On the Broadcast and Validity-Checking Security of PKCS#1 v1.5 Encryption*

Aurélie Bauer[1], Jean-Sébastien Coron[2], David Naccache[1], Mehdi Tibouchi[1,2,**], and Damien Vergnaud[1]

[1] École normale supérieure – C.N.R.S. – I.N.R.I.A.
Département d'informatique, Groupe de cryptographie
45, rue d'Ulm, F-75230 Paris CEDEX 05, France
{aurelie.bauer, david.naccache, mehdi.tibouchi, damien.vergnaud}@ens.fr
[2] Université du Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg, Luxembourg
jean-sebastien.coron@uni.lu

Abstract. This paper describes new attacks on PKCS#1 v1.5, a deprecated but still widely used RSA encryption standard.

The first cryptanalysis is a broadcast attack, allowing the opponent to reveal an identical plaintext sent to different recipients. This is nontrivial because different randomizers are used for different encryptions (in other words, plaintexts coincide only partially).

The second attack predicts, using a *single* query to a validity checking oracle, which of two chosen plaintexts corresponds to a challenge ciphertext. The attack's success odds are very high.

The two new attacks rely on different mathematical tools and underline the need to accelerate the phase out of PKCS#1 v1.5.

Keywords: PKCS#1 v1.5, Encryption, Broadcast Encryption, Cryptanalysis.

1 Introduction

PKCS stands for *Public-Key Cryptography Standards* [14]. PKCS is a large *corpus* of specifications covering RSA encryption, Diffie-Hellman key agreement, password-based encryption, syntax (extended-certificates, cryptographic messages, private-key information and certification requests) and selected attributes. PKCS was initially developed by RSA Laboratories, Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell and Sun

* An extended abstract of this paper will appear at ACNS 2010. This is the full version.
** This research was completed while the fourth author was visiting the Okamoto Research Laboratory at the NTT Information Sharing Platform (Tokyo, Japan).

and regularly updated since. Today, PKCS has become part of several standards and of a wide range of security products including Internet Privacy-Enhanced Mail.

Amongst the PKCS collection, PKCS#1 v1.5 describes a particular encoding method for RSA encryption called `rsaEncryption`. In essence, the protected data is first encrypted under a randomly chosen key κ using a symmetric block-cipher (*e.g.* a triple DES in CBC mode) then κ is RSA-encrypted (wrapped) with the recipient's public key.

In 1998, Bleichenbacher [4] published an adaptive chosen-ciphertext attack against PKCS#1 v1.5 capable of recovering arbitrary plaintexts from about half a million ciphertexts. Although active adversary models are generally regarded as theoretical concerns, Bleichenbacher's attack makes use of an oracle that only detects conformance with respect to the padding format, a real-life assumption that resulted in a practical threat. PKCS#1 v1.5 was subsequently updated (release 2.0 [15]) and patches were issued to users wishing to continue using the old version of the standard. As we write these lines[3] and despite its vulnerabilities, PKCS#1 v1.5 *is still widely used*. Millions of (patched) PKCS#1 v1.5 programs remain deployed. Provably secure algorithms such as RSA-OAEP [10] and RSA-KEM are recommended replacements, but not widespread yet [16].

Independently, there exist several well-known chosen-plaintext attacks on RSA-based encryption schemes [7, 5]. These typically enable an attacker to decrypt ciphertexts at moderate cost without factoring the public modulus. The most powerful cryptanalytic tool applicable to low exponent RSA is certainly an attack due to Coppersmith [6]. As a matter of fact, one major reason for adding randomness to encrypted messages[4] is to thwart such attacks.

The last publication concerning PKCS#1 v1.5's security [8] presented a somewhat atypical attack allowing the opponent to retrieve plaintexts ending by enough zero bits.

This paper describes two new weaknesses in PKCS#1 v1.5:

- The possibility to predict, *using a single decryption query*, which of two chosen plaintexts corresponds to a challenge ciphertext. The attack's success odds are very high.
- A broadcast attack allowing to decrypt an identical message sent to several recipients.

[3] November 2009

[4] Besides attempting to achieve indistinguishability.

From a mathematical perspective, the two techniques are totally different. The authors regard these as a wake-up call to accelerate the phase out of PKCS#1 v1.5.

2 PKCS#1 v1.5 Encryption

We assume that the reader is familiar with the traditional public-key encryption definitions and security model preliminaries. For the sake of completeness we refer the reader to Appendix A.

2.1 The PKCS#1 v1.5 Encoding Function

PKCS#1 v1.5 describes a particular encoding method (`rsaEncryption`) for RSA encryption [19]. Consider an RSA modulus N, and let k denote its byte-length (*i.e.* $2^{8(k-1)} < N < 2^{8k}$). Let m be an $|m|$-byte message with $|m| < k - 11$. The PKCS#1 v1.5 padding $\mu(m)$ of m is defined as follows:

1. A randomizer r consisting in $k-3-|m| \geq 8$ nonzero bytes is generated uniformly at random;
2. $\mu(m) = \mu(m, r)$ is the integer converted from the octet-string:

$$\mu(m, r) = \mathtt{0002}_{16}||r||\mathtt{00}_{16}||m \tag{1}$$

   (the leading `00` octet guarantees that the encryption block is an integer smaller than N).

The encryption of a message m of $|m| < k - 11$ bytes is defined as

$$c = \mu(m, r)^e \bmod N$$

for some randomizer r of $k - 3 - |m|$ nonzero bytes.

To decrypt $c \in \mathbb{Z}_N^*$, compute $c^d \bmod N$, convert the result to a k-byte octet-string and parse it according to equation (1). If the string cannot be parsed unambiguously or if r is shorter than eight octets, the decryption algorithm $\mathcal{D}$ outputs $\perp$; otherwise, $\mathcal{D}$ outputs the plaintext m.

2.2 Previous attacks on PKCS#1 v1.5

In 1998, Bleichenbacher [4] published an attack on PKCS#1 v1.5 capable of recovering arbitrary plaintexts from a large number of ciphertexts validation queries. This attack established that PKCS#1 v1.5 is not ℓ-GOAL-ATTACK for a large[5] ℓ, $\mathsf{GOAL} \in \{\mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ and $\mathsf{ATTACK} \in \{\mathsf{VCA}, \mathsf{CCA}\}$ (see Appendix A.3 for definitions of these security notions).

In 2000, Coron, Naccache, Joye and Paillier [8] introduced two new CPAs on PKCS#1 v1.5. The first attack can be considered as a IND-CPA when e is small (for plaintext ending by sufficiently many zeroes). The second attack applies to arbitrary e, provided that $|m|$ is large and most message bits are zeroes. Thus PKCS#1 v1.5 is not GOAL-CPA for a small e or a large $|m|$, for $\mathsf{GOAL} \in \{\mathsf{IND}, \mathsf{NM}\}$.

$$\begin{array}{ccccc}
\mathsf{UBK\text{-}CCA} & \Longleftarrow & \mathsf{UBK\text{-}VCA} & \Longleftarrow & \mathsf{UBK\text{-}CPA} \\
 & & & & = \text{Factoring} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{OW\text{-}CCA} & \Longleftarrow & \cancel{\mathsf{OW\text{-}VCA}} & \Longleftarrow & \mathsf{OW\text{-}CPA} \\
 & & \text{large } \ell \text{ ([4])} & & \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{IND\text{-}CCA} & \Longleftarrow & \cancel{\mathsf{IND\text{-}VCA}} & \Longleftarrow & \cancel{\mathsf{IND\text{-}CPA}} \\
 & & \text{large } \ell \text{ ([4])} & & \text{small } e \text{ or large } |m| \text{ ([8])} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{NM\text{-}CCA} & \Longleftarrow & \mathsf{NM\text{-}VCA} & \Longleftarrow & \mathsf{NM\text{-}CPA}
\end{array}$$

Fig. 1. PKCS#1 v1.5 Security

The previous crytanalytic results are summarized in Figure 1. UBK-CPA is equivalent to Factoring but establishing the UBK-VCA and UBK-CCA security is equivalent to proving (or refuting) the equivalence of the factoring and the RSA Problem (which is a long-standing open question in cryptography). In the rest of the paper, we will study the remaining security notions and prove that PKCS#1 v1.5 is:

- OW-CPA assuming the intractability of the RSA problem (§ 3);
- not OW-CCA for $\ell = 2$ (§ 3);
- not NM-CPA (§ 4.1);
- not IND-VCA for $\ell = 1$ (§ 4.2);
- not OW-CPA in a multi-user setting (§ 5).

3 On PKCS#1 v1.5's OW-CPA-Security

In this paragraph, we prove the following result:

[5] $3 \cdot 10^5 < \ell < 2 \cdot 10^6$ for $512 < \log_2 N < 1024$

Proposition 1. *The OW-CPA security of* PKCS#1 v1.5 *is equivalent to the* RSA *Problem.*

In [9, Lemma 2], Coron, Joye, Naccache and Paillier proved that for suitable parameters, the existence of an algorithm that on input $y \in \mathbb{Z}_N^*$ outputs the k_1 least significant bits of $y^d \bmod N$ is equivalent to the existence of an RSA inverter. Following their approach, we prove the following lemma:

Lemma 1. *Let $\mathcal{A}$ be a OW-CPA-adversary against* PKCS#1 v1.5 *with success probability ε within time τ, with uniformly distributed messages of (maximum) length $k - 11$. There exists an algorithm $\mathcal{B}$ that solves the* RSA *Problem with success probability ε' within time τ', where:*

$$\begin{cases} \varepsilon' \geq \eta^2 \cdot \varepsilon^2 - 2^{-k}\varepsilon \\ \tau' \leq 2 \cdot \tau + \texttt{poly}(k) \end{cases}$$

where η is a constant independent of k and $\eta \geq 5 \cdot 10^{-8}$.

Proof. Let $\mathcal{A}$ be a OW-CPA-adversary against PKCS#1 v1.5 with success probability ε within time τ. We construct an algorithm $\mathcal{B}$ that on input $y \in \mathbb{Z}_N^*$ outputs y^d with success probability ε' within time τ':

1. $\mathcal{B}$ picks $\alpha \in \mathbb{Z}_N^*$ uniformly at random;
2. $\mathcal{B}$ sets $y_0 = y$ and $y_1 = y \cdot \alpha^e$
3. Let us denote, for $i \in \{0, 1\}$:

   $$y_i^d = \omega_i \cdot 2^\beta + m_i$$

   with $\beta = 8(k - 11)$. y_i is a valid PKCS#1 v1.5 ciphertext if $\omega_i = \mathsf{0002}||r_i||00$ where r_i is a 8-(nonzero)-octet string. This happens with probability $\eta \geq (255/256)^8 \cdot 2^{-24}$. If this happens, then with probability ε, $\mathcal{A}$ will return m_i on input y_i (for $i \in \{0, 1\}$).
4. Therefore with probability at least $(\eta\varepsilon)^2$, we obtain:

   $$\alpha(\omega_0 \cdot 2^\beta + m_0) = \omega_1 2^\beta + m_1 \bmod N.$$

   Letting $c_1 = 2^{-\beta}(\alpha m_0 - m_1) \mod N$, we get the equation in (ω_0, ω_1):

   $$\omega_1 - \alpha \cdot \omega_0 = c_1 \bmod N \tag{2}$$

From [9, Lemma 3], there exists an algorithm that given this system will output a solution (ω_0, ω_1) (should such a solution exist) with probability at least $1 - 2^{-k}$ on the choice of α. $\square$

Using the same technique, this can be extended to messages of different length, with a possibly higher constant loss in the reduction.

As a byproduct of the previous proof one immediately gets that:

Proposition 2. PKCS#1 v1.5 *is not* 2-*OW-CCA*.

Proof. Thanks to RSA's homomorphic properties, an adversary can mask the challenge ciphertext c as $c' = cr^e$ for some random r and with two decryption oracle queries, compute the e-th root x' of c' as in the previous proof. Then $x = x'/r$ is the e-th root of c from which, the adversary retrieves readily the plaintext. □

4 PKCS#1 v1.5 Malleability and Indistinguishability

We now show that μ is neither NM-CPA-secure nor IND-VCA-secure. The general idea is the following. Let m be some message to be encrypted, and $\mu(m)$ a corresponding PKCS#1 v1.5 padded encryption block. If m has $Z > 2$ trailing zero bits, then $\mu(m)$ is divisible by 2^Z, and $\mu(m) - \mu(m)/2^Z \bmod N$ has a good probability of still being a valid encryption block. This is not usually the case when the Z LSBs of m are not all zero.

Let $c = \mu(m)^e \bmod N$ be a ciphertext of some message m. If m has $Z > 2$ trailing zeroes, $c' = c \cdot (1 - 2^{-Z})^e \bmod N$ will often be a valid ciphertext of some other message m' which can be related to m: this contradicts non-malleability under chosen plaintext attack. Moreover, if one is granted *one* query to a validity oracle, it is possible to distinguish ciphertexts of plaintexts with trailing zeroes and ciphertexts from plaintexts whose LSBs are not all zero: this contradicts indistinguishability under validity checking attack. Note that if i queries are allowed, the distinguishing success odds can quickly approach one by iterating the test with $c'_i = c \cdot (i - 2^{-Z})^e \bmod N$ for $i = 1, 2, \ldots$

We will develop this idea in further detail in the coming sections.

4.1 On PKCS#1 v1.5's NM-CPA Security

Let k be the byte-size of N, $Z = 4k$, and M a positive integer such that $M + Z + 1$ is a multiple of 8 and $(M + Z + 1)/8 < k - 11$. We consider messages of the following form:

$$m = \underbrace{\bar{m}}_{M \text{ bits}} \| 1_2 \| \underbrace{0 \cdots 0_2}_{Z \text{ zero bits}}$$

Let $\mathcal{M}$ denote the uniform distribution over messages of this form. Furthermore, we define a relation $\mathcal{R}$ over messages of length $l = M + Z + 1$ as follows: for any l-bit two messages m_1, m_2 (not necessarily of the previous form), $\mathcal{R}(m_1, m_2)$ holds if and only if the M MSBs of m_1 and m_2 coincide. In particular, for any given message m_2, there is exactly one $m_1 \in \mathcal{M}$ such that $\mathcal{R}(m_1, m_2)$.

Now, consider $m \leftarrow \mathcal{M}$. We can write $\mu(m) \cdot (1 - 2^{-Z})$ as:

$$\begin{array}{rl} & \mathsf{0002}_{16}\|r\|\mathsf{00}_{16}\|\bar{m}\|\mathsf{1}_2\|\ \mathsf{0}\cdots\mathsf{0}_2 \\ - & \qquad\qquad\qquad\qquad \mathsf{0002}_{16}\|r\|\mathsf{00}_{16}\|\bar{m}\|\mathsf{1}_2 \\ \hline = & \mathsf{0002}_{16}\|r\|\mathsf{00}_{16}\|\bar{m}\|\mathsf{0}_2\| \text{ some digits} \cdots \text{ some digits} \end{array}$$

Hence, $\mu(m) \cdot (1 - 2^{-Z}) = \mu(m')$ for some message $m' \neq m$ such that $\mathcal{R}(m, m')$.

Consider, the NM-CPA adversary $\mathcal{A}$ which outputs sampling algorithm $\mathcal{M}$ in the setup stage, and transforms a challenge ciphertext c into $c' = c \cdot (1 - 2^{-Z})^e \bmod N$. $\mathcal{A}$'s advantage is:

$$\mathrm{Adv}_{\mathcal{A}}^{\mathsf{NM\text{-}CPA}} = \Pr[\mathcal{R}(m, m')] - \Pr[m_0 \xleftarrow{\$} \mathcal{M}; \mathcal{R}(m_0, m')] = 1 - 2^{-M} \geq 1/2$$

which is non-negligible. Therefore, PKCS#1 v1.5 encryption is not NM-CPA-secure.

Noted that $\mathcal{A}$'s advantage is, in fact ~ 1. For a 1024-bit modulus and a 128 bit randomizer, we have $M = 383$, making it exceedingly unlikely that $\mathcal{A}$ will ever fail.

4.2 On PKCS#1 v1.5's IND-VCA Security

We now show how to contradict ciphertext indistinguishability under validity checking attack using a single oracle query. There are two natural types of validity oracles: one which determines whether a given query is a valid ciphertext associated to a plaintext of any length, and the other which also checks message length. We can always contradict IND-VCA-security in the non-length-checking case (which is the one considered in Bleichenbacher's attack [4]) with a single oracle query. Furthermore, if the byte-length of the randomizer is *constant*, as permitted by the PKCS#1 v1.5 standard, it is also possible to break IND-VCA-security with a single query to a length-checking oracle. Both attacks stem from the following result.

Proposition 3. *Let $c = \mu(m)^e \bmod N$ be the ciphertext associated to some byte-string message m, ω the byte-length of the randomizer and $c' = c \cdot (1 - 2^{-4})^e \bmod N$.*

1. *If the least significant nibble of m is not $\texttt{0}_{16}$, then c' is never a valid ciphertext.*
2. *If m is a message consisting of a string of $\texttt{00}_{16}$ bytes, then c' is a valid ciphertext with probability at least 0.47. c' is a valid ciphertext corresponding to a message of the same length as m with probability at least:*

$$\frac{64}{1445}\left(\frac{239}{255}\right)^{\omega-1}$$

Proof. Starting with the first assertion, consider a message m such that c' is a valid ciphertext. This implies that $\mu(m) - \mu(m) \cdot 2^{-4} \bmod N$ is a valid encoding string that, in particular, begins with the same $\texttt{0002}_{16}$ pattern as $\mu(m)$.

If, for an integer x, we denote by $\bar{x}$ the only integer in $(-N/2, N/2)$ such that $x \equiv \bar{x} \mod N$, it follows that:

$$|\overline{\mu(m) \cdot 2^{-4} \bmod N}| < 2^{8k-16}$$

Consider the set S of residue classes $x \bmod N$ such that $|\bar{x}| < 2^{8k-16}$. Clearly, $|S| = 2^{8k-15} - 1$. On the other hand, let T^+ be the set consisting of k-byte strings of the form $\texttt{000}_{16}\|u\|\texttt{0}_{16}$, and T be the union of T^+ and $-T^+$ (where opposites are taken mod N). We also have $|T| = 2^{8k-15} - 1$ and T maps into S under multiplication by $2^{-4} \bmod N$. Since multiplication by 2^{-4} is a permutation of $\mathbb{Z}_N$, we infer that $(y \cdot 2^{-4} \bmod N) \in S$ if and only if $y \in T$.

In particular, if c' is a valid ciphertext, we get $\mu(m) \in T$. By inspection of its top bits, we see that $\mu(m)$ cannot be in $-T^+$, so it has to be in T^+. Its least-significant nibble must thus be $\texttt{0}_{16}$ as required.

Turning now to the second assertion, let m be the zero-message of some fixed byte-length. Write the encryption block $\mu(m)$ as follows:

$$\mu(m) = \texttt{0002}_{16}\|r_{2\omega-1}\|r_{2\omega-2}\|\cdots\|r_1\|r_0\|\texttt{00}_{16}\|\texttt{00}\cdots\texttt{00}_{16}$$

where $r_0, \ldots, r_{2\omega-1}$ are randomizer's nibbles. Recall that the randomizer bytes $r_{2j}\|r_{2j+1}$ are chosen uniformly and independently at random in the range $\texttt{01}_{16}$, ..., $\texttt{FF}_{16}$.

Assuming that $r_{2\omega-1}$ is at least 4 (which happens with probability $(256 - 4 \times 16)/255 = 64/85$, and which we will henceforth assume), we can write $(1 - 2^{-4})\mu(m)$ as:

$$\begin{array}{rcccccccc}
\texttt{0002}_{16} \;\| & r_{2\omega-1} & \| \; r_{2\omega-2} & \| \cdots \| & r_1 & \| \; r_0 \;\| & \texttt{00}_{16} & \| \; \texttt{00}\cdots\texttt{00}_{16} \\
-\;\texttt{0000}_{16} \;\| & \texttt{2}_{16} & \| \; r_{2\omega-1} & \| \cdots \| & r_2 & \| \; r_1 \;\| & r_0\|\texttt{0}_{16} & \| \; \texttt{00}\cdots\texttt{00}_{16} \\
\hline
=\;\texttt{0002}_{16} \;\| & r'_{2\omega-1} & \| \; r'_{2\omega-2} & \| \cdots \| & r'_1 & \| \; r'_0 \;\| & s & \| \; \texttt{00}\cdots\texttt{00}_{16}
\end{array}$$

where $r'_j \equiv r_j - r_{j+1} - \kappa_j \mod 16$ for some carry bit κ_j.

Then, $\mu' = (1 - 2^{-4})\mu(m)$ is a valid encoding block if and only if the first 8 randomizer bytes, namely $r'_{2\omega+1-2j} \| r'_{2\omega-2j}$, $j = 1, \ldots, 8$, are all nonzero. μ' is a valid encoding block for a message of the same length as m (or for short, "*strongly valid*") if and only if $s = 0$ and all the padding bytes $r'_{2j+1} \| r'_{2j}$, $j = 0, \ldots, \omega - 1$, are nonzero. We will find an explicit lower bound for the probability of these events.

Note first that a sufficient condition for $r'_{2j+1} \| r'_{2j}$ to be nonzero is that $r'_{2j+1} \neq 0$. This nibble is zero if and only if $r_{2j+2} \equiv r_{2j+1} - \kappa_{2j+1}$. Now r_{2j+2} is picked independently of r_{2j+1}, since they belong to different bytes; it is also independent of κ_{2j+1}, which only depends on lower order bytes. Consequently:

$$\begin{aligned}
\Pr[r'_{2j+1} = 0] &= \sum_{\rho=0}^{15} \Pr[r_{2j+2} = \rho] \cdot \Pr[r_{2j+1} - \kappa_{2j+1} \equiv \rho \mod 16] \\
&\leq \max_{\rho} \Pr[r_{2j+2} = \rho] = \frac{16}{255}
\end{aligned}$$

and this bound still holds conditionally to any assignment of the lower order nibbles r'_{2i+1}, $i < j$. Therefore:

$$\begin{aligned}
\Pr[\mu' \text{ is valid}] &\geq \Pr[r_{2\omega-1} \geq 4 \wedge r'_{2\omega-3} \neq 0 \wedge r'_{2\omega-5} \neq 0 \wedge \cdots \wedge r'_{2\omega-15} \neq 0] \\
&\geq \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^7 \geq 0.47
\end{aligned}$$

Furthermore:

$$\begin{aligned}
\Pr[\mu' \text{ is strongly valid}] &\geq \Pr[r_{2\omega-1} \geq 4 \wedge r'_{2\omega-3} \neq 0 \wedge \cdots \wedge r'_1 \neq 0 \wedge r_0 = 0] \\
&\geq \frac{64}{85} \cdot \left(1 - \frac{16}{255}\right)^{\omega-1} \cdot \frac{15}{255} = \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega-1}
\end{aligned}$$

The corresponding validity assertions for c' follow immediately. $\square$

Consider the $\mathsf{IND\text{-}VCA}$ adversary $\mathcal{A}$ defined as follows. In the setup stage, $\mathcal{A}$ outputs two equal-length messages m_0, m_1 with m_1 not zero-terminated (e.g. $\mathtt{00} \cdots \mathtt{0001}_{16}$) and m_0 consisting of $\mathtt{00}_{16}$ bytes only. Then, upon receiving a challenge ciphertext $c = \mu(m_b)^e \bmod N$, $\mathcal{A}$ makes a single oracle query and outputs $b' = 0$ or 1 according to whether $c' = c \cdot (1 - 2^{-4}) \bmod N$ is a valid ciphertext or not. Its advantage is then:

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{A}}^{\mathsf{IND\text{-}VCA}} &= \Pr[b' = 0 | b = 0] - \Pr[b' = 1 | b = 0] = \Pr[b' = 0 | b = 0] - 0 \\
&\geq \begin{cases} 0.47 & \text{if the oracle doesn't check message length} \\ \frac{64}{1445} \cdot \left(\frac{239}{255}\right)^{\omega-1} & \text{otherwise} \end{cases}
\end{aligned}$$

which is non-negligible. In the length-checking case, it is over 2.8% (resp. 1.6%) for 64-bit (resp. 128-bit) randomizers.

In the non-length-checking case, we can obtain an even better advantage using $c' = c \cdot (1 - 2^{-8}) \bmod N$ (shifting by 8 bits instead of 4). The proof works similarly provided that N satisfies $N > 2^{8k-7}$, which is not required by the PKCS#1 v1.5 standard but is usually verified in practice. This yields an advantage of at least:

$$\frac{252}{255} \cdot \left(\frac{254}{255}\right)^7 > 0.96$$

5 Broadcast Attacks on PKCS#1 v1.5

We now examine the security of PKCS#1 v1.5 in a multiple users context.

In such a scenario, *i.e.* when broadcast encryption is performed, the sender wishes to transmit the same message m to ℓ parties $P_1, \ldots, P_\ell$. As each party has its own key $\mathsf{pk}_i = (e, N_i)$ (with a common public exponent e), the sender encrypts m using all the pk_i's and sends the resulting ciphertexts $c_1, \ldots, c_\ell$ to the corresponding recipients. It has long been known that textbook RSA encryption should not be used in such a context, since an attacker can easily recover the plaintext using the Chinese Remainder Theorem as long as $\ell \geq e$. Therefore, m has to be padded before applying the RSA function, and the padding has to be different for each recipient.

In 1988, Håstad [11] showed that using different linear paddings $\mu_i(m)$ for all parties is not enough to guarantee security. Indeed, when e is small, e ciphertexts are again sufficient to efficiently recover m provided that the encoding functions μ_i are known to the attacker. To achieve this result, Håstad expressed the attack in terms of finding small roots of a univariate modular polynomial, which he accomplishes using Coppersmith's techniques [6].

Håstad's attack does not apply to PKCS#1 v1.5, since the padding used for a given recipient is random, and is thus unknown to an attacker. The following sections will overcome this difficulty. Our main result is as follows:

Proposition 4. *Let $c_1, \ldots, c_\ell$ be ℓ PKCS#1 v1.5 ciphertexts of the same message m, of byte length $|m|$. Each c_i is encrypted for a receiver having $\mathsf{pk}_i = (e, N_i)$. All N_i are k-byte long. Then there exists a heuristic algorithm which, given $c_1, \ldots, c_\ell$, outputs m, if:*

$$\ell > \frac{e|m|}{k - e(k - |m| - 3)} > 0$$

Its complexity is polynomial in $e, |m|$ *and* k *but exponential in the number of ciphertexts* ℓ.

We describe this algorithm in the coming sections. The core idea is to reduce the problem to finding small modular roots of a multivariate polynomial equation, which can be achieved using a standard generalization of Coppersmith's techniques. As usual, this generalization relies on an assumption concerning independence between polynomials, which makes the algorithm heuristic.

5.1 The Multivariate Polynomial of Broadcast PKCS#1 v1.5

Recall (section 2.1) that to encrypt message m for recipient P_i, a PKCS#1 v1.5 sender first generates an $|r_i|$-byte randomizer r_i, and then computes the encoding function:

$$\mu(m, r_i) = \mathsf{0002}_{16} \| r_i \| \mathsf{00}_{16} \| m$$

Numerically, this gives:

$$\mu(m, r_i) = m + 2^{8|m|+8} r_i + 2^{8|m|+8|r_i|+9}$$

The ciphertext c_i is then computed as $c_i = \mu(m, r_i)^e \bmod N_i$.

Consider then an adversary $\mathcal{A}$ who obtains $c_1, \ldots, c_\ell$. Since the N_i are of the same size, the randomizers r_i have a common byte length $|r|$. Therefore, the ciphertexts collected by $\mathcal{A}$ can be written as:

$$c_1 = (m + Ar_1 + B)^e \mod N_1, \quad \ldots, \quad c_\ell = (m + Ar_\ell + B)^e \mod N_\ell$$

where $A = 2^{8|m|+8}$ and $B = 2^{8|m|+8|r|+9}$. Obviously, the N_i are pairwise co-prime (otherwise, the factorization of some of the N_i could easily be recovered). Thus, the Chinese Remainder Theorem ensures that the previous equations can be rewritten as a single congruence mod $N = N_1 \cdots N_\ell$:

$$u_1 c_1 + \cdots + u_\ell c_\ell = u_1(m + Ar_1 + B)^e + \cdots + u_\ell(m + Ar_\ell + B)^e \mod N$$

where the constants $u_1, \ldots, u_\ell$ are given by the extended Euclidean algorithm. It follows that the tuple $(m, r_1, \ldots, r_\ell)$ is a root of the multivariate modular polynomial:

$$f(x, y_1, \ldots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N \qquad (3)$$

where $C = u_1 c_1 + \cdots + u_\ell c_\ell$. This root is *small* in the sense that all of its components are bounded by quantities that are small compared to N: m is smaller than $2^{8|m|}$ and each r_i is bounded by $2^{8|r|}$. Under suitable conditions on $|m|$ and $|r|$ which will be detailed below, it will thus become feasible to recover this root, and hence m, in polynomial time using Coppersmith's techniques, as recalled in Appendix B.

In particular, we show in section B.3 that the lattice construction of Jochemsz and May [13] yields a heuristic polynomial time algorithm for recovering the small root of f under the following condition:

$$\ell|r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k$$

where ν is a parameter which determines the attack's complexity, and $s(\nu)$, $t(\nu)$ are defined as:

$$s(\nu) = \sum_{i=0}^{e\nu} i \binom{e\nu - i + \ell}{\ell} = \binom{\ell + e\nu - 1}{\ell} \frac{(\ell + e\nu)(1 + \ell + e\nu)}{2 + 3\ell + \ell^2}$$

and

$$t(\nu) = \sum_{i=1}^{\nu} \binom{e\nu - ie + \ell + 1}{\ell + 1}$$

We also prove in the same section that $t(\nu)/s(\nu) \to 1/e$ as ν tends to $+\infty$, so that the best achievable bound on $|m|$ and $|r|$ for which the attack applies is:

$$\ell|r| + |m| < \frac{\ell k}{e}$$

Since $|r| = k - |m| - 3$ in PKCS#1 v1.5 we obtain the bound announced in Proposition 4. As usual when using Coppersmith's techniques, the complexity of the attack is polynomial in the dimension of the constructed lattice and in the size of the entries (see Appendix B for details on these parameters).

Given PKCS#1 v1.5's widespread use and the heuristic nature of multivariate Coppersmith-like techniques, it is important to *practically assess* our attack. We report practical experiment results in Appendix C.

6 Conclusion

Figure 2 summarizes our current knowledge of PKCS#1 v1.5 security.

The authors regard the new flaws as an indication that the process of phasing out PKCS#1 v1.5 should be accelerated.

$$
\begin{array}{ccccc}
\textsf{UBK-CCA} & \Longleftarrow & \textsf{UBK-VCA} & \Longleftarrow & \textsf{UBK-CPA} = \text{Factoring} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\cancel{\textsf{OW-CCA}} & \Longleftarrow & \cancel{\textsf{OW-VCA}} & \Longleftarrow & \textsf{OW-CPA} = \textsf{RSA}, \text{ single user } (\S\ 3) \\
\ell = 2\ (\S\ 3) & & \text{large } \ell\ ([4]) & & \cancel{\textsf{OW-CPA}} \text{ small } e \text{ and large } |m|, \text{ multi-user } (\S\ 5) \\
\Downarrow & & \Downarrow & & \Downarrow \\
\textsf{IND-CCA} & \Longleftarrow & \cancel{\textsf{IND-VCA}} & \Longleftarrow & \cancel{\textsf{IND-CPA}} \\
 & & \text{large } \ell\ ([4]) & & \text{small } e \text{ or large } |m|\ ([8]) \\
 & & \ell = 1\ (\S\ 4.2) & & \\
\Downarrow & & \Downarrow & & \Downarrow \\
\textsf{NM-CCA} & \Longleftarrow & \textsf{NM-VCA} & \Longleftarrow & \cancel{\textsf{NM-CPA}}\ (\S\ 4.1)
\end{array}
$$

Fig. 2. Updated Security Status for PKCS#1 v1.5

References

1. O. Baudron, D. Pointcheval, J. Stern, *Extended notions of security for multicast public key cryptosystems*, in *Automata, Language and Programming*, LNCS, vol. 1853, Springer-Verlag, 2000, pp. 499–511.
2. M. Bellare, A. Boldyreva, S. Micali, *Public-key encryption in a multi-user setting: Security proofs and improvements*, Proceedings of Eurocrypt 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 259–274.
3. M. Bellare, A. Desai, D. Pointcheval and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, Proceedings of Crypto 1998, LNCS, vol. 1462, Springer-Verlag, 1998, pp. 549–570.
4. D. Bleichenbacher, *Chosen ciphertext attacks against protocols based on the RSA encryption standard*, Proceedings of Crypto 1998, LNCS, vol. 1462, Springer-Verlag, 1998, pp. 1–12.
5. D. Coppersmith, *Finding a small root of a univariate modular equation*, Proceedings of Eurocrypt 1996, LNCS, vol. 1070, Springer-Verlag, 1996, pp. 155–165.
6. D. Coppersmith, *Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities*, Journal of Cryptology, 10(4), 1997, pp. 233–260.
7. Y. Desmedt, A.M. Odlyzko, *A chosen text attack on the RSA cryptosystem and some discrete logarithm schemes*, Proceedings of Crypto 1985, LNCS, vol. 218, Springer-Verlag, 1985, pp. 516–522.
8. J.-S. Coron, D. Naccache, M. Joye and P. Paillier, *New attacks on PKCS#1 v1.5 encryption*, Proceedings of Eurocrypt 2000, LNCS, vol. 1807, Springer-Verlag, 2000, pp. 369–381.
9. J.-S. Coron, D. Naccache, M. Joye and P. Paillier, *Universal Padding Schemes for RSA*, Proceedings of Crypto 2002, LNCS, vol. 2442, Springer-Verlag, 2002, pp. 226–241.
10. E. Fujisaki, T. Okamoto, D. Pointcheval and J. Stern, RSA-OAEP *is secure under the* RSA *assumption*, Journal of Cryptology, 17(2), 2004, pp. 81–104.
11. J. Håstad, *Solving simultaneous modular equations of low degree*, SIAM Journal on Computing, vol. 17(2), SIAM, 1988, pp. 336–341.
12. N. Howgrave-Graham, *Finding small roots of univariate modular equations revisited*, Proceedings of Cryptography and Coding, LNCS, vol. 1355, Springer-Verlag, 1997, pp. 131–142.

13. E. Jochemsz, A. May, *A strategy for finding roots of multivariate polynomials with new applications in attacking* RSA *variants*, Proceedings of Asiacrypt 2006, LNCS, vol. 4284, Springer-Verlag, 2006, pp. 267–282.
14. B. Kaliski, PKCS*#1:* RSA *Encryption Standard, Version 1.5*, RSA Laboratories, November 1993.
15. B. Kaliski, PKCS*#1:* RSA *Encryption Standard, Version 2.0*, RSA Laboratories, September 1998.
16. B. Kaliski, RSA Laboratories, personal communication, October 2009.
17. A. K. Lenstra, H. W. Lenstra and L. Lovàsz, *Factoring polynomials with rational coefficients*, Math. Annalen, vol. 261, pp. 513–534, 1982.
18. D. Pointcheval, *Provable security for public-key schemes*, in *Contemporary cryptology*, Advanced courses in mathematics, Birkhäuser Basel, 2005, pp. 133–190.
19. R.L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM, vol. 21(2), ACM, 1978, pp. 120–126.

A Preliminaries

In this appendix we recall a few basic definitions necessary for an accurate description of our attack:

A.1 Public-Key Encryption

A *Public-Key Encryption Scheme* $\mathcal{P} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ is a collection of three (probabilistic) algorithms:

A Key Generation Algorithm $\mathcal{K}$: Given a security parameter $k \in \mathbb{N}$, the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and private keys (which implicitly define a message space $\mathcal{M}$).

An Encryption Algorithm $\mathcal{E}$: Given a message $m \in \mathcal{M}$ and a public key pk, $\mathcal{E}$ produces a ciphertext $c = \mathcal{E}(\mathsf{pk}, m)$.

A Decryption Algorithm $\mathcal{D}$: Given a ciphertext c and the private key sk, $\mathcal{D}(\mathsf{sk}, c)$ recovers a plaintext m or a special symbol $\perp \notin \mathcal{M}$.

For all $m \in \mathcal{M}$ and for properly generated $\{\mathsf{pk}, \mathsf{sk}\}$, the following *correctness* condition holds:

$$\mathcal{D}(\mathsf{sk}, \mathcal{E}(\mathsf{pk}, m)) = m$$

We will denote by $\mathcal{A}$ adversaries, defined in the next section.

A.2 RSA Encryption

In [19], the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk} = (N, e), \mathsf{sk} = d)$ where $N = pq$ is the product of two large primes and e and d are such that $ed = 1 \bmod \varphi(N)$. The encryption of a message $m \in \mathcal{M} = \mathbb{Z}_N^*$, is simply $c = m^e \bmod N$. c can be easily decrypted using d: $m = c^d \bmod N$.

A.3 Security Definitions

Security is traditionally defined by combining an *adversarial goal* and an *attack model*. We refer to classical texts on provable security, such as [18], for precise statements of security definitions. Intuitively, a public-key encryption scheme $\mathcal{P}$ is:

Unbreakable (UBK) If no adversary $\mathcal{A}$ can compute sk given pk.

One-Way (OW) If no adversary $\mathcal{A}$ can recover $m \in \mathcal{M}$ given c and pk.

Indistinguishable (IND) If no adversary $\mathcal{A}$ can derive significant information about $m \in \mathcal{M}$ given only c and pk. IND is sometimes alternatively referred to as "semantically secure".

Non-Malleable (NM) If no adversary $\mathcal{A}$ can, produce, given c and pk, a new ciphertext corresponding to a plaintext m' meaningfully related to m.

In the above, "no adversary" should be read as "an efficient adversary succeeding with a significant probability". The terms "efficiently", "significant probability", "significant information" and "meaningfully related" admit formal definitions which are beyond our scope.

Security notions for encryption schemes are obtained by combining an adversarial goal with an *attack model*:

Chosen-Plaintext Attack (CPA) $\mathcal{A}$ is given nothing more than pk.

Validating-Checking Attack (VCA) $\mathcal{A}$ is given access to a validity-checking oracle indicating only if a given ciphertext is a valid or not (*i.e.* returning the bit $\mathcal{D}(\mathsf{sk}, c) \stackrel{?}{\neq} \perp$). Note that this is *not* the same thing as a Plaintext-Checking Attack (PCA) [18].

Chosen-Ciphertext Attack (CCA) $\mathcal{A}$ has access to a decryption oracle.

Whenever oracle access is used $\mathcal{A}$ cannot submit to the oracle the challenge ciphertext he has to attack. These definitions are classical and we refer the reader to [18] for more details.

We denote security notions positively: *e.g.* OW-VCA[$\mathcal{S}$] is the problem of contradicting the one-wayness of scheme $\mathcal{S}$ under validity-checking attack. This convention permits the easy description of hierarchies between security notions using reductions (see Figure 3). When oracle access is permitted (*i.e.* either for VCA or CCA), we denote by ℓ-GOAL-ATTACK[$\mathcal{S}$] the problem of contradicting $\mathsf{GOAL} \in \{\mathsf{UBK}, \mathsf{OW}, \mathsf{IND}, \mathsf{NM}\}$ in less than ℓ oracle queries under an $\mathsf{ATTACK} \in \{\mathsf{VCA}, \mathsf{CCA}\}$.

$$
\begin{array}{ccccc}
\mathsf{UBK\text{-}CCA} & \Longleftarrow & \mathsf{UBK\text{-}VCA} & \Longleftarrow & \mathsf{UBK\text{-}CPA} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{OW\text{-}CCA} & \Longleftarrow & \mathsf{OW\text{-}VCA} & \Longleftarrow & \mathsf{OW\text{-}CPA} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{IND\text{-}CCA} & \Longleftarrow & \mathsf{IND\text{-}VCA} & \Longleftarrow & \mathsf{IND\text{-}CPA} \\
\Downarrow & & \Downarrow & & \Downarrow \\
\mathsf{NM\text{-}CCA} & \Longleftarrow & \mathsf{NM\text{-}VCA} & \Longleftarrow & \mathsf{NM\text{-}CPA}
\end{array}
$$

Fig. 3. Public-Key Encryption Security Hierarchy

A.4 RSA Security

Single User RSA Security RSA encryption is UBK-CPA-secure assuming the hardness of factoring. Contradicting RSA's OW-CPA security is known as *the* RSA *Problem* and is believed to be intractable.

RSA's determinism makes the validity-checking oracle useless and RSA is therefore OW-VCA-secure relative to the RSA problem as well. Unfortunately, because of this determinism, RSA cannot be semantically secure: given the encryption c of either m_0 or m_1, the adversary simply computes $c_0 = m_0^e \bmod N$ and checks whether $c_0 = c$. Many variants of this basic scheme have been proposed with the hope of making them semantically secure through the use of randomized padding schemes. This paper focuses on all the previously mentioned security notions for the padding PKCS#1 v1.5.

Multiple User RSA Security The previous definitions do not capture security in a multicast context. In the early 1990s, attacks against low-exponent RSA highlighted the danger of multi-user security threats left uncaptured by single-user models. In independent works, Baudron, Pointcheval and Stern [1] and Bellare, Boldyreva and Micali [2] proved that for strong security notions (namely IND or NM), schemes with reductionist security in the multi-user setting are exactly those permitting security proofs in the single-user setting. However, encryption one-wayness (OW) in a multi-user setting is not guaranteed only when the plaintexts are somehow correlated.

For RSA encryption, if a common e is used, then e encryptions of a given message under different public keys $N_1, \ldots, N_e$ lead to an easy plaintext recovery. Again, several countermeasures have been proposed (*e.g.* padding the message with random bits) but with an unclear security guarantees. The second part of this paper explores broadcast attacks on PKCS#1 v1.5.

B Finding Small Modular Roots of a Multivariate Polynomial

The problem of solving modular polynomial equations is believed to be difficult in the general case. Nevertheless, when we restrict the problem to finding small roots only, the problem becomes easier to solve. Indeed, in 1996, Coppersmith [5] introduced a technique, based on lattice reduction, allowing to recover the root of a univariate modular polynomial provided that this root is small enough. This construction was reformulated in simpler terms by Howgrave-Graham [12] and its extensions to more variables found numerous cryptanalytic applications.

B.1 Coppersmith's Technique.

Starting from a polynomial f modulo a known composite integer N, the idea behind Coppersmith's method is to construct a set of polynomials $h_1, \ldots, h_n$ sharing the same sought root over the integers. If the number of these generated polynomials is sufficiently large (greater than the number of variables) and under the assumption that all resultant computations lead to non-zero results, then the root can easily be recovered. Note that this assumption makes the method heuristic.

A sufficient condition ensuring that the polynomials $h_1, \ldots, h_n$ share a common root in $\mathbb{Z}$ was formulated by Howgrave-Graham.

Lemma 2 (Howgrave-Graham [12]). *Let $h \in \mathbb{Z}[x_1, \ldots, x_n]$ be an integer polynomial that consists of at most ω monomials. Suppose that*

1. $h(x_{01}, \ldots, x_{0n}) \equiv 0 \bmod N$ *for some* $|x_{01}| < X_1, \ldots, |x_{0n}| < X_n$
2. $\|h(x_1 X_1, \ldots, x_n X_n)\| < \frac{N}{\sqrt{\omega}}$.

Then $h(x_{01}, \ldots, x_{0n}) = 0$ holds over the integers.

The problem can thus be reduced to finding polynomials $h_1, \ldots, h_n$ of small norm having the same modular root as f. This can be achieved by representing polynomials as coefficient vectors (using a suitable ordering on monomials) and using lattice reduction techniques such as LLL [17] to search for small vectors in a lattice spanned by polynomials which are known to have the sought modular root.

If $\mathcal{L}$ is a lattice of polynomials consisting of at most ω monomials and all having the same modular root as f, then the condition

$$2^{\frac{\omega(\omega-1)}{4(\omega+1-n)}} \det(\mathcal{L})^{\frac{1}{(\omega+1-n)}} < \frac{N}{\sqrt{\omega}} \tag{4}$$

ensures first n polynomials obtained by applying LLL to the lattice $\mathcal{L}$ match Howgrave-Graham's bound. In the analysis, we let terms that do not depend on N contribute to an error term ε, and simply use the determinant condition $\det(\mathcal{L}) \leq N^{w+1-n}$.

B.2 Lattice Construction

A variety of methods for constructing the lattice $\mathcal{L}$ have been proposed in the literature. In what follows, we choose to rely on the technique introduced by Jochemsz and May in [13].

Recall that we have a polynomial f with an unknown root $\boldsymbol{x_0} = (x_{01}, \ldots, x_{0n})$ modulo some composite integer N whose factorization is unknown. This root is *small* in the sense that each of its components is bounded: $|x_{0i}| < X_i$ for $i \in \{1, \ldots, n\}$. We denote by λ the leading monomial of the polynomial f and by $\mathcal{M}(f)$ the set of monomials appearing in f. Of course, λ can be assumed to be monic as otherwise one simply has to multiply f by the modular inverse of its initial coefficient.

Given $\varepsilon > 0$, we fix an integer $\nu = \nu(\varepsilon)$ and without loss of generality we assume that $\mathcal{M}(f^j) \subseteq \mathcal{M}(f^\nu)$ for $j \in \{1, \ldots, \nu-1\}$. If k is an integer between 0 and $\nu+1$, we define the set M_k as $\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k})$ (in particular $M_0 = \mathcal{M}(f^\nu)$ and $M_{\nu+1} = \varnothing$). Next, we define the following *shift polynomials*:

$$g_{i_1 \ldots i_n}(x_1, \ldots, x_n) = \frac{x_1^{i_1} \cdots x_n^{i_n}}{\lambda^k} f^k N^{\nu-k}$$

for $k \in \{0, \ldots, \nu\}$ and $x_1^{i_1} \cdots x_n^{i_n} \in M_k \setminus M_{k+1}$. By definition, all polynomials g have the root $(x_{01}, \ldots, x_{0n})$ modulo N^ν. We can now define $\mathcal{L}$ as the lattice generated by the coefficient vectors of all polynomials $g_{i_1 \ldots i_n}(x_1 X_1, \ldots, x_n X_n)$. If the monomial ordering has been chosen correctly, the matrix corresponding to that lattice is lower triangular and the determinant becomes easy to compute. Indeed, the diagonal elements are those corresponding to the monomial λ^k in f^k for each row. Therefore, the diagonal terms of the matrix are $X_1^{i_1} \cdots X_n^{i_n} N^{\nu-k}$ for $k \in \{0, \ldots, \nu\}$ and $x_1^{i_1} \ldots x_n^{i_n} \in M_k \setminus M_{k+1}$. By doing a simple computation and neglecting low order terms, one can finally reduce the condition (4) to the following new one:

$$\prod_{j=1}^{n} X_j^{s_j} < N^{s_N} \quad \text{for} \quad \begin{cases} s_j = \sum_{x_1^{i_1} \cdots x_n^{i_n} \in M_0} i_j \quad (1 \leq j \leq n) \\ s_N = \sum_{k=1}^{\nu} |M_k| \end{cases} \tag{5}$$

This formula expresses an asymptotic condition on the bounds $X_1, \ldots, X_n$ allowing to recover the root in polynomial time.

Remark 1. The method outlined above is what Jochemsz and May called the "*basic strategy*"; they also proposed an "extended strategy" in which we can use extra shifts of a certain variable and replace M_k for instance by $M_k = \bigcup_{j=1}^{t} x_1^j \left(\mathcal{M}(f^\nu) \cap \lambda^k \mathcal{M}(f^{\nu-k})\right)$ for some well-chosen parameter t.

B.3 The Jochemsz-May Lattice in Broadcast PKCS#1 v1.5

Let us examine what the lattice $\mathcal{L}$ looks like in the particular setting of broadcast PKCS#1 v1.5 encryption.

Recall from section 5.1 that recovering m from $c_1, \ldots, c_\ell$ reduces to finding the root $(x_0, y_{0,1}, \ldots, y_{0,\ell}) = (m, r_1, \ldots, r_\ell)$ of the following modular polynomial:

$$f(x, y_1, \ldots, y_\ell) = u_1(x + Ay_1 + B)^e + \cdots + u_\ell(x + Ay_\ell + B)^e - C \mod N$$

We know that this root satisfies the bounds $|x_0| < X$ and $|y_{0i}| < Y$ for all $i \in \{1, \ldots, \ell\}$ with $X = 2^{8|m|}$ and $Y = 2^{8|r|}$. We examine how the Jochemsz-May bounds described in the previous section translate into bounds on $|m|$ and $|r|$ allowing the message to be recovered in polynomial time.

Form of the Sets M_k. The analysis' first step consists in describing the sets M_k. Note first that the set of monomials $\mathcal{M}(f)$ is included in $\{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\}$. In other words, the geometrical shape of the polynomial f is included in a "pyramid" of dimension $\ell+1$ of monomials with total degree less than e. We choose the *deglex* monomial order, according to which the leading monomial of f is x^e. The sets M_k can then be described as follows:

$$\begin{aligned}
M_0 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu\} \\
M_1 &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu \text{ with } a \geq e\} \\
&\vdots \\
M_\nu &= \{x^a y_1^{b_1} \cdots y_\ell^{b_\ell} \mid a + b_1 + \cdots + b_\ell \leq e\nu \text{ with } a \geq e\nu\}
\end{aligned}$$

which makes it easy to count the number of monomials in each of them.

Condition on the Bounds. Given the above description, we can evaluate the quantities s_j and s_N of equation (5) as follows. First, by symmetry, $s_i, s_{j_1}, \ldots s_{j_\ell}$ are all equal to:

$$s(\nu) = \sum_{i=0}^{e\nu} \frac{i(e\nu - i + 1)(e\nu - i + 2) \cdots (e\nu - i + \ell)}{\ell!}$$

Furthermore, we have:

$$s_N = t(\nu) = \sum_{i=1}^{\nu} \frac{(e\nu - ie + 1)(e\nu - ie + 2) \cdots (e\nu - ie + \ell + 1)}{(\ell + 1)!}$$

Condition (5) can then be rewritten as $X^{s(\nu)} Y^{\ell s(\nu)} < N^{t(\nu)}$, and since N is of byte size ℓk, this gives:

$$\ell |r| + |m| < \frac{t(\nu)}{s(\nu)} \cdot \ell k$$

Asymptotic Bound. The functions $s(\nu)$ and $t(\nu)$ are polynomials in ν. Hence, it suffices to evaluate their leading coefficients to obtain an asymptotic estimate as $\nu \to +\infty$. Note further that $s(\nu)$ and $t(\nu)$ are easily expressed in terms of the antidifference operator, which takes a polynomial $P(X)$ to the polynomial $\sigma(P)(X)$ defined by $\sigma(P)(j) = \sum_{i=1}^{j} P(i)$ for

$j \in \mathbb{N}$. Indeed:

$$s(\nu) = \sum_{i=0}^{e\nu} (e\nu - i)P(i) = e\nu \cdot \sigma(P)(e\nu) - \sigma(XP)(e\nu) + e\nu$$

$$\text{with} \quad P(X) = \frac{(X+1)\cdots(X+\ell)}{\ell!}$$

$$t(\nu) = \sum_{i=1}^{\nu} Q(i) = \sigma(Q)(\nu)$$

$$\text{with} \quad Q(X) = \frac{(eX - e + 1)\dots(eX - e + \ell + 1)}{(\ell+1)!}$$

Now it is easily seen that if the leading coefficient of P is $c_d X^d$, the leading coefficient of $\sigma(P)$ is $c_d X^{d+1}/(d+1)$. It follows that, as $\nu \to +\infty$, we have:

$$s(\nu) \sim e\nu \cdot \frac{(e\nu)^{\ell+1}}{\ell!(\ell+1)} - \frac{(e\nu)^{\ell+2}}{\ell!(\ell+2)} = \frac{(e\nu)^{\ell+2}}{(\ell+2)!} \quad \text{and} \quad t(\nu) \sim \frac{e^{\ell+1}\nu^{\ell+2}}{(\ell+2)!}$$

In particular, $t(\nu)/s(\nu) \to 1/e$ when $\nu \to +\infty$. Thus, the best asymptotic bound on $|m|$ and $|r|$ for which the attack is theoretically possible is:

$$\ell|r| + |m| < \frac{\ell k}{e}$$

C Broadcast Attack Experimental Results

Given PKCS#1 v1.5's widespread use and the heuristic nature of Coppersmith's techniques in the multivariate case, it is important to *practically assess* our attack. In particular, one of the main questions remains to know how many ciphertexts an attacker really needs *in practice* to recover m. In the particular instance $\{\log_2 N = 1024, e = 3\}$, the number of required ciphertexts is, in fact, really low.

Corollary 1. *If a* PKCS#1 v1.5 *user encrypts the same message m with* 64-*bit randomizers to multiple recipients using* 1024-*bit moduli and $e = 3$, then there exists a heuristic polynomial time algorithm that recovers m from $\ell = 4$ ciphertexts.*

Proof. This is a direct consequence of Proposition 4, given that, for 1024-bit moduli and 64-bit randomizers, message size is equal to 936 bits.

These parameters, corresponding to optimal message size and encryption speed for 1024-bit moduli, are quite realistic and widely implemented. The practical implications of this result are potentially serious.

C.1 Partial Information.

Consider an attacker who does not collect all the ℓ required ciphertexts. In that specific case, even if m can not be fully recovered, the attacker can nevertheless obtain partial information on m. In particular, in a scenario where m would not be of full size and would have been previously padded with zero bits (*e.g.* using an AES key), the attack can still be performed.

C.2 Practical Implementations.

To check the applicability of the attack, we investigated three configurations: An attacker having access to two, three and four ciphertexts. Before implementing the attack in each scenario, we first evaluated the dimension of the corresponding lattices (for reasonably small values of the parameter ν) and then expressed the number of bits on m that should be recovered in practice. The results obtained for 1024-bit moduli and $e = 3$ are shown in the following table.

	$\ell = 2$		$\ell = 3$		$\ell = 4$	
ν	$\dim(\mathcal{L})$	$\lvert m\rvert$	$\dim(\mathcal{L})$	$\lvert m\rvert$	$\dim(\mathcal{L})$	$\lvert m\rvert$
2	84	213	210	246	462	249
3	220	306	715	395	2002	451
4	455	359	1820	483	6188	578
5	816	394	3876	542	15504	664
6	1330	418	7315	584	33649	726
7	2024	435	12650	615	65780	773
10	5456	469	46376	675	324632	863

As we can see, the number of bits of m that we are able to recover increases with ν, and approaches 936 bits for $(\ell, \nu) = (4, 10)$. Unfortunately, the dimensions of the constructed lattices are often quite impractical. In many cases, matrix size turns out to exceed 1000, making lattice reduction unfeasible in practice. As a result, and because we had limited processor time at our disposal, we only ran practical experiments in the small cases, namely $\ell = 2$ and $\nu = 2, 3$.

Experiments have been performed on a hepta-processor Intel Xeon clocked at 1.86GHz. Each test was done in the same way: construction of the appropriate lattice, LLL-reduction and then extraction of short vectors. Although one only theoretically requires a number of vectors equal to the number of variables, in practice we decided to take as much vectors as possible to increase the attack's success odds. Most CPU time was claimed by the LLL-reduction step (approximately 3 hours for $(\ell, \nu) = (2, 3)$). With 1024-bit moduli and 64-bit randomizers, we managed to recover a 115-bit message (padded with zero MSBs).

C.3 Toy example.

To better illustrate the attack process, we present here a toy example for 150-bit moduli and 5-bit message m. Once the polynomial f had been constructed and the lattice generated (the lattice is of dimension 84 in this case), LLL-reduction was performed. Here the first 50 vectors corresponded to polynomials having the desired root over the integers. We then took all these polynomials and computed a Gröbner basis of the ideal they generated. The results were the following:

$$N_i = 150 \text{ bits}, m = 5 \text{ bits}, r = 6 \text{ bits}, e = 3, \nu = 2$$
$$f(x, y, z) \text{ with modular root } (24, 58, 34)$$

$$\begin{cases} p_1 = z^4 + 512z^3 + 98304z^2 + 86093442y - 94710946z - 1908346880 \\ p_2 = yz^2 - \frac{89}{81}z^3 + 256yz - \frac{7936}{27}z^2 + 16384y - \frac{573440}{27}z - \frac{33783328}{81} \\ p_3 = y^2 - \frac{62}{27}yz + \frac{961}{729}z^2 - \frac{1024}{27}y + \frac{31744}{729}z + \frac{262144}{729} \\ p_4 = x - 55297y + 63489z + 1048576 \end{cases}$$

The Gröbner basis computation does not allow us to directly recover the message m, since the corresponding subvariety is not of dimension zero, as was usually the case in most experiments. In fact, we commonly faced problems of algebraic dependence between the resulting polynomials (hence our choice to take a large number of polynomials, rather than the first few, to compute the Gröbner basis). Nevertheless, it was usually possible to recover the message, as the polynomials in the Gröbner basis had a very simple form. In this particular case, for instance, p_4 is affine and p_3 can be written as $(ay + bz + c)^2$, making it easy to recover the common root.