 2. *Let N, b, $k < b$ be positive integers such that $N > 2^{b+k}$; let $B = 2^b$ and $M \in \{0, \ldots, B-1\}$. Let z be a positive integer such that $zM < N$ and $N - zM < 2^{L(N)-k-2}$, i.e. the most significant $k+2$ bits of zM and N match. With $M' := \lfloor N/z \rfloor$ the approximation error is bounded by $|M - M'| < 2^{b-k-1}$.*

Proof. The existence of such a z follows from Proposition 1. To prove the error bound, we use the first order Taylor expansion of $N/(z+x)$ at z with $x \geq 0$:

$$\frac{N}{z+x} = \frac{N}{z} - \frac{Nx}{\xi^2} \quad \text{with } \xi \in [z, z+x] \tag{2}$$

and thus

$$\delta(x) \stackrel{\text{def}}{=} \frac{N}{z} - \frac{N}{z+x} = \frac{Nx}{\xi^2} \leq \frac{Nx}{z^2} \text{ , as } \xi \geq z. \tag{3}$$

From $M < B$ it follows that $z > N/B$, and as $x = (N - zM)/z$, we have $x < 2^{L(N)-k-2}/z$. Thus

$$\delta(x) < \frac{N2^{L(N)-k-2}B^3}{N^3} < \frac{B^3}{2^{k+1}N} < 2^{b-k-1}, \tag{4}$$

as $B = 2^b$, $N > 2^{L(N)-1}$ and $N > B^2$. □

The condition $L(N) > 3b + c$ for the algorithm is necessary in order to have at all times the correct message being present as the b least significant bits plus the approximation using copies of $M_{\mathcal{O}}$ shifted to the more significant bits while the least significant bits of the shifted copies must not interfere with the most significant bits of $M_{\mathcal{O}}$ in the original position (as these bits are used by the oracle to check the equivalence).

The first two steps of the attack algorithm make sure that leading zero bits in $M_{\mathcal{O}}$ are taken care of, thus allowing to apply Proposition 2 using a modified bound $B' = 2^{L(M_{\mathcal{O}})}$. Step 4 is basically binary search with the condition that $M_{\mathcal{O}}(z + y + 1) < N$ during each iteration. The loop count makes sure that the error condition of Proposition 2 with $k = b$ is fulfilled; thus Step 5 results in $\hat{M} = M_{\mathcal{O}}$ as the approximation error is less than 1/2.

The algorithm is very efficient as it uses only $b + 2$ queries, thus proving the intuition that the oracle leaks about one bit per query. Furthermore it should be noted that this attack is independent of the size of the modulus, as long as the condition for the size of B and N are fulfilled. The algorithm requires nearly no memory.

4 Attacking the MSB-Oracle

The MSB-oracle can be attacked with a similar method as the LSB-oracle in the last section. In the sequel we will be mostly concerned with the variant that uses the most significant $L := \lceil \frac{b}{8} \rceil$ octets of the decrypted query message M (see Section 2). An adaption to the situation where this is done for single bits or words is rather straight-forward.

Remark 3. In order for this attack to work it is necessary that the message $M_{\mathcal{O}}$ has full length in octets, i.e. $\lceil L(M_{\mathcal{O}})/8 \rceil = L$, as otherwise any shifted copy of $M_{\mathcal{O}}$ would result in the oracle reporting failure. A random message M is expected to have full length in octets in 255 out of 256 cases.

Roughly speaking, the essential observation is that any multiple of $M_{\mathcal{O}}$ of the form $M = (2^{8Lc} + z)M_{\mathcal{O}}$ for $z < 2^{8(L(c-1))}$, $c > 1$, results in the oracle reporting success whenever $M_{\mathcal{O}}$ has full length, i.e. consists of L octets. Then

the bound 2^{8Lc} can be approximated by multiples of $M_{\mathcal{O}}$. Thus this is another instance of the approximation attack.

In order to use Proposition 2 with $k = b$ it is necessary to have slightly more than $8L$ bits in an approximation, thus we have to use at least one more octet. The attack algorithm approximates $2^{8(Lc+1)}$ for $c = 2$ by a multiple $zM_{\mathcal{O}}$ by binary search in the same way as N is approximated in Section 3. The attack algorithms works for $N > 2^{8(Lc+1)}B$.

Input: MSB-oracle $\mathcal{O}$ with public RSA-key (N, e), challenge $C_{\mathcal{O}} = M_{\mathcal{O}}^e \mod N$ derived from $M_{\mathcal{O}}$ with $2^{8(L-1)} \leq M_{\mathcal{O}} < 2^{8L}$; $M_{\mathcal{O}}$ is known only to $\mathcal{O}$.
Output: The message $M_{\mathcal{O}}$.

1. Set $f \leftarrow 2^{8(2L+1)}$, $z \leftarrow 2^{8(L+1)}$, $b' \leftarrow b$.
2. Compute the query $C = C_{\mathcal{O}}(f+z)^e \mod N$; if $\mathcal{O}(C) = \mathsf{success}$, set $z \leftarrow 2z$, $b' \leftarrow b' - 1$ and repeat this step.
3. Set $y \leftarrow z/2$.
4. Do for $b' + 1$ times:
   (a) Compute the query $C = C_{\mathcal{O}}(f+z+y)^e \mod N$; if $\mathcal{O}(C) = \mathsf{success}$, set $z \leftarrow z + y$.
   (b) Set $y \leftarrow y/2$.
5. Compute a candidate $\hat{M} \leftarrow \lfloor f/z \rfloor$ for $M_{\mathcal{O}}$ and return it.

The correctness of this attack algorithm can be seen in basically the same way as in the case of the LSB-oracle in Section 3, again using Proposition 2 with $k = b$.

The algorithm is as efficient as the one in Section 3, it uses only $b+2$ queries to retrieve the complete message $M_{\mathcal{O}}$, provided that $M_{\mathcal{O}}$ has the full length in octets; this is fulfilled for 255 out of 256 messages. Thus this oracle also leaks about one bit per query. Again, the algorithm needs virtually no memory.

Remark 4. A difficulty for the attack algorithm may arise if some bits of an RSA-encrypted session key are ignored, e.g. a Triple-DES key where the parity bits are ignored or corrected whenever they are incorrect. In this situation a cleared parity bit in the octet located at the least significant position will result in Step 2 in z being too big by a factor of 2; the reason is that the most significant set bit of $M_{\mathcal{O}}$ has to be shifted past this critical parity bit to lead to a negative result of the oracle query. This also happens in Step 4, thus leading to $\hat{M} = \lfloor M_{\mathcal{O}}/2 \rfloor$; therefore it is necessary to modify the last step of the attack algorithm as follows:

5'. Compute two candidates for $M_{\mathcal{O}}$ as $\hat{M}_1 \leftarrow \lfloor f/z \rfloor$ and $\hat{M}_2 \leftarrow \lfloor 2f/z \rfloor$,

which effectively leaves a single bit of entropy of the session key; this bit has to be determined by the adversary from other sources of redundancy like the parity bits or the symmetrically-encrypted message itself.
Interestingly, the fact that keys for Triple-DES employ odd parity results in the situation that all messages $M_{\mathcal{O}}$ have full length as a leading zero octet – which would have even parity – is impossible; thus all messages can be recovered by the attack.

5 Attacking the Size-Checking Oracle

Given the attacks in the previous sections one might try to fix the vulnerability by checking on the receiver's side that the decrypted message is below an a-priori fixed bound B. We show here that simply checking the size of the message after decryption may not be sufficient to prevent the attacks of this paper. When not done very carefully an implementation might actually implement a size-checking oracle leaking information to an adversary. This is a setting similar to that of Manger's attack on RSA-OAEP [12], although his attack cannot be applied efficiently in our situation here. This is due to the running time containing an expression exponential in N/B.

Definition 1. *(see [14]) A number n with factorisation $n = \prod_{i=1}^{k} p_i$ with p_i prime is called* y-smooth *if $p_i \leq y$ for $1 \leq i \leq k$.*

Definition 2. *(see [1]) A number n with factorisation $n = \prod_{i=1}^{k} p_i$ with p_i prime and $p_i \geq p_{i+1}$ is called* semismooth with respect to y and z *if $p_1 \leq y$ and $p_2 \leq z$. We also say in this case that n is (y, z)-semismooth.*

The next Proposition essentially allows to convert the size-checking oracle into an oracle that answers requests of the type "Does p divide the secret message $M_{\mathcal{O}}$?", i.e. a test for divisibility. Therefore we call the attack based on this oracle the *divisibility attack*.

Proposition 3. *Let $B < N$, $x \in \{0, \ldots, B-1\}$ and a such that $1 \leq a < N/B$ and $\gcd(a, N) = 1$. Let $y = x/a \mod N$. Then $y \in [0, B)$ if and only if $a|x$.*

Proof. The claim is clear for $a|x$. Assume now $y \in [0, B)$. Then we have $ay \in \{0, a, 2a, \ldots, (B-1)a\}$. Furthermore, for $0 \leq t \leq (B-1)$ it is $ta < N$ as $a < N/B$, i.e. no wrap-around occurs when reducing modulo N. It follows that $ay = a\ y \mod N$ and thus $ay = x$. We find that $x \in \{aj | 1 \leq j < B\}$, i.e. $a|x$. □

The attack based on the oracle provided by Proposition 3 tries to extract a large smooth part of the message $M_{\mathcal{O}}$ and to find the rest of the message by other means, e.g. by offline search using brute force.

The attack algorithm is as follows:

Parameters: Fix a bound S for the smooth part of $M_{\mathcal{O}}$ and a bound T for the amount of offline work; both S and T may depend on the bound B.
Input: A size-checking oracle $\mathcal{O}$ with public RSA-key (N, e) and a challenge $C_{\mathcal{O}} = M_{\mathcal{O}}^e \mod N$ derived from $M_{\mathcal{O}}$ with $0 < M_{\mathcal{O}} < B$; $M_{\mathcal{O}}$ is known only to $\mathcal{O}$.
Output: If successful, the message $M_{\mathcal{O}}$.
State variables: m', the currently known smooth part of $M_{\mathcal{O}}$; c', the RSA encryption of $M_{\mathcal{O}}/m'$.

1. Initialise $m' \leftarrow 1$, $c' \leftarrow C_{\mathcal{O}}$.
2. For each prime $p \leq S$ (assuming $\gcd(p, N) = 1$)
   (a) Compute the next query $C = c'/p^e \mod N$.
   (b) If $\mathcal{O}(C) = \mathsf{success}$ (i.e. $p \mid \frac{M_{\mathcal{O}}}{m'}$), set $m' \leftarrow pm' \mod N$, $c' \leftarrow c'/p^e \mod N$, and go to step 2a.
   (c) Otherwise, go to the next prime.
3. For each $m_t \leq T$ test if $c' = m_t^e \mod N$. If so, return $\hat{M} \leftarrow m_t m' \mod N$ as candidate for $M_{\mathcal{O}}$.
4. If the execution path arrives here, abort, as $M_{\mathcal{O}}$ cannot be recovered with these settings.

The correctness of the attack algorithm is based on an invariant that is maintained in Steps 1 and 2:

$$C = (m')^e c' \mod N. \tag{5}$$

During Step 2, m' resp. c' are modified if and only if Proposition 3 says that the decryption of c' is divisible by p. As a result of the loop the S-smooth part of $M_{\mathcal{O}}$ is extracted. Finally, Step 3 tries to find the non-S-smooth part of $M_{\mathcal{O}}$.

Selection of Parameters. The attack works for those messages $M_{\mathcal{O}}$ that are divisible by primes $p < S$ possibly except for a factor bounded by T. Thus the probability of success is given by the probability that a random message is of this form.

For certain choices of S and T this is related to the concept of semismooth numbers, as the attack works for all numbers that are (T, S)-semismooth. Therefore probabilities of semismoothness give a lower bound on the success probabilities of the attack (see [1, Table 1]). On the other hand, if $T < S^2$ this bound is also tight, as the non-S-smooth part m_t of $M_{\mathcal{O}}$ must be a prime in this case.

The following example indicates that an adversary can find the message with a non-negligible advantage:

Example 1. Assume that keys for 2-key Triple-DES with correct parity, i.e. keys with 128 bits but only 112 bits of entropy, are encrypted by plain RSA using a 1024-bit modulus. Thus, $B = 2^{128}$. Using $S = 2^{22}$ and $T = 2^{42}$ the bound provided by the semismoothness probability is tight. Experiments with $22 \cdot 2^{14} = 360448$ random 128-bit numbers with adjusted parity yielded 559 numbers being (T, S)-semismooth, representing a fraction of $0.155\% \approx 2^{-9.3}$.

Improvements. The attack can be improved by combining it with the approximation attack, namely by using Proposition 2 with $k < b$. We will first describe the method with general parameters and then give a concrete setting.

Assuming that an adversary is willing to invest the queries to the server to find all factors of $M_{\mathcal{O}}$ below S using the divisibility attack, and further assuming that for the resulting rest $\hat{M}$ (free of divisors $\leq S$) $\hat{M} < T = 2^t$ holds, Proposition 2 guarantees that $b - t - 2$ further bits of $\hat{M}$ can be found with a process

similar to the approximation attacks of Sections 3 and 4 with $b-t$ queries, thus leaving $t-(b-t-2)=2t-b+2$ bits to be found by brute force.

The probability of success is that of a random number $M_{\mathcal{O}} < 2^b$ being divisible by primes below S and leaving a non-S-smooth remainder below T.

In a concrete setting using $b = 128$ (thus $M_{\mathcal{O}} < 2^{128}$), $S = 2^{16}$ and $t = 79, T = 2^{79}$ results in less than 7000 queries (see [14, Table 3]) to find the small prime factors of $M_{\mathcal{O}}$, approximation of 47 bits of $\hat{M}$ using 49 queries and finding of 32 bits by brute force. The probability of success has been experimentally found (using 2^{16} random numbers) to be about $2^{-6.4}$ or 1 in 82 random messages. Many other trade-offs are possible.

6 Attacking ElGamal Encryption

In [8] ElGamal encryption of a message $M_{\mathcal{O}}$ is described as $C = (g^r, M_{\mathcal{O}} y^r)$ where $p, g, y = g^x$ is the public key, x the private key, and computation is done in $\mathbb{Z}_p$. Similar to the RSA case, for plain ElGamal encryption no further padding is applied to the message before encryption. Plain ElGamal encryption has been shown in [6] to be attackable under certain conditions on the order of the generator g and the bit-size b of the messages using $2 \cdot 2^{b/2}$ modular exponentiations.

As the message $M_{\mathcal{O}}$ is masked by multiplying with y^r there is basically the same multiplicative property as in RSA (this was also noted by Bleichenbacher [4]): Given $(g^r, M_{\mathcal{O}} y^r)$, the ciphertext $(g^r, z M_{\mathcal{O}} y^r)$ decrypts to $z M_{\mathcal{O}}$.

This implies that the attacks presented above do also work for plain ElGamal encryption after minor adaptions without changing the analysis. It should be noted that, given a susceptible implementation, the attacks do not depend on the choice of p or the generator g, provided that $L(p) > 3b + c$ for small c and $M_{\mathcal{O}} < B = 2^b$.

Remark 5. While in [8] ElGamal encryption is proposed with multiplication modulo p to blind $M_{\mathcal{O}}$ with y^r, it is also noted there that other reversible operations could be used, e.g. bitwise XOR. This would destroy the multiplicative property used here and thus make the attacks much harder or even impossible.

The same problem exists with ElGamal encryption based on any other group, e.g. elliptic curves, provided that the blinding value is combined with the message by modular multiplication without any other precautions.

7 Practical Impact

In the sequel we will discuss three systems that make use of plain RSA encryption. It should be noted that the assessment that these systems might be susceptible to the attacks given in this paper is based purely on theoretical analysis of the specification. Generally, any system using plain RSA might be susceptible to our attacks if the highlighted implementation issues are present.

Furthermore it should be noted that only the confidentiality of payload messages is in danger, not their integrity, if the integrity of the messages is properly protected, e.g. by a digital signature.

7.1 HBCI

The Home Banking Computing Interface (HBCI) [9] is a protocol for providing a means of online banking to a bank's customers. To ensure confidentiality of the messages HBCI employs two alternatives: one is based fully on Triple-DES and will be ignored here, the other uses a hybrid encryption scheme on which we will focus. The actual banking messages (including a digital signature) are encrypted with 2-key Triple-DES in CBC mode under a random session key that is generated anew for each banking message; this session key is encrypted using plain RSA. The length parameters given in [9] are a modulus size of 768 bits for RSA and session keys of 128 bits for Triple-DES (including the correct parity bits).[4]

While it is possible to implement the encryption scheme such that our attacks do not work, we are aware of implementations that may be vulnerable. To be more precise, we found implementations of the cryptographic functionality with freely available source code, and, according to a source code inspection, they implement the LSB- and the MSB-oracle.[5] On the other hand we have no example of an implementation instantiating a size-checking oracle.

Interestingly the HBCI specification [9, Sect. VI.2.2.1] explains very carefully how to generate session keys and how to format them for encryption, but does not specify how the decryption and session key extraction have to be done.

7.2 OSCI

Another system that might be vulnerable is the (now outdated) version 1.0 of the Online Services Computing Interface (OSCI) [7] which is a proposal for an

[4] The recently published draft of version 3.0 [10] specifies, besides the old method, encryption of the message key using PKCS #1 padding and longer RSA keys; thus our attacks do not apply in this case, but one has to guard against the attack of [3].

[5] One is an implementation of an HBCI-Client which retrieves the session key from the last 16 octets of the result of the RSA decryption; these are the least significant octets. While the attack does not seem to work against a client, the attack from Section 3 can reveal any intercepted message with only 130 queries to a server that uses the same method.

The other is a generic implementation, suitable both for server and client. Here the session key is retrieved from an octet array with the leading zeroes being removed; but the result is not checked if it is too long, while too short arrays are padded. Thus the session key is constructed from the most significant 16 octets resulting from the RSA decryption. Here the attack from Section 4 needs only 130 queries to a server for obtaining the plaintext of an intercepted message. Remark 4 applies here as the parity is automatically corrected when it is incorrect.

XML-based e-government protocol. This version uses basically the same cryptographic functionality as HBCI to conceal messages, although with longer RSA keys (1023 to 1024 bits). Thus, depending on the actual implementation, a server implementing this protocol might instantiate one of the oracles examined here. Again the specification explains in great detail how to encrypt but not in detail how to decrypt.

Note that the newer version 1.2 as specified in [13] uses PKCS #1 v1.5 padding as part of XML encryption so that the attacks of this paper do not apply, but instead an implementation has to be guarded against [3].

7.3 Early Version of PEM

A third system that used plain RSA to transport session keys is an early, now obsoleted version of the Privacy-enhancement for Internet electronic mail (PEM) as specified in [11]. An important difference to the other systems mentioned above is that with PEM no server is normally present; instead, a mail reader performs the decryption under human supervision, so persuading the user to answer more than 100 queries might be difficult. But as newer versions [2] do not use plain RSA anymore this possible vulnerability does no longer exist.

8 Conclusion

In this paper we have shown that the decryption oracles examined here all result in insecure implementations of the decryption process for plain RSA, effectively destroying the confidentiality of the messages. The oracles are the result of assumptions on the messages during decryption; these assumptions are easily made when implementing the decryption process, we even found examples of seemingly vulnerable implementations.

In order to protect oneself against our attacks one has to rigorously check the padding with zeroes and the size of the message – this avoids the approximation attacks – and furthermore make sure that an adversary cannot get any information on the size of the decrypted message from error codes etc., effectively avoiding the divisibility attack. Our attacks show that plain RSA is indeed a padding scheme, contrary to the common belief that it does not use any padding.

It should be noted that a decryption process has not only to deal with valid ciphertexts but also with ciphertexts that are not the outcome of the corresponding encryption process; thus the specification of the decryption has to contain more than just the reversal to the encryption process. Indeed, the decryption process has to be specified and implemented very carefully.

Acknowledgment

The author is thankful to the anonymous referees for their helpful comments. Thanks are also due to Stefan Lucks for helpful discussions.

References

[1] E. Bach and R. Peralta. Asymptotic semismoothness probabilities. *Mathematics of Computation*, 65(216):1701–1715, 1996. 331, 332

[2] D. Balenson. RFC 1423: Privacy enhancement for Internet electronic mail: Part III: Algorithms, modes, and identifiers, Feb. 1993. Obsoletes RFC1115 [11]. 335, 336

[3] D. Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standards PKCS #1. In H. Krawczyk, editor, *Advances in Cryptology – CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12. Springer Verlag, 1998. 324, 325, 334, 335

[4] D. Bleichenbacher. Decrypting ElGamal messages. Message to ietf-open-pgp mailing list on imc.org, April 1999. `http://www.imc.org/ietf-open-pgp/mail-archive/msg02431.html`. 333

[5] D. Boneh. Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, 46(2):203–213, February 1999. 324

[6] D. Boneh, A. Joux, and P. Q. Nguyen. Why Textbook ElGamal and RSA Encryption Are Insecure. In T. Okamoto, editor, *Advances in Cryptology – ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 30–43. Springer Verlag, 2000. 324, 333

[7] Bremen Online Services. OSCI – Online-Services-Computer-Interface. Candidate for Version 1.0, November 2000. `http://www.bos-bremen.de/downloads/kap10_1.html`. 334

[8] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and D. Chaum, editors, *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag, 1985, 19–22 Aug. 1984. 333

[9] HBCI – Home Banking Computer Interface. Specification Version 2.2, May 2000. `http://www.hbci.de/`. 334

[10] HBCI – Home Banking Computer Interface. Draft Specification Version 3.0, July 2002. `http://www.hbci.de/`. 334

[11] J. Linn. RFC 1115: Privacy enhancement for Internet electronic mail: Part III — algorithms, modes, and identifiers, Aug. 1989. Obsoleted by RFC1423 [2]. 335, 336

[12] J. Manger. A Chosen Ciphertext Attack on RSA Optimal Asymmetric Encryption Padding (OAEP) as Standardized in PKCS #1 v2.0. In J. Kilian, editor, *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 230–238. Springer Verlag, 2001. 324, 325, 327, 331

[13] OSCI Leitstelle. OSCI-Transport Version 1.2, June 2002. See `http://www.osci.de/projekte/osci.html`. 335

[14] H. Riesel. *Prime Numbers and Computer Methods for Factorization*. Birkhäuser, 2nd edition, 1994. 331, 333